OBJECT ORIENTED CONCEPTS

PGDCA-102

BLOCK 1:

INTRODUCTION TO PROGRAMMING WITH JAVA

Dr. Babasaheb Ambedkar Open University
Ahmedabad

OBJECT ORIENTED CONCEPTS

Knowledge Management and
Research Organization
Pune

Acknowledgment
Every attempt has been made to trace the copyright holders of material reproduced in this book. Should an infringement have occurred, we apologize for the same and will be pleased to make necessary correction/amendment in future edition of this book.
The content is developed by taking reference of online and print publications that are mentioned in Bibliography. The content developed represents the breadth of research excellence in this multidisciplinary academic field. Some of the information, illustrations and examples are taken "as is" and as available in the references mentioned in Bibliography for academic purpose and better understanding by learner.'

ROLE OF SELF INSTRUCTIONAL MATERIAL IN DISTANCE LEARNING

The need to plan effective instruction is imperative for a successful distance teaching repertoire. This is due to the fact that the instructional designer, the tutor, the author (s) and the student are often separated by distance and may never meet in person. This is an increasingly common scenario in distance education instruction. As much as possible, teaching by distance should stimulate the student's intellectual involvement and contain all the necessary learning instructional activities that are capable of guiding the student through the course objectives. Therefore, the course / self-instructional material are completely equipped with everything that the syllabus prescribes.

To ensure effective instruction, a number of instructional design ideas are used and these help students to acquire knowledge, intellectual skills, motor skills and necessary attitudinal changes. In this respect, students' assessment and course evaluation are incorporated in the text.

The nature of instructional activities used in distance education self-instructional materials depends on the domain of learning that they reinforce in the text, that is, the cognitive, psychomotor and affective. These are further interpreted in the acquisition of knowledge, intellectual skills and motor skills. Students may be encouraged to gain, apply and communicate (orally or in writing) the knowledge acquired. Intellectual-skills objectives may be met by designing instructions that make use of students' prior knowledge and experiences in the discourse as the foundation on which newly acquired knowledge is built.

The provision of exercises in the form of assignments, projects and tutorial feedback is necessary. Instructional activities that teach motor skills need to be graphically demonstrated and the correct practices provided during tutorials. Instructional activities for inculcating change in attitude and behavior should create interest and demonstrate need and benefits gained by adopting the required change. Information on the adoption and procedures for practice of new attitudes may then be introduced.

Teaching and learning at a distance eliminates interactive communication cues, such as pauses, intonation and gestures, associated with the face-to-face method of teaching. This is particularly so with the exclusive use of print media. Instructional activities built into the instructional repertoire provide this missing interaction between the student and the teacher. Therefore, the use of instructional activities to affect better distance teaching is not optional, but mandatory.

Our team of successful writers and authors has tried to reduce this.

Divide and to bring this Self Instructional Material as the best teaching and communication tool. Instructional activities are varied in order to assess the different facets of the domains of learning.

Distance education teaching repertoire involves extensive use of self-instructional materials, be they print or otherwise. These materials are designed to achieve certain pre-determined learning outcomes, namely goals and objectives that are contained in an instructional plan. Since the teaching process is affected over a distance, there is need to ensure that students actively participate in their learning by performing specific tasks that help them to understand the relevant concepts. Therefore, a set of exercises is built into the teaching repertoire in order to link what students and tutors do in the framework of the course outline. These could be in the form of students' assignments, a research project or a science practical exercise. Examples of instructional activities in distance education are too numerous to list. Instructional activities, when used in this context, help to motivate students, guide and measure students' performance (continuous assessment)

PREFACE

We have put in lots of hard work to make this book as user-friendly as possible, but we have not sacrificed quality. Experts were involved in preparing the materials. However, concepts are explained in easy language for you. We have included may tables and examples for easy understanding.

We sincerely hope this book will help you in every way you expect.

All the best for your studies from our team!

OBJECT ORIENTED CONCEPTS

Contents

BLOCK 1: INTRODUCTION TO PROGRAMMING WITH JAVA

UNIT 1 PROBLEM SOLVING WITH COMPUTERS

Need of programming, Algorithms, Flowcharts, Pseudo-Code, Programming Languages Overview, and Overview of Object Oriented Concepts, Skills check, Exercises

UNIT 2 INTRODUCTION TO JAVA

History, Overview of JVM & Bytecode, Java versions (J2me etc.), Java class library, JDK, Your First Java Application, Skills check, Exercises

UNIT 2 BEGINNING WITH JAVA PROGRAMMING

Java Keywords, Comments in Java, Variables and Assignments, Strings and Characters, Arithmetic Operators and Expressions, Type Conversion in Expressions, Type Conversion, Skills check, Exercises

BLOCK 2: JAVA CONTROL STATEMENTS, OPERATORS AND CLASSES

UNIT 1 JAVA CONTROL STATEMENTS

The if statement, The if-else statement, Blocks of Code, The for statement, Increment & Decrement operators, Backslash codes, Relational and Boolean logical operators, Ternary operators, Skills check, Exercises

UNIT 2 MORE ABOUT CONTROL STATEMENTS AND OPERATORS

Nested if statement, Variations of the for loop, The while loop, The do loop, Nested loops, The break statement, The continue statement, The switch statement, The bitwise operators, Skills check, Exercises

UNIT 3	**CREATING CLASSES**
	The General form of a class, Creating Simple Classes, Adding Constructors, Constructor Overloading, The this keyword, Instance variables and methods, Static variables and methods, Local variables and its scope, Method overloading, Argument Passing, Wrapper Classes, System Class, Garbage Collection, Skills check, Exercises
BLOCK 3:	**INHERITANCE, INTERFACE, PACKAGES AND EXCEPTIONS IN JAVA**
UNIT 1	**INHERITANCE**
	Overview of Re-usability, Subclasses, Inheritance and Variables, Method Overriding, Inheritance and Methods, Inheritance and Constructors, Class Modifiers, Variable Modifiers, Constructor Modifiers, Method Modifiers, Object and Class classes, Skills Check, Exercises
UNIT 2	**INTERFACES AND PACKAGES**
	Interfaces, Interface References, Interface Inheritance, The instanceof Operator, Packages, The import statement, Access control and Packages, Skills Check, Exercises
UNIT 3	**EXCEPTIONS**
	Overview of Exceptions, Exception Handling, Catch Block and searches, The throw statement, Exception and Error classes, The throws clause, Custom exceptions, Skills check, Exercises
BLOCK 4:	**JAVA CLASS LIBRARY, FILE HANDLING AND GUI**
UNIT 1	**INTRODUCTION TO JAVA CLASS LIBRARY**
	Classes of java.util package, Classes of java.net package, Classes of java.lang package, Overview of Collection Framework, Skills check, Exercises

UNIT 2 **FILE HANDLING IN JAVA**

Overview of File Handling, File Class in Java, Using Character Stream Classes, Using Byte Stream Classes, Using Scanner and Console Classes in Java, Skills check, Exercises

UNIT 3 **BUILDING GRAPHICAL USER INTERFACE (SWING)**

Building applications using classes related to Graphical Interface, Creating Layouts in GUI, Using Events in GUI applications, Skills check, Exercises

Dr. Babasaheb
Ambedkar
Open University

PGDCA-102

OBJECT ORIENTED CONCEPTS

BLOCK 1: INTRODUCTION TO PROGRAMMING WITH JAVA

UNIT 1
PROBLEM SOLVING WITH COMPUTERS 03

UNIT 2
INTRODUCTION TO JAVA 22

UNIT 3
BEGINNING WITH JAVA PROGRAMMING 42

BLOCK 1: INTRODUCTION TO PROGRAMMING WITH JAVA

Block Introduction

Programming languages make it easier to design and implement programming ideas, by ensuring that you don't have to learn binary, which is the series of on/off or 1/0 commands the computer uses to calculate everything. The JVM is an abstract instead of actual machine or processor. It specifies certain instructions set, register sets, stack, garbage heap as well as required method. The Java program is a set of instructions which a computer understand.

In this block we will learn and study about concept of programming with basic idea about writing instructions sets. The knowledge about algorithm with its systematic procedures will necessary concept of flowchart will be detailed for use to student. The detailed about Java history and evolution along with pseudo code and byte codes are well explained that will normally be used in any programming language.

After completing this block, students will be able to know more about Java programming and necessary codes involved with basic of operators and variables which can more often be applied anywhere in the program. After this block, students will get knowledge about various types of Java version present as of today.

Block Objective

After learning this block, you will be able to understand:

- Basic about Java programming
- Concept of writing an algorithm
- Idea about Flowchart
- Understanding about pseudo code
- Knowledge about Java virtual machine
- Explanation about Java 2 Platform editions
- Introduction to class library in Java
- Idea about Java Comments

- Types of variables in Java
- Knowledge about Java Expressions

Block Structure

Unit 1: Problem Solving with Computers

Unit 2: Introduction to Java

Unit 3: Beginning with Java Programming

UNIT 1: PROBLEM SOLVING WITH COMPUTERS

Unit Structure

1.0 Learning Objectives

1.1 Introduction

1.2 Need of Programming

1.3 Algorithms

1.4 Flowcharts

1.5 Pseudo-Code

1.6 Programming Languages Overview

1.7 Overview of Object Oriented Concepts

1.8 Skills Check

1.9 **Exercises**

1.10 Let Us Sum Up

1.11 Answers for Check Your Progress

1.12 Glossary

1.13 Assignment

1.14 Activities

1.15 Case Study

1.16 Further Readings

1.0 Learning Objectives

After learning this unit, you will be able to understand:

- Study the need of programming
- Features of Algorithms
- Basic of Flowcharts
- Know about Pseudo-Code
- Overview of Programming Languages

- Overview of Object Oriented Concepts

1.1 Introduction

Programming is writing of instructions sets which will guide the computer how to do certain work. Doing work relates to reading list of names from a file which could be alphabetical and writing it back again to the file. As seen, it could be much more complex as it involves displaying of graphical user interface which could mainly mean for games and other game's logic. Since, it is analysed that Java is relatively a current object-oriented programming language which has nowadays achieved more popularity and is easy to apply. It is a general-purpose language that can be used for many types of programming tasks.

1.2 Need of Programming

With programming languages, the commands break into generalised and simplified form. The comparison of various programming languages involves physical tasks which normally not require certain natural talent or skill which could be like playing outdoor games or doing painting or singing. Simply, a programming needs care and art of writing certain particular scripts called as codes.

Nowadays, computers do not really think as these are mathematical machines which can perform calculations which can be required by certain programming languages to be used in various software programmes.

Check your progress 1

1. Programming language involves:

a. computer	c. software
b. codes.	d. all

1.3 Algorithms

An algorithm is a series of definite number of steps which is shown in particular order so as to generate the solution for required problems. In computing, algorithms are essential because they serve as the systematic procedures that computers require. A good algorithm is like using the right tool in the workshop. It does the job with the right amount of effort.

The needs for algorithms can be:

- Input
- Generates an output
- Unambiguous
- Generality
- Correctness
- Finiteness
- Efficiency

Advantages of Algorithms:

There are following advantages for using an algorithm:

Efficiency: In many problems, process such as sorting is most commonly appears in computing. A good speed algorithm should be applied to solve such problems that should consider the time and cost involved.

Abstraction: The algorithms should carry certain concept involved in solving problems which purify into easy ones. Using concept, a problem can be solved by breaking into simpler levels.

Reusability: As seen many commonly known algorithms are generalizations of difficult ones and many complicated problems can be purify into simpler ones, an effective way of solving many simpler problems will allow us to find many difficult problems.

It is studied that an algorithm is a commonly defined computational process that carries set of instructions which contains some value or may be set of values in shape of input and generates set of values as output. The algorithm accepts data, manipulate the data and gives the desired values as output as shown in fig 1.1.

Input
Algorithm
Output

Fig 1.1 Algorithm process

There are certain properties of an algorithm.

1. **Algorithm should be exact -** It is seen that an algorithm should be specific and particularly be explained in such a way that there remains no doubt.

2. **Algorithm should terminate -** The scope of algorithm concerns with solution of certain problem that can be solved on execution. So there should be finite number of steps involved in an algorithm.

3. **Algorithm should be effective -** It is seen that an algorithm should be straight and clear that results in correct output of a problem.

4. **Algorithm should be general -** Normally an algorithm is able to solve every part of problem. It should stress on whats, and not hows, apart from details for program version.

Check your progress 2

1. The comment in an Algorithm begins with_______.

a. /*	c. */
b. /	d. //

1.4 Flowcharts

A Flowchart is a graphical representation of an algorithm or its process. In this, every step in the process is shown by symbol that gives short description of the process. The symbols in the flow chart are connected together with arrows which show the flow of process in the particular direction. It typically describes the flow of data in a process which shows operations/steps in shape of pictures format that is convenient to understand in textual format.

A flowchart shows the process of operations that works in sequence in order to solve a particular problem. To understand the process, the flowchart gives you the way out step by step. Since it is the pictorial or graphical representation of a process, the idea behind flow chart is to communicate how a process should work.

Basically, flowchart analyses design and manage a process or program in various fields. Fig 1.2 shows the generic flowchart.

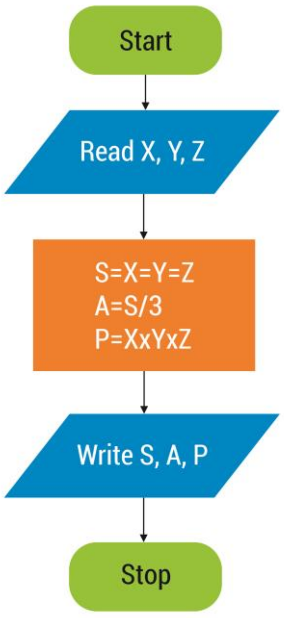


Fig 1.2 Flowcharts

It is studied that flowcharts are normally drawn in the beginning of making a computer solution. It serves as a communication tool among the programmers and people. With this, the programming of a particular problem is done that is useful in understanding complicated logic. Once the flowchart is drawn, it becomes easy for programmer to design a program in any high level language.

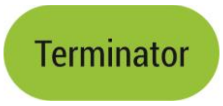
Advantages:

- It is a good way of communicating the logic of a system.
- It is helpful in analysing problems in an effective way.
- It serves as effective program documentation that is required for various jobs.
- It is a kind of blueprint of a particular program which is helpful in development phase.
- It is effective tool for debugging a process.
- It helps the programmer to show efforts efficiently on particular part of program.

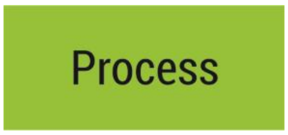
Flowchart Symbols:

To draw a flowchart of a particular program or process, you need some symbols to represent it. Some of the standard symbols that are required for drawing flowchart are:

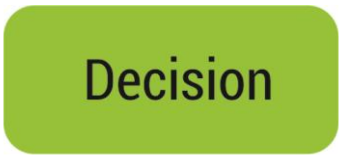
Terminator: It is an oval shape symbol that shows start or end process.



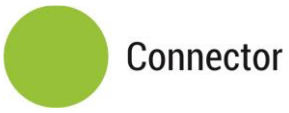
Process: It is a rectangular shaped symbol which shows normal/generic process flow step.



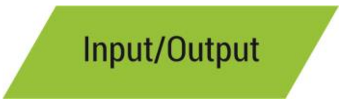
Decision: This is of diamond shaped symbol which shows a branch in the process flow. It is applied when you need to have a decision which is in shape of Yes/No question or True/False.



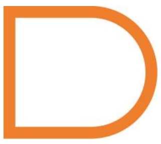
Connector: This is of circular shaped symbol which shows a jump in the process flow.



Data: This is represented as parallelogram shape symbol which shows data input or output (I/O) for a process.



Delay: This is like D shaped symbols which shows a delay or wait in particular process for input to other process.



Arrow: The arrow shows direction of control flow in a process. Arrow coming from one symbol and entering into another symbol will shows the passing of control to the symbol.

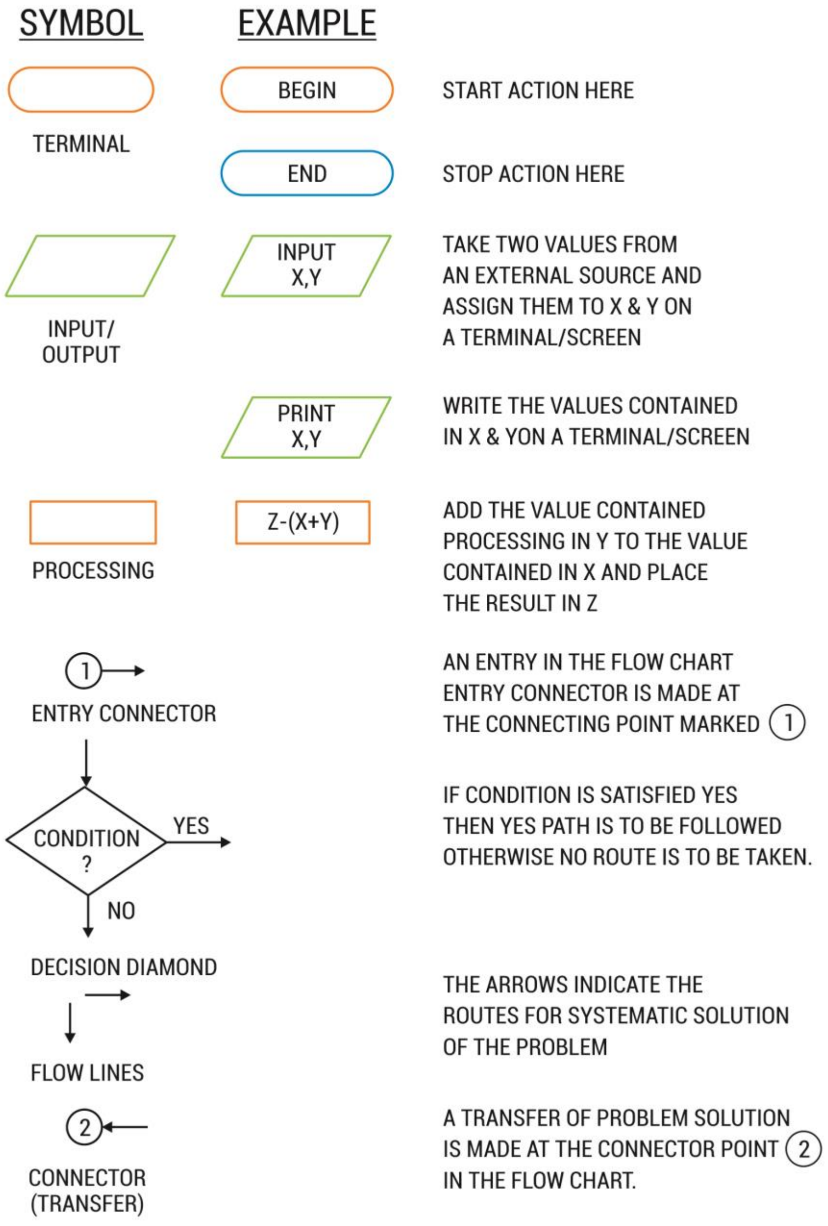

SYMBOL
EXAMPLE
TERMINAL
BEGIN
START ACTION HERE
END
STOP ACTION HERE
INPUT/ OUTPUT
INPUT X,Y
TAKE TWO VALUES FROM AN EXTERNAL SOURCE AND ASSIGN THEM TO X & Y ON A TERMINAL/SCREEN
PRINT X,Y
WRITE THE VALUES CONTAINED IN X & YON A TERMINAL/SCREEN
PROCESSING
Z-(X+Y)
ADD THE VALUE CONTAINED PROCESSING IN Y TO THE VALUE CONTAINED IN X AND PLACE THE RESULT IN Z
1
ENTRY CONNECTOR
AN ENTRY IN THE FLOW CHART ENTRY CONNECTOR IS MADE AT THE CONNECTING POINT MARKED 1
CONDITION ?
YES
NO
DECISION DIAMOND
IF CONDITION IS SATISFIED YES THEN YES PATH IS TO BE FOLLOWED OTHERWISE NO ROUTE IS TO BE TAKEN.
FLOW LINES
THE ARROWS INDICATE THE ROUTES FOR SYSTEMATIC SOLUTION OF THE PROBLEM
2
CONNECTOR (TRANSFER)
A TRANSFER OF PROBLEM SOLUTION IS MADE AT THE CONNECTOR POINT 2 IN THE FLOW CHART.

Fig 1.3 Flowchart Symbol

Check your progress 3

1. In a flowchart, process can be represented by:

   a. circular shaped object

   b. rectangular shaped object

   c. diamond shaped object

   d. none of above

1.5 Pseudo-Code

Pseudo-code is a tool which is used for planning program logic. Pseudo-code is an artificial and informal language that helps programmers develops algorithms. Pseudo-code is a "text-based" detail (algorithmic) design tool.

To repeat an instruction, often it is required that all data should be processed in continuation in loop. Loop basically states for as a repetition or iteration of control structure. These codes are made up of following basic logic structures which are used for writing any computer program. These are:

1. Sequence
2. Selection (IF…THEN…ELSE or IF…THEN)
3. Iteration (DO…WHILE or REPEAT…UNTIL)

Sequence

It is a logic applied for performing instructions that will work one after another in a particular series. So, in sequence logic, pseudo-code instructions are developed in order or sequence where they are used. Logically, the flow of pseudo-code is from top to bottom. Figure 1.4 shows an example of sequence logic structure.

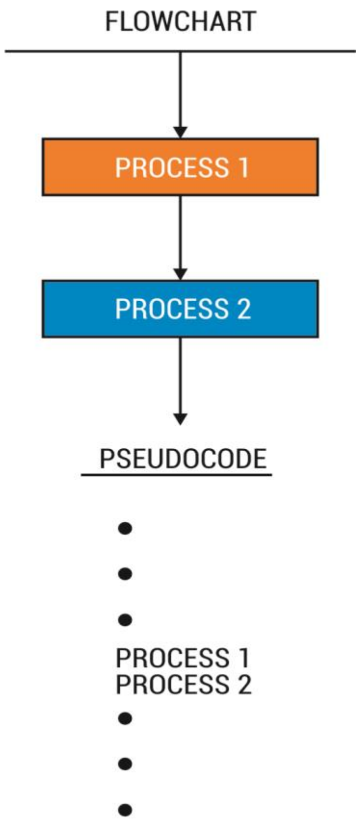
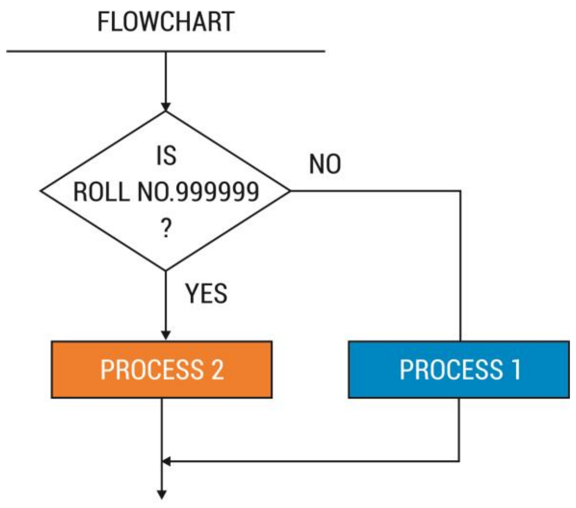


Fig 1.4 pseudo-code sequence logic structure

Selection

Such logic is also called as decision logic where the need of decision making is done. This logic will choose the particular path from two or more optional paths located inside the program logic. This logic is either called as IF...THEN...ELSE structural logic or IF.....THEN structural logic. Figures 1.4 and 1.5, shows such type of logical structures with their respective pseudo-code.

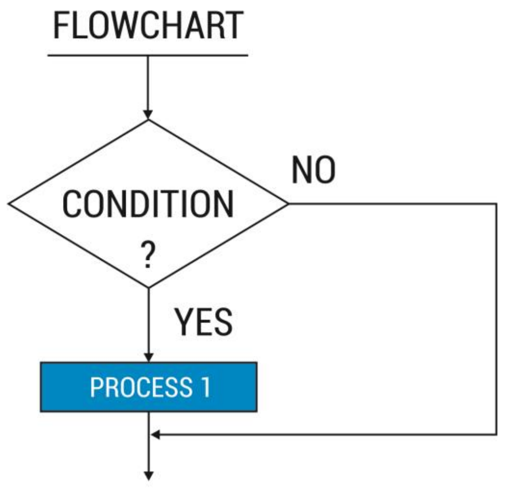


PSEUDOCODE

```
●
●
●
If Condition
    THEN  process 1
  ELSE
              process 2
END IF
●
●
●
```

Fig 1.5 pseudo-code for If-Then-Else structure

In fig 1.5, the IF...THEN structure shows that when the condition is true, then process 1 will work and if this condition is not true, then the process will skip. As seen, the process 1 and process 2 can be one or more processes.



PSEUDOCODE

```
●
●
●
If Condition
   THEN process 1
END IF
●
●
●
```

Fig 1.6 pseudo-code for If-Then selection structure

Check your progress 4

1. The purpose of Pseudo-code is to:

a. plan for program logic	c. prepare a software
b. design logic program	d. all of above

1.6 Programming Languages Overview

It is a type of computer generated language which is used by language programmers to develop applications, scripts or set of instructions so that a computer can carry out. Out of all the most basic language is machine language or a low level language which uses binary numbers 1 and 0 which a computer can easily understand and run without the use of translator or interpreter. There are many high-level languages such as Basic, C and Java which are simple and are like English which requires a compiler or an interpreter to change high-level code into the machine code. Because of this conversion, these languages are slower.

There are varieties of programming languages which are also called as computer language.

Check your progress 5

1. In programming language, the basic language is:

a. machine language	c. both a and b
b. low level language	d. none of the above

1.7 Overview of Object Oriented Concepts

Object-oriented technology is very wide and extensive. The users of computer systems found the effects of such technology as increasingly easy-to-use software applications and operating systems which are flexible that is catered by several industries such as banking, telecommunications and television. In case of software technology, such object-oriented technology will cover object-oriented management of projects, computer hardware and computer aided software engineering, among others. There are certain concepts that laid the Object-oriented technology to spread its scope which can be:

Concept: Object behaviour

In earlier programming, we have:

- Data which is inactive
- Functions which will handle any sort of data

It is seen that an object:

- Carries both data and methods which will handle every data
- Active and not passive which does things
- Charges for its own data
- Exploits data to other objects

Concept: Object State

It is seen that in an object state:

- Both data and methods helps to control data
- Data has definite state of object

- Shows relationships among several objects

Concept: Object Classes

It found that in an object class:

- Every object shows a class
- Object has field or variables
- Field is shown by class
- Object carries methods
- Class represents certainty of such methods
- Class works as template or cookie cutter

Concept: Classes as Abstract Data Types

In an Abstract Data Type:

- Data is shown by object or thing
- Operations are done on data

Concept: Classes form a hierarchy

In case of classes as hierarchy:

- Classes presents as tree like structure
- Root class is known as object
- Every class excluding object is a superclass
- Carries old object
- To define a class, you need to specify as superclass
- If superclass is not defined, it means object
- Every class has one or more subclasses

Concept: Objects from Superclasses

A class can be called as fields or methods

Objects of such class contains fields and methods

Object contains:

- Fields which are in class's superclasses
- Methods which are in class's superclasses

It is not a detailed for an object.

Concept: Created Objects

It is seen that an n integer will performs:

- Declaration of n integer variable
- Allocation space keeping values for n

Concept: Variables to hold subclass objects

If B is a subclass of A, then:

- A objects will allocate to A variables
- B objects will allocate to B variables

B objects will allocate to A variable, but A objects cannot be allocated to B variable.

All B objects is A but not all A object is B

So bVariable = (B) aObject;

Concept: Constructors make objects

In this:

- Every class has a constructor which generates an object
- Keyword new applies to call constructor as secretary = new Employee ( )
- Wring of a constructor if allowed

Java gives default constructor having no arguments by setting up of new object to 0 and if this works out, then a constructor cannot be written, so syntax for writing constructors will remain similar for writing methods.

Check your progress 6

1. The object oriented technology includes:

   a. object oriented management of projects

   b. computer hardware

   c. computer aided software engineering

   d. all of above

1.8 Skills Check

Programming skills will allow someone to test proficiency level that exists in different programming languages. There are certain programming languages such as C, Java and .Net which allows various skills to test. To check your skills you need to work on certain small programmes that can be evaluated and test with related softwares. To know about better skills the need for good programming skills are essential.

Check your progress 7

1. Programming skills can be checked:

a. by writing a programme

b. by testing a program

c. by working on software

d. all of these

1.9 Exercises

Example 1:

Write an algorithm and flowchart for computing the average number of the default 4 numbers.

Algorithm:

```
Start-
---------------------------------------------------
Input A,B,C,D
AVERAGE = (A + B + C + D) / 4
Print AVERAGE
---------------------------------------------------
End
CLS
CLEAR
INPUT a, b, c, d
avr = (a + b + c + d) / 4
```

```
PRINT avr

END
```

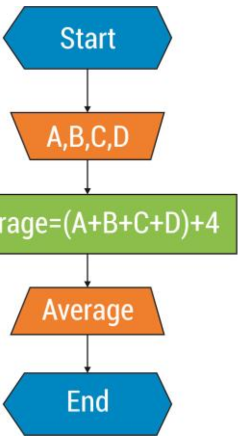


Fig 1.7 Example Flowchart

Check your progress 8

1. What will be output of the program?

```
int x = 1, y = 6
while (y--)
{
      x++;
}
System.out.println("x="+x"y="+y)
```

a. x = 1 y = 0

b. x = -7 y = 6

c. x = 3 y = -4

d. compilation error

1.10 Let Us Sum Up

In this unit we have learnt that a programming is a sort of writing instructions sets that guides computer how to do certain work

It is studied that an algorithms are essential as they serve as systematic procedures which computers require.

It is seen that an algorithms carries certain concept which is involved in solving problems which purify into easy ones.

A Flowchart is a graphical representation of an algorithm or its process in which the diamond shaped symbol represents process flow.

It is found that pseudo-code is a tool which is used for planning program logic.

In case of software technology, such object-oriented technology will cover object-oriented management of projects, computer hardware and computer aided software engineering, among others.

1.11 Answers for Check Your Progress

Check your progress 1

Answers: (1-d)

Check your progress 2

Answers: (1-d)

Check your progress 3

Answers: (1-b)

Check your progress 4

Answers: (1-a)

Check your progress 5

Answers: (1-c)

Check your progress 6

Answers: (1-d)

Check your progress 7

Answers: (1-d)

Check your progress 8

Answers: (1-d)

1.12 Glossary

1. **Algorithm -** It is a series of fixed number of steps that are arranged in particular order.
2. **Programming -** It is writing of instructions sets that will guide computer how to do certain work.
3. **Efficiency -** It determines the speed at which an algorithm be able to solve problems.
4. **Abstraction -** It involves solving of certain program by breaking up into simpler levels.
5. **Pseudo-code -** It is a tool which is used for planning program logic.

1.13 Assignment

Write short note of necessity of programming?

1.14 Activities

Calculate the output of the following program?

```
int I = 0;
outer;
while(true)
{
    I++;
    Inner;
    For (int j = 0; j < 10; j++)
    {
    I += j;
```

```
        If(j == 3)
        continue inner;
        break outer;
        }
continue outer;
}
System.out.println(I);
```

1.15 Case Study

Compare and discuss the Java programming operators?

1.16 Further Readings

1. Learning Programming by Peter Norvig's
2. Approach programming with a more positive by P. Brian. Mackey

UNIT 2: INTRODUCTION TO JAVA

Unit Structure

2.0 Learning Objectives

2.1 Introduction

2.2 History

2.3 Overview of JVM and Bytecode

2.4 Java Versions (J2me etc.)

2.5 Java Class Library

2.6 JDK

2.7 Your First Java Application

2.8 Skills Check

2.9 Exercises

2.10 Let Us Sum Up

2.11 Answers for Check Your Progress

2.12 Glossary

2.13 Assignment

2.14 Activities

2.15 Case Study

2.16 Further Readings

2.0 Learning Objectives

After learning this unit, you will be able to understand:

- Overview of JVM and Bytecode
- Basic of Java versions
- Study of Java class library
- Know-how of JDK
- First Java Application

2.1 Introduction

Java is a programming language created by James Gosling from Sun Microsystems in 1991. The first publicly available version of Java was released in 1995 as Java 1.0. This is commonly used programming language that has variety of uses and applications which makes them to use in desktop applications, Mobile Applications, Enterprise applications etc.

Java is:

- Class based and object oriented programming language
- Computing platform
- Fast, secure, and reliable
- Free
- General purpose
- Concurrent

The features of Java language are:

- It is a simple object oriented language and is quiet familiar.
- It is quiet protected and rough and tuff language.
- It is quiet portable and has varied architecture.
- It is quiet portable and has varied architecture.
- It works with high performance.
- It is interpreted, threaded and dynamic.

2.2 History

Java was initially framed by Sun Microsystems in the beginning of 1990s. The language was created with the idea to solve problem related to connectivity of several household machines together. As it was designed for special connectivity purpose only, so the particular project was failed as no one wants to use it. Initially the name of Java is OAK.

James Gosling worked initially on Java and was known as father of Java. The name OAK was renamed as Java in the year 1994. During the mid of year 1995 in month of May, Java was publicly released. Java was targeted for Internet development. The release of applets was initially supported by big companies such as Netscape Communications. Some of the Java Versions are shown in table

Versions of Java

Java Version	**Release Date**	**Year**
JDK 1.0	January 21	1996
JDK 1.1	February 19	1997
J2SE 1.2	December 8	1998
J2SE 1.3	May 8	2000
J2SE 1.4	February 6	2002
J2SE 5.0	September 30	2004
Java SE 6	December 11	2006
Java SE 7	July 28	2011

Check your progress 1

1. OAK was renamed as Java in year:

a. 1995 c. 1997

b. 1996 d. 1994

2.3 Overview of JVM and Bytecode

JVM

A JVM is called as Java virtual machine which interprets assembled Java binary code which is called as bytecode for computer's processor to do certain Java program instructions. The idea of inventing Java is only to allow application programs which possibly could work on any platform without the need of rewriting it or compiling again by programmer for individual platform. With the help of Java virtual machine, it is possible as it carries particular instruction lengths and related speciality of the platform.

Once the machines have been implemented for a particular platform, any type of Java program will be able to work on such base. Java virtual machine will either understand the bytecode per instruction at a time or by accumulating bytecode for actual processing with the help of just-in-time compiler.

It is studied that a Java virtual machine (JVM) as shown in fig 2.1 is a software implementation of computer which will carry out programs such as real machine. The Java virtual machine is designed particularly for particular operating system.

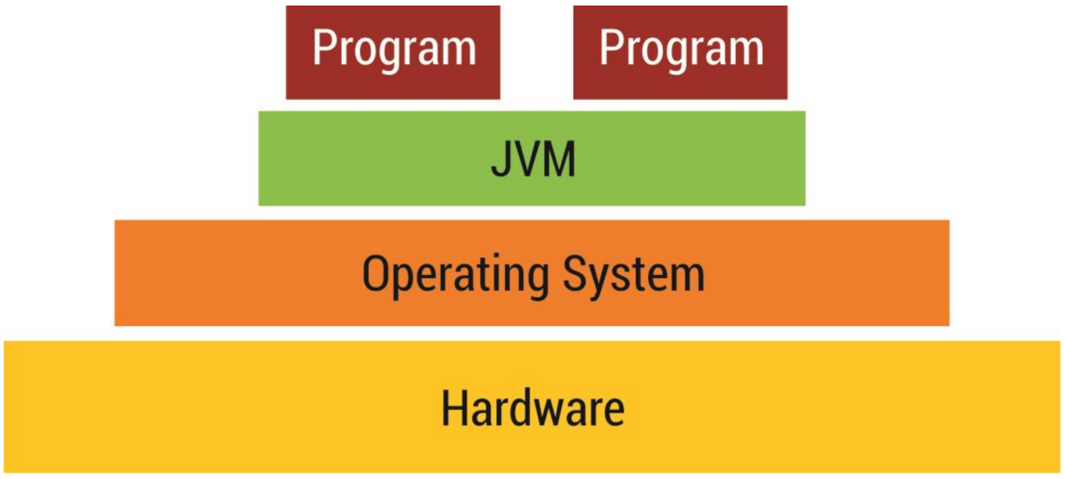


Fig 2.1 Java Virtual Machine

Bytecode

It is a computer object code which is obtained by a program machine called as virtual machine instead of actual computer machine as the processor. Such type of machine will convert standard machine instruction into particular machine instruction or instructions which a computer processor is able to understand. The result of such compilation is bytecode. The bytecode is the result obtained after compilation of source code which is written in language that is able to handle by such machine. Many computer languages like C and C++, requires an individual compiler for every computer platform like for all computer operating system along with its hardware set of instructions.

It is seen that the best known language at present which uses bytecode and virtual machine approach is Java. Apart from this, LISP language applied in artificial intelligence applications was also an earlier language which was compiled by bytecode.

Instead of understanding single instruction at a particular time, Java bytecode will be able to recompile at every system platform with the use of just-in-time compiler which will make the Java program to run faster. It is found that in Java, the bytecodes are present in binary file that carries .CLASS suffix.

Check your progress 2

1. The full form of JVM is called as:

a. Java virtual machine	c. Java vertical machine
b. Java visual machine	d. None of above

2.4 Java Versions (J2me etc.)

Java is formally called as Java 2 Platform which carries three editions:

- Java 2 Standard Edition (J2SE)
- Java 2 Enterprise Edition (J2EE)
- Java 2 Micro Edition (J2ME)

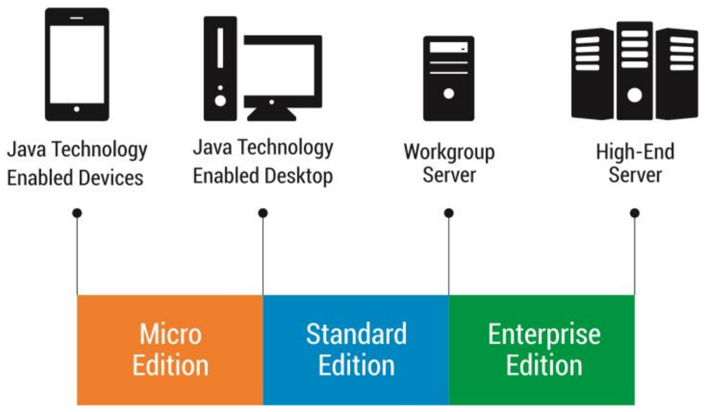


Fig 2.2 Java Editions

All these three editions will focus on different kinds of applications which run on different devices. It is found that:

- Desktop based applications were developed with the help of J2SE that carries necessary user interface classes.
- Server based applications was formed with the help of J2EE that stress more on component based programming as well as deployment.
- Handheld along with embedded devices are formed with the help of J2ME.

In the year 1995, the JDK 1.0 exists which was upgraded to JDK 1.1 and to Java 2 in the year 1999.

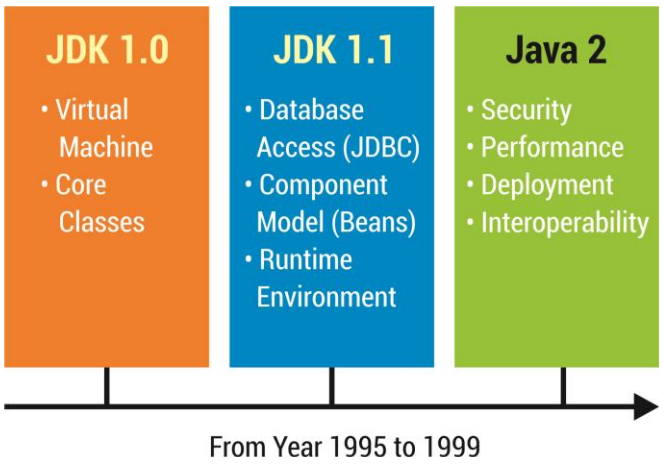


Fig 2.3 JDK

It is seen that there exists single Java platform with multiple profiles such as:

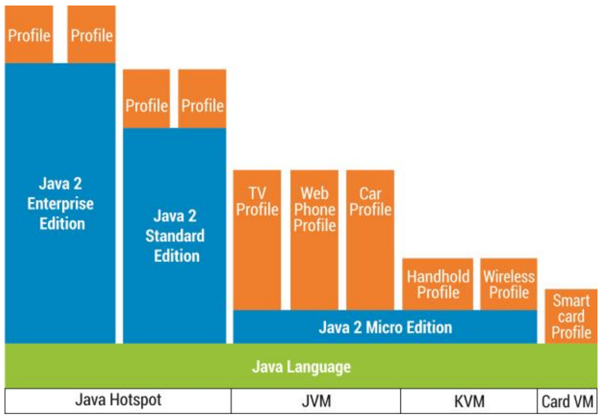


Fig 2.4 Java Platforms

J2ME

The Java Platform Micro Edition which is also known as Java ME, is a platform which is designed for particular embedded systems. In this, the target devices such as industrial controls to mobile phones and set-top boxes are present. It is seen that Java ME was earlier called as Java 2 Platform which is Micro Edition as J2ME.

As studied, the edition of Java ME was initially invented by Sun Microsystems which was made advanced by Oracle Corporation who named it as Personal Java. Earlier, the different brands of Java ME have evolved in different JSRs. During the month of December 2006, Java ME source code was licensed under GNU which was released as phone ME.

J2SE

Since Java is a dynamic programming languages used by computer programmers today, this language carries advance features with its current edition on Java 2 Platform which is the standard edition called as J2SE. This edition is mainly used for writing applets and other applications.

The main advantage of J2SE edition is that it is used in development of certain Java applications that are utilised for single computers. The J2SE edition applets and several other applications allow such functions to run smoothly. In the absence of such applications, various transactions and several Internet interactions will not takes place? With this, the edition is of a great enabler of carrying web activity.

J2EE

In order to lower the costs and fast track application design and development, Java 2 Platform Enterprise Edition called as J2EE shows a component based mechanism so as to construct, develop, assemble and deploy enterprise applications. Such platform uses multitier distributed application model. It is studied that application logic is framed into parts as per the function with certain application components which makes J2EE application to be kept on various machines according to the tier present in the multitier J2EE environment as per which the application belongs.

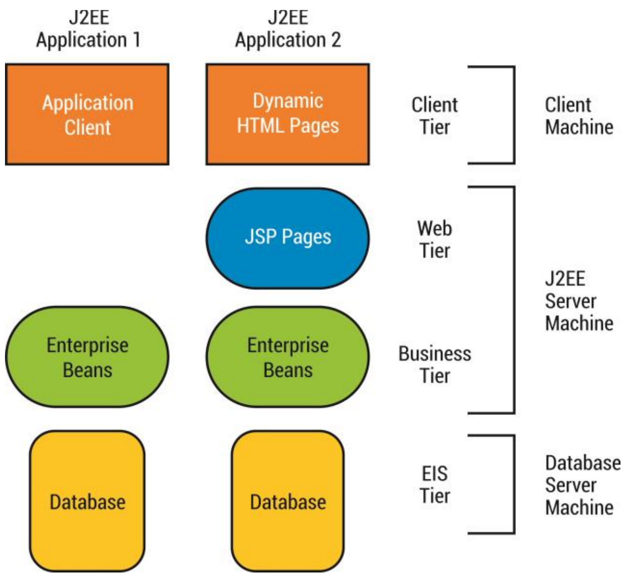


Fig 2.5 J2EE Multitier Applications

Figure 2.5 shows two multitier J2EE applications divided into the tiers described above. The parts shown are presented in J2EE Components as:

- Client-tier components run on the client machine.
- Web-tier components run on the J2EE server.
- Business-tier components run on the J2EE server.

- Enterprise information system (EIS)-tier software runs on the EIS server.

Check your progress 3

1. Which is a multitier distributed application model?

a. J2ME	c. J2EE
b. J2SE	d. all of above

2.5 Java Class Library

It is found that there are many classes available in the library in Java. A wide range of extensive library of pre-written classes which can be applied in certain programs are listed below which are grouped into Java 1.1 packages.

• Java.applet	java.awt
• Java.awt.data transfer	java.awt.event
• Java.awt.image	java.awt.peer
• Java.beans	java.io
• Java.lang	java.lang.reflect
• Java.math	java.net
• Java.rmi	java.rmi.dgc
• Java.rmi.registry	java.rmi.serve
• Java.security	java.security.acl
• Java.security.interfaces	java.sql
• Java.text	java.util
• Java.util.zip	

In this we find that every package defines a number of classes, interfaces, exceptions and errors. Also, the packages further gets splitted into sub-packages, as in case of java.lang package, which has sub-package as java.lang.reflect. It is seen that a class in sub package will not approach to a class located inside parent package. It is seen that java.net package having interfaces, classes as well as exceptions are shown below:

Interfaces in java.net

- ContentHandlerFactory
- FileNameMap
- SocketImplFactory
- URLStreamHandlerFactory

Classes in java.net

• ContentHandler	DatagramPacket
• DatagramSocket	DatagramSocketImpl
• HttpURLConnection	InetAddress
• MulticastSocket	ServerSocket
• Socket	SocketImpl
• URL	URLConnection
• URLEncoder	URLStreamHandler

Exceptions in java.net

- BindException
- ConnectException
- MalformedURLException
- NoRouteToHostException
- ProtocolException
- SocketException
- UnknownHostException
- UnknownServiceException

Check your progress 4

1. In ________package, gc() method is present.

a. java.lang c. java.awt

b. java.util d. java.io

2.6 JDK

JDK also known as Java Development Kit is a program development environment which is basically employed for writing Java applets and applications. It comprises of runtime environment which remains on top of an operating system layer along with tools and programming which is required by the developer in order to compile, debug and run applets as well as applications that are written in Java language.

It is found that JDK carries JRE and Javac tools which will allow the development as well as execution of certain Java applications as well as applets. It carries lots of important useful tools like:

- **Applet viewer:** It runs and debugs Java applets without Web browser.
- **Javadoc:** It will extract comments in particular format from Java source files which will form an HTML file tree.
- **Jar:** It is a sort of archive tool which combines various files into single .jar archive file.
- **Jarsigner:** It is a Java application that will produce signatures for .jar files which will verify signatures of particular signed .jar files.
- **Javaws:** This is software that downloads and run Java applications from Web.
- **pack200:** It is another Java application which will transform .jar file to compressed pack200 file with the help of Java's gzip compressor.

Check your progress 5

1. _______will run and debug Java applets without the use of Web browser.

a. Applet viewer	c. Jar
b. Javadoc	d. Jarsigner

2.7 Your First Java Application

To write your First Java program, you have to first install Java Development Kit. After installation of JDK, you can create java program by using a class having main method as shown:

- Intitially install JDK.

- Now set path of jdk/bin directory.
- Create java program
- Compile and run java program
- Finally the program will create first hello java example.

To create hello java program, write the code as:

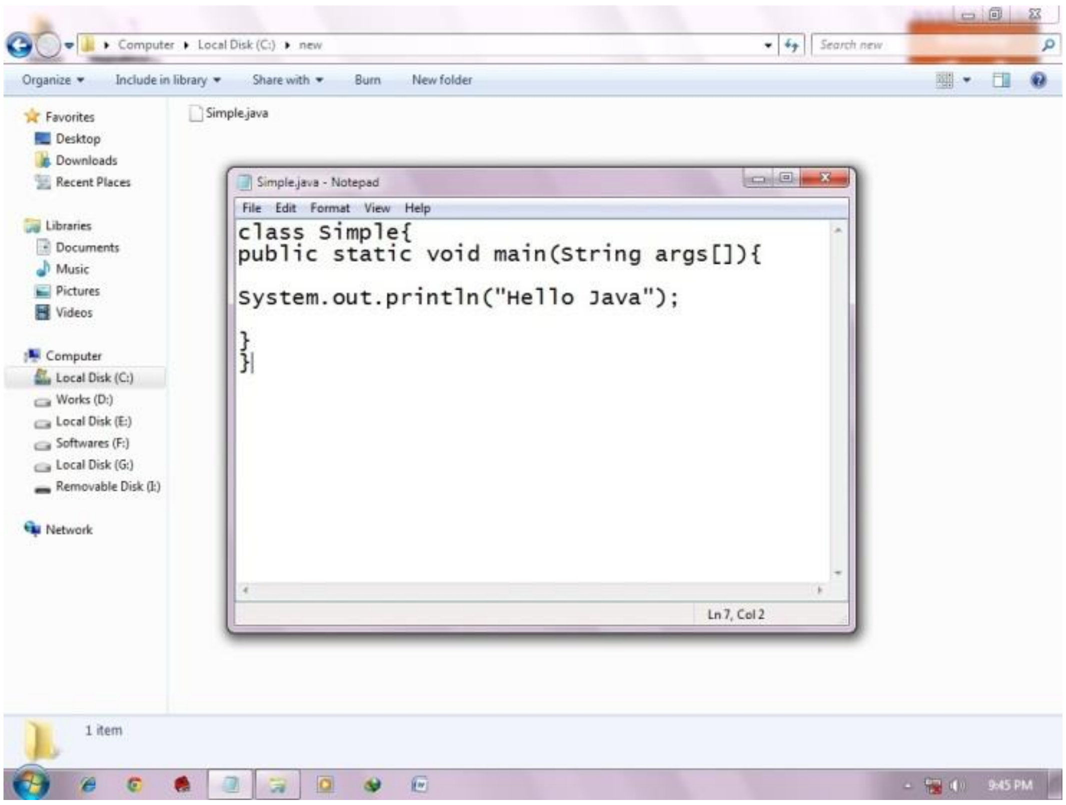
```
        Class simple{
        Public static void main (String args []){
        System.out.println(“Hello Java”);
        }
}
```

Save this file as Simple.java. You will find that when you run the above code, then the compiled output will show cannot open shared object file; No such file or directory.

To understand the first java program, you need to understand the meaning of class, public, static, void, main, String[], System.out.println().

- class: It is applied to declare a class in java.
- public: It is an access modifier that shows visibility to all.
- static: It declares certain statics method which is not required to fonn an object to invoke such method. It is done by JVM hence doesn't require an object to invoke further saves memory.
- void: It is a return type of method which doesn't returns any value.
- main: It shows startup of a program.
- String': It is applied for command line argument.
- System.out.printlnO: This command is applied for printing statement.

You can write a simple Java program in a notepad by selecting start menu----> All Programs -----> Accessories ------> notepad as shown:

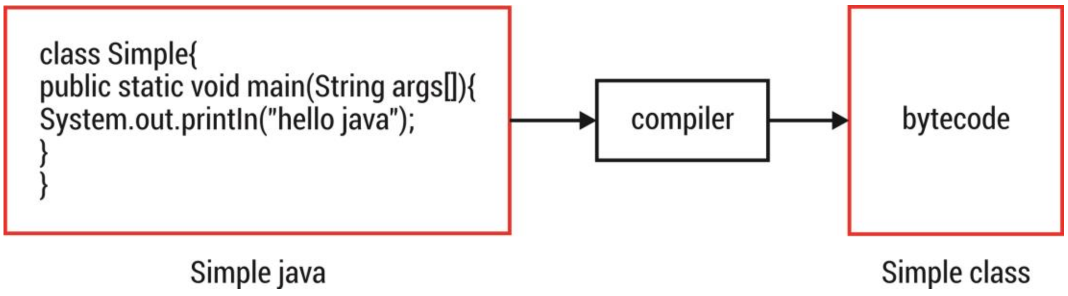


```
class Simple{
public static void main(String args[]){

System.out.println("Hello Java");

}
}
```

Fig 2.6 Simple Java Program in Notepad

Now in order to compile and run the above program, you need to open command prompt by clicking on start menu ------> Choosing All Programs -----> Selecting Accessories --------> command prompt.

When you compile a program, then the Java File converts Java code into bytecode as shown:



```
class Simple{
public static void main(String args[]){
System.out.println("hello java");
}
}
```

Fig 2.7 Compilation Process

Also after the compilation, the above program will run with a process as shown in flowchart below:

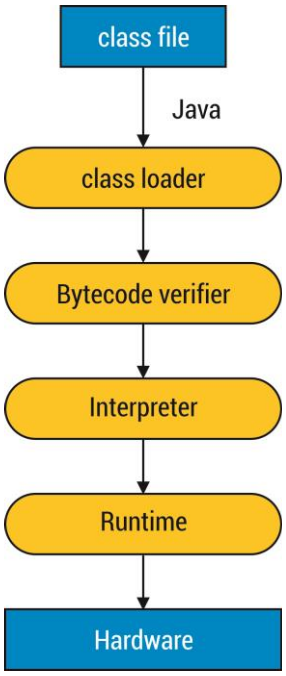
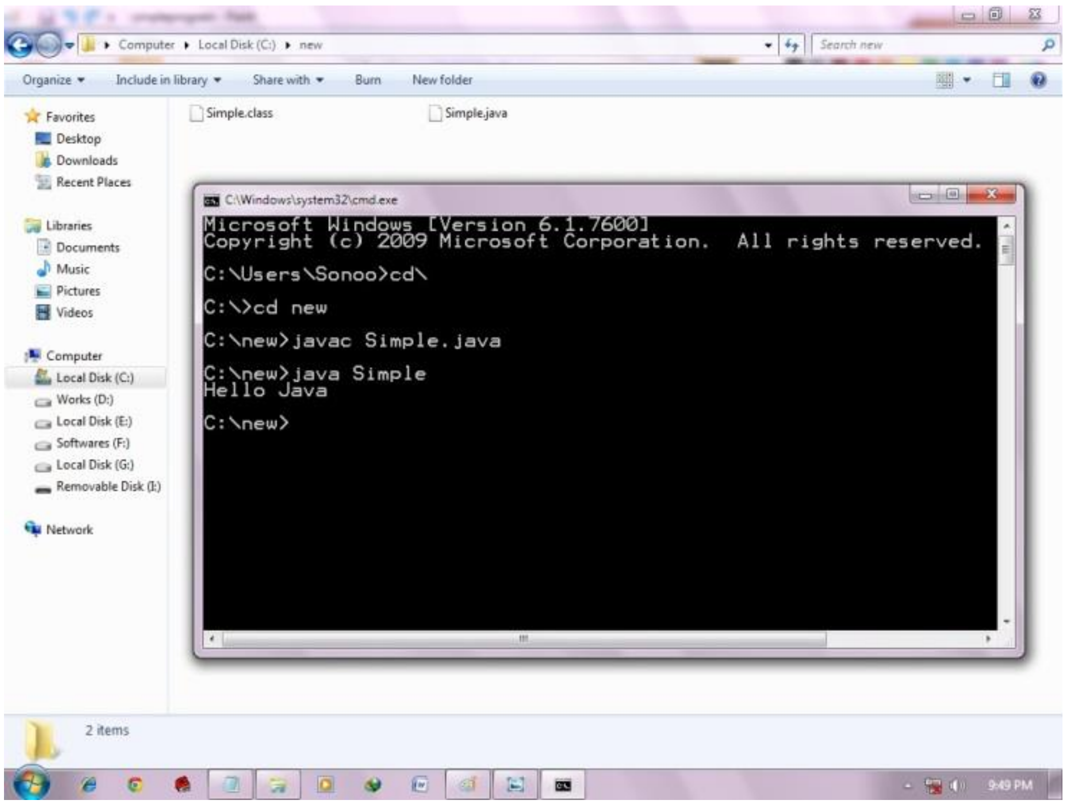


Fig 2.8 Flowchart showing running process

Figure 2.9 shows the compilation and running output of the program on notepad as shown below:



```
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Sonoo>cd\

C:\>cd new

C:\new>javac Simple.java

C:\new>java Simple
Hello Java

C:\new>
```

Fig 2.9 Compilation of Java Program

You need to have following information in order to compile and run program. You need to go to your current directory first; my current directory is c:\new and then writes here:

```
To compile:     javac Simple.java

To execute:     java simple
```

Check your progress 6

1. Which is not associated with first Java program?

a. class	c. static
b. public	d. syntax

2.8 Skills check

There are certain skills involved in Java that can be worked out by considering the Java programs. If you want to develop and run any Java program, in such situation you have to first load JDK in your computer system that can be made available to you from

http://www.oracle.com/technetwork/java/javase/downloads.

After downloading you find that the initial screen will look as shown in fig 2.10. In the figure, the selected red box will allow you to download the kit.

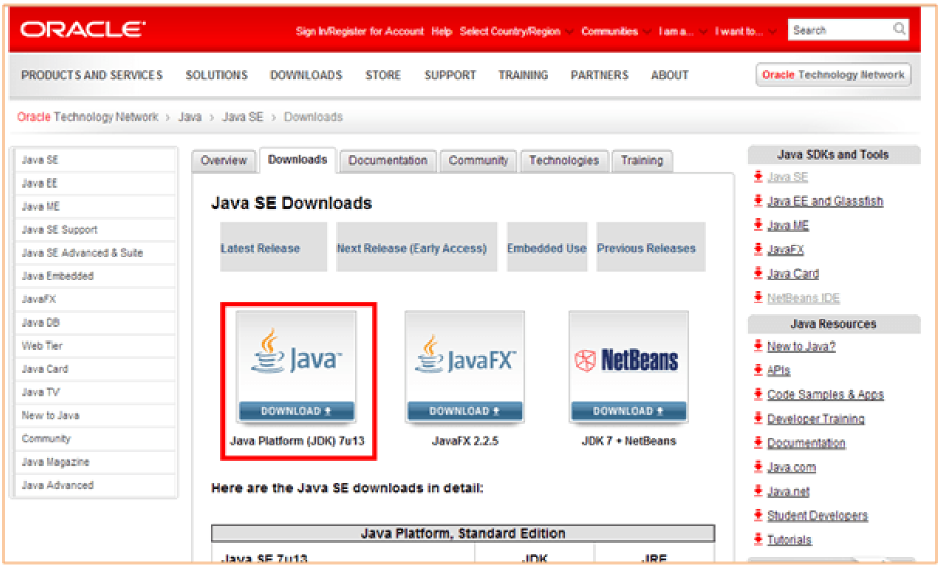


Fig 2.10 Java Downloads

On selecting windows option and clicking on download your download will start that will ask you to save your file.

Check your progress 7

1. To work on Java, you have to have:

a. computer	c. tool kit
b. webcamera	d. development kit

2.9 Exercises

Example 1

Design a program in Java which will calculate the Fibonacci series of first 20 numbers as F(n), where F(n)=F(n–1)+F(n–2) and F(1)=F(2)=1.

Solution:

```
public class Fibonacci {
    public static void main (String args[]) {
        int n = 3;          // the index n for F(n), starting from n=3
        int fn;             // F(n) to be computed
        int fnMmusl = 1;        // F(n-1), hilt to F(2)
        int fnMinus2 = 1;       // F(n-2), hilt to F(1)
        int nMax = 20;          // maximum n, inclusive
        int sum = fnMlnusl fnMinus2;
        double average;
    System.out.println("The first " nMax " Fibonacci numbers are:");
    ……..
    while (n C nMax) {
            // Compute F(n), print it and add to sum
            …….
            //Adjust the index n and shift the numbers
```

```
                    …….
                }
                // Compute and display the average (=sum/nMax)
                ……..
        }
}
```

On executing the above program, we see that first 20 fibonacci numbers are:

```
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765
```

Example 2

Write a Java program which will declare one float variable and other string variable which assigns numbers 5, 11 and shows values on screen.

Solution:

```
public class JavaExercises {
public static void main(String[] args)
{
accessVariables();
}
static void accessVariables0{
int x;
float y;
String s;
x = 5;
y= 11f;
s = "Java programming";
System.out.println(x);
System.out.println(y);
System.out.println(s);
        }
}
```

Check your progress 8

1. Find the result obtained after executing this program.

```
int disp()
        {
         int x,y=10,z=10;
         x=(y==z);
         System.out.println(x);
         return 0;
        }
```

a. 0 c. -1

b. 1 d. 2

2.10 Let Us Sum Up

In this unit it is seen that Java is a programming language which was invented by James Gosling in 1991.

It was found that the first publicly available version of Java was released in 1995 as Java 1.0.

JVM is Java virtual machine that interprets assembled Java binary code known as bytecode for computer's processor which can do Java program instructions.

Java is called as Java 2 Platform having editions as Java 2 Standard Edition (J2SE), Java 2 Enterprise Edition (J2EE) and Java 2 Micro Edition (J2ME). It is noted that Java contains several classes of library that is pre-written.

2.11 Answers for Check Your Progress

Check your progress 1

Answers: (1-d)

Check your progress 2

Answers: (1-a)

Check your progress 3

Answers: (1-c)

Check your progress 4

Answers: (1-a)

Check your progress 5

Answers: (1-a)

Check your progress 6

Answers: (1-d)

Check your progress 7

Answers: (1-d)

Check your progress 8

Answers: (1-b)

2.12 Glossary

1. **Applet -** It is an application which is a type of utility program that will do one or few simple functions.
2. **Version -** It is a sort of description or an account that differs from one point of view to another.
3. **JVM -** It is a program called as Java virtual machine that calculates assembled Java binary codes.
4. **Bytecode -** It is a computer object code which runs through a program known as virtual machine instead by actual computer machine.

5. **Class library -** It is a collection of prewritten classes or codes in shape of templates that can be specified and used by programmer during development of an application program.

2.13 Assignment

Write a program which will print "PASS" if int variable "mark" is more than or equal to 60 else it will print "FAIL".

2.14 Activities

Discuss the output obtained from this program.

```
int a=new int [10];
int I,s=0;
for(i=0;i<10;i++)
{
a[i]=I;
s=s+a[i]+I;
}
System.out.println(s);
```

2.15 Case Study

Discuss about the Java program shown?

```
public class CheckOddEven {        // saved as "CheckOddEven.java"
    public static void main(String0 args) {
    int number = 49;               // set the value of number here!
    System.out.println("The number is " + number);
    if ( …… ){
    System.out.println(……. );
    } else {
    System.out.println( ………… );
```

```
                }
            }
        }
```

2.16 Further Readings

1. Rich Green, Open JDK 2011-11-25.

2. Fitzsimmons, Thomas, 2007 JAVA

3. Angel, Lillian, 2008 OpenJDK

UNIT 3: BEGINNING WITH JAVA PROGRAMMING

Unit Structure

3.0 Learning Objectives

3.1 Introduction

3.2 Java Keywords

3.3 Comments in Java

3.4 Variables and Assignments

3.5 Strings and Characters

3.6 Arithmetic Operators and Expressions

3.7 Type Conversion in Expressions

3.8 Type Conversion

3.9 Skills check

3.10 Exercises

3.11 Let Us Sum Up

3.12 Answers for Check Your Progress

3.13 Glossary

3.14 Assignment

3.15 Activities

3.16 Case Study

3.17 Further Readings

3.0 Learning Objectives

After learning this unit, you will be able to understand:

- Basic of Java Keywords
- Comments in Java
- Study about Variables and Assignments
- Strings and Characters

- Features of Arithmetic Operators and Expressions
- Types of Conversion in Expressions

3.1 Introduction

Java is a strong and simple object oriented programming language which is quiet similar to C++ language. It was invented by Sun Microsystems in year 1991. With the contribution of James Gosling, Patrick Naughton, Chris Warth, Ed Frank and Mike Sheridan, this language was intended to provide a platform-independent programming language.

The Java program is a set of instructions which a computer understand. The Java Compiler will translate programs from Java to computer language which functions as per computer to computer and with different operating systems. While performing, the Byte Code Program will translate Java into bytecode language.

3.2 Java Keywords

Java contains several different types of keywords and reserved words that are:

Abstract Keyword	Boolean Keyword
Break Keyword	Byte Keyword
Case Keyword	Catch Keyword
Char Keyword	Class Keyword
Continue Keyword	Default Keyword
Do Keyword	Double Keyword
Else Keyword	Extends Keyword
False Keyword	Final Keyword
Finally Keyword	Float Keyword
For Keyword	If Keyword
Implements Keyword	Import Keyword
Instance of Keyword	Int Keyword

Interface Keyword	Long Keyword
New Keyword	Null Keyword
Package Keyword	Private Keyword
Protected Keyword	Public Keyword
Static Keyword	Super Keyword
Switch Keyword	Synchronized Keyword
This Keyword	Throw Keyword
Throws Keyword	Transient Keyword
Try Keyword	True Keyword
Void Keyword	While Keyword

Check your progress 1

1. Which of the following keywords are used to define abstract class?

a. abst	c. Abstract
b. abstract	d. abstract class

3.3 Comments in Java

It is seen that comments serves as internal part of any program. With the help of comment, the programmer can read the code which will make them to understand better and can function as required.

Type of Comments

Java consists of three language styles of comments such as:

- Single Line
- Multi-line
- Java doc

It is seen that comments in Java can be inserted anywhere in a program code where a white space is available. These comments are not included by Java compiler in final exe file. We can make as many Java comments as we want in our program as per their usability. In java, the following styles of comments are entertained.

Java single line comments or slash-slash comments or end of line comments (//...)

Java multi-line or traditional comments (/*...*/)

Javadoc or Java documentation comments (/**...*/)

In Java, these single line and multi-line comments are commonly called as:

- documentation comments
- implementation comments

A documentation comment is related to the detail study of mathematics involved in case of class, field or method. It is analysed that a good documentation comments will make use of class and its methods without reading the source code.

An implementation comment will clarify the type of particular piece of code which works. Such type of comment is not commonly applied, but can be used as per need. In Java, the comment will appear on the right before declaration of class, interface or member with every line starting with "*".

Java Single Line Comments

The single line comment in Java begins with two forward slashes (//) without white spaces that will go till the last line. Every line will contain additional two slashes, if the comment exceeds next line. These single line comments are useful for giving short explanations for variables, function declarations and expressions. The single line comment is shown as:

```
If (x c y)
{ // begin if block
x=y;
y = 0;
} // end if block
```

Java Multi-Line Comments

The multi-line comments in Java will show text kept within slash-star (/*) and star-slash (*/). Here while applying such comments, there should be no white space between slash and star. It is required when comment text not fits in single line and needs spanning across lines. The multi-line comment is shown:

```
// CommentDemojava

//Demonstrating multi-line comments

public class CommentDemo

{

        public static void main(String0 args)

        {

                for (hit i = 0; i C 5; /* exits when i reaches to 5 */ i++)

                {

                        System.outplint(i +" ');

                }

        }

}
```

If you perform above program, then the output obtained will be 0 1 2 3 4

In the above program, it is seen that in CommentDemo.java a comment is put inside loop header.

Check your progress 2

1. Which is incorrect in case of Java comments?

   a. Comments should be nested

   b. Comments should have slash star

   c. Comments should be surrounded by double quotes

   d. All of these

3.4 Variables and Assignments

Variables:

In a programming language, a variable is a place to keep the values that are used in a Java program. Before using such values, you need to declare it. Normally, declaring a variable is an initial part in any program. There are four types of variables available in Java programming language such as:

- **Instance Variables:** It is a type of Non-Static Fields where fields are declared without static keyword.
- **Class Variables:** It is a kind of Static Fields which can be declared with the help of static modifier that directs a compiler of its presence.
- **Local Variables:** It is a process where temporary state is stored inside the local variable which is visible to methods in which they are declared.
- **Parameters:** It is a main method which is public static void main(String[] args) where args variable serves as a parameter that are classified as variables not fields.

Declare a Variable

Since Java is a strong programming language, so each variable in this should carry data types associated with it. The benefit of declaring a variable is that the following mentioned primitive data types can be used such as byte, short, int, long, float, double, char and boolean. Declaring a variable in Java requires a data type with variable name such as:

```
int numberOfDays;
```

Here, the variable "numberOfDays" was declared along with data type “int”. In this the line gets ends with a semi-colon which tells the completion of declaration to the Java compiler. As seen, the declared variable, numberOfDays will only carry values which will be compatible with definition of data type.

Initializing Variables

Initializing a variable means assigning value to a variable. If you are using a variable without assigning a value to it, then:

```
int numberOfDays;

// try and add 10 to the value of numberOfDays

numberOfDays = numberOfDays + 10;
```

In this we find that the compiler while execution will give an error as:

```
Variable numberOfDays might not have been initialized
```

Variables can be assigned in Java with the help of an assignment statement which carries similar pattern of equation as found in mathematics such as 2 + 2 = 4. Here we see left side of an equation, right side of an equation and an equal sign ("=") in the middle. In order to assign the value to a variable, we find that left side is the name of variable while right side is the value of the variable as shown:

```
int numberOfDays; numberOfDays = 7;
```

We find that, numberOfDays was declared with data type of int with an initial value as 7. On adding 10 to value of numberOfDays to initialize, we see that:

```
int numberOfDays;
numberOfDays=7;
numberOfDays = numberOfDays + 10;
System.out.println(numberOfDays);
```

Normally, initializing a variable can be done at the same time of its declaration as:

//declare variable and give it a value all in one statement int numberOfDays = 7

Assignments

Normally, there are certain statements where you are adding something to a variable. So it is seen that in Java, the variables are assigned or given values with the help of assignment operators. As seen above, the variables are placed on left-hand side of assignment operator and values are placed on right-hand side of assignment operator. It is seen that an assignment operator will be calculated from right to left, hence x = y = z = 0; will be assigned 0 to z, then z to y then y to x. Consider a variable shown:

```
i = i + 2;
```

Here assign i's value to new value which is i+2. You also write assignments using the += operator. Such operator can be applied for Arithmetic Operators.

Check your progress 3

1. Which of these cannot be used for a variable name in Java?

a. identifier

b. keyword

c. identifier

d. keyword

3.5 Strings and Characters

Strings

In Java, Strings serves as series of characters which are basically objects. Java platform will show a String class in order to form and control strings. Strings are common type of data present in the computer. Java contains certain basics such as chars, +, length () and substring ().

These are series of characters which are collected together like word "Hello". It creates a string in code with the help of chars in double quotes as:

String str = "Hello";

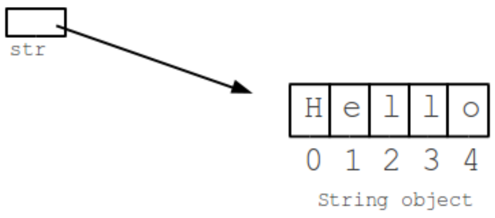


Fig 3.1 String Object

Fig 3.1 shows the string object in memory having individual chars H e l l o representing by index numbers 0, 1, 2........

Creating Strings:

One of the direct ways to form a string is to write as:

String greeting = "Hello world!";

On encounting a string literal in a code, the compiler will form a String object along with its value in 'Hello world!'. Apart from this object, you can create String objects with new keyword and constructor. It is seen that a String class carries 11 constructors which shows the initial value of string with various sources like an array of characters.

```
Public class StringDemo{
        public static void main(String args0){
        char[] helloArray = {'h' , 'e' , 'l' , 'l' , 'o' , '.'};
        String helloString = new String(helloArray);
        System.out.println( helloString );
        }
}
```

It will give an output as:

```
hello
```

String Length:

The procedure used to achieve information about particular object is known as accessory method. The method used with strings is length () method that returns the number of characters present in the string object. After the following two lines of code have been executed, len equals 17:

```
public class StringDemo {
public static void main(String args[]) {
String palindrome = "Dot saw I was Tod";
int len = palindrome.length();
System.out.println("String Length is : " + len );
        }
}
```

This would produce the following result:

```
String Length is : 17
```

Check your progress 4

1. Which among the following operators can be used to join character strings end-to-end in a string object?

   a. + c. &

   b. += d. ||

3.6 Arithmetic Operators and Expressions

Arithmetic Operator

In Java, the basic arithmetic operators are performed by certain operators as shown in table.

Arithmetic Operators

Operator	**Meaning**	**Example**
+	Addition	4 + 5
-	Subtraction	6 - 3
*	Multiplication	3 * 4
/	Division	35 / 7
%	Modulus	34 % 5

In the table shown above, every operator carries two operands on either side of the operator. Here the subtraction operator is applied to negate a single operand, by multiplying operand by -1.

It is carefully assumed in case of division where on storing a division operation in integer, the output will be truncated to next lower whole number as int data type will not be able to carry out floating-point numbers. As an example, the expression 25/7 results in 3 if stored as an integer. In modulus division, % operator will form the remainder of division operation as 25% 7 results in 4 because 25 divided by 7 leaves a remainder of 4.

Consider a simple arithmetic example in Java, where there exists a Source File Weather.java

```
1: class Weather {
2: public static void main(String[] arguments) {
3: float fah = 86;
4: System.out.println(fah +" degrees Fahrenheit is ...");
5: // To convert Fahrenheit into Celsius
6: // Begin by subtracting 32
7: fah = fah - 32;
8: // Divide the answer by 9
 9: fah = fah / 9;
10: // Multiply that answer by 5
11: fah = fah * 5;
12: System.out.println(fah +" degrees Celsius\n");
13:
14: float cel = 33;
15: System.out.println(cel +" degrees Celsius is ...');
16: // To convert Celsius into Fahrenheit
17: //Begin by multiplying it by 9
18: cel = cel * 9;
19: // Divide the answer by 5
20: cel = cel / 5;
21: // Add 32 to the answer
 22: cel = cel + 32;
23: System.out.println(cel +" degrees Fahrenheit");
24:    }
25: }
```

On running such Java program, we will find the output as:

```
86.0 degrees Fahrenheit is…
30.0 degrees Celsius
33.0 Degrees Celsius is…
91.4 degrees Fehrenheit
```

In Lines 3–12 of this application, you will see that temperature in Fahrenheit is converted to Celsius with the use of arithmetic operators as:

Line 3: The floating-point variable fah is created with a value of 86.

Line 4: The current value of fah is displayed.

Line 5: The comments are ignored by Java compiler.

Line 7: The fah is set to its current value minus 32.

Line 9: The fah is set to its current value divided by 9.

Line 11: The fah is set to its current value multiplied by 5.

Line 12: Since fah is converted to Celsius value, fah is shown again.

If we analyse the lines from 14–23, we see that temperatures in Celsius is converted to Fahrenheit with the help of System.out.println() in many statements. Such statement method is used in application to show strings and related information to particular output device. Such method takes a single argument in its parentheses which is a string.

Expressions

Expressions in Java are a type of statement which shows a value. It is made up of variables, constants and method returned values along with operators. It contains certain side-effects in shape of actions that carries out at the time of finding an expression. The common mathematical expressions are shown with source code:

```
int x = 3;

int y = x;

int z = x * y;
```

The above three statements are considered as expressions if they shows values which are given to variables. If we consider the first expression, we see that the literal 3 is assigned to variable x. The second expression will assigns value of variable x to variable y with multiplication operator * for multiplying x and y integers resulting and storing in z integer.

Check your progress 5

1. The decrement operator "–" decreases the value of variable by ____.

a. 1	c. 3
b. 2	d. 4

3.7 Type Conversion in Expressions

It is seen in Java programming, various operation having different types of data can be applied in a single expression. For this consider an example shown:

```
char c;
int i;
float f;
c = ‘1’;
i = 1;
f = 1.1f;
f = c + i – f;
```

As studied, Java make use of certain types of variable that can be in single expression as it contains a particular set of change over rules that can solve type differences.

It is seen that when a variable of type char, byte or short is applied in an expression, then its value will be directly changed to int on calculating particular expression. Such type of conversion rules in java is called as integral promotion which will only be in effect at the time of evaluation of an expression. In this the variable memory size will remain same.

Check your progress 6

1. What will be the output of the following program?

```
int i;
float f;
i = 10;
f = 10.5f;
f = f - i;
```

a. 0.5 c. 1.5

b. 1 d. 10

3.8 Type Conversion

Type conversion is basically changing from given data type to specific data type. In Java, the type conversion is done directly when the type of expression present on right side of an assignment operator is endorsed to type of variable located on left of an assignment operator.

```
// 64 bit long integer
Long myLongInteger;
// 32 bit standard integer
int myInteger;
myLongInteger = myInteger;
```

The type conversion in Java can be:

- Implicit
- Explicit

An implicit conversion is such that in which the value of one type of an expression is changed to value of another type of expression without the use of particular directive from the programmer, while an Explicit conversions is basically carried off with the help of casting where the name of type to which a value is converted be kept in parentheses in front of value.

Check your progress 7

1. Which among the following information is correct in case of automatic type conversion in Java?

   a. Source type is bigger than destination type.

   b. Destination type is bigger than source type.

   c. Destination type is same as source type.

   d. None of above

3.9 Skills check

In order to check your shills in Java you need to practice it. Since Java is a strong and simple object oriented programming language having similar commands as C++ has set of instructions that can be easily understood by computers. You can write programs in Java if you have sufficient knowledge about its operators, strings etc.

In this comments made the programmers to easily read the code that can make them understand better and can function as required. You need to master the variables in Java such as Instance Variables, Class Variables, Local Variables and Parameters that are used to apply in small programs where calculations are involved.

Check your progress 8

1. Which is not a data type?

a. bits	c. short
b. byte	d. int

3.10 Exercises

Example 1:

Design a Java program which will make the user to input two integer values and shows the result as operations of such integer?

To solve this using Java program as designed below:

```
import java.util.Scanner;
public class
{
public static void main(String0 args)
{
caculateValues0;
}
static void caculateValues0{
int a,b;
int resultaxesults,resultm;
float resultd;
Scanner sc=new Scanner(System.in);
System.out.print("Enter a:");
a=sc.nextInt();
System.out.print("Enter b:");
b=sc.nextInt();
resulta=a+b;
results=a-b;
resultm=a*b;
resultd=(float)a/b;
System.outprintln("The result of adding is "+resulta);
System.outprintln("The result of subtracting is "-Fresults);
System.outprintln("The result of multiplying is "-Fresultm);
System.out.println("The result of dividing is "+resultd);
        }
}
```

In the above program, if we:

```
        Enter value a:20
        Enter value b:5
        The result of adding is 25.
        The result of subtracting is 15;
        The result of multiplying is 100.
        The result of dividing is 4.
```

3.11 Let Us Sum Up

In this unit we have learnt that Java is a strong and simple object oriented programming language that is similar to C++.

It is seen that Java program contains a set of instructions that can be easily understood by computer.

Comments in Java will allow the programmer to read the code that will be make them understand better.

As studied, variable is a place where values are kept for use by Java program the includes Instance Variables, Class Variables, Local Variables and Parameters

The benefit of declaring a variable is that the mentioned primitive data types can be used such as byte, short, int, long, float, double, char and boolean.

In Java, Strings serves as series of characters which are basically objects. Java platform will show a String class in order to form and control strings.

Expressions in Java are a type of statement which shows a value. It is made up of variables, constants and method returned values along with operators.

It is seen in Java programming, various operation having different types of data can be applied in a single expression

3.12 Answers for Check Your Progress

Check your progress 1

Answers: (1-b)

Check your progress 2

Answers: (1-a)

Check your progress 3

Answers: (1-b)

Check your progress 4

Answers: (1-a)

Check your progress 5

Answers: (1-a)

Check your progress 6

Answers: (1-a)

Check your progress 7

Answers: (1-b)

Check your progress 8

Answers: (1-a)

3.13 Glossary

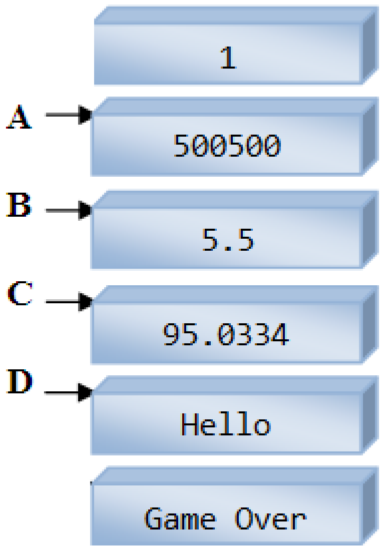
1. **Comments** - These are document containing java codes and program logic.
2. **Statement** - It is an independent unit in a program which is similar as english language.
3. **Block** - It is type of compound statement which is surrounded by curly braces { }.
4. **White Spaces** - These are blank, tab and newline spaces in java which are normally blank.
5. **Variable** - It is the name of storage location which will stores value of definite data type.
6. **Identifier** - It is a series of characters which is used to name a variable.

3.14 Assignment

Compare the types of variable present in Java and discuss.

3.15 Activities

In a variable diagram shown, which alphabet shows the string variable?



3.16 Case Study

In a year having month from 1-12 and days from 1-31, what will be the boolean expression that will return true for dates that appears before September 8, 2015.

3.17 Further Readings

1. Learning Programming by Peter Norvig's

2. Approach programming with a more positive by P.Brian.Mackey

Block Summary

In this block we will get sufficient knowledge about basic of Java and its history. The origination of Java along with its different versions is well explained. The three types of Java 2 Platform such as Java 2 Standard Edition (J2SE), Java 2 Enterprise Edition (J2EE) and Java 2 Micro Edition (J2ME) are illustrated with diagrams. The block detailed with sample exercises on Java program explained with different set of instructions made the students to practice more by varied concepts. The idea about Comments and different variables such as Instance Variables, Class Variables, Local Variables and Parameters are well explained.

After this block study, student can implement basic description about Java and its necessary tools that are used to design first Java program and subsequent Java versions. It is important for readers to read this block as basic of Java program design style are explained with which a student can write and design a program of its own.

Block Assignment

Short Answer Questions

1. What do you mean by programming?
2. What are Java Comments?
3. When was Java invented?
4. What do you mean by Java binary code?
5. What are Java Expressions?

Long Answer Questions

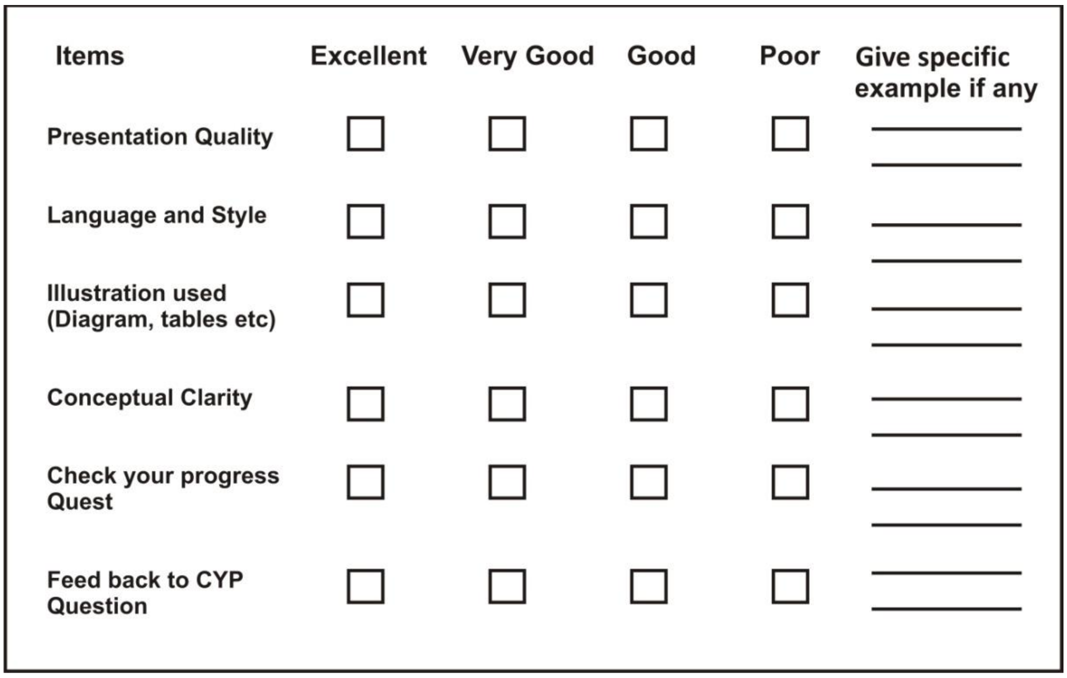
1. What is the benefit of declaring a variable in a Java Program?
2. Explain Flowchart with examples?
3. What is the function of class library in Java?

Enrolment No.

1. How many hours did you need for studying the units?

Unit No	**1**	**2**	**3**	**4**
Nos of Hrs				

2. Please give your reactions to the following items based on your of the block:



Items	Excellent	Very Good	Good	Poor	Give specific example if any
Presentation Quality	☐	☐	☐	☐	
Language and Style	☐	☐	☐	☐	
Illustration used (Diagram, tables etc)	☐	☐	☐	☐	
Conceptual Clarity	☐	☐	☐	☐	
Check your progress Quest	☐	☐	☐	☐	
Feed back to CYP Question	☐	☐	☐	☐	

3. Any Other Comments

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

“Education is something which ought to be brought within the reach of every one.”

- Dr. B. R. Ambedkar

Dr. Babasaheb Ambedkar Open University
Jyotirmay’ Parisar, Opp. Shri Balaji Temple, Sarkhej-Gandhinagar Highway, Chharodi,
Ahmedabad-382 481.

OBJECT ORIENTED CONCEPTS

PGDCA-102

BLOCK 2:

JAVA CONTROL STATEMENTS, OPERATORS AND CLASSES

Dr. Babasaheb Ambedkar Open University
Ahmedabad

OBJECT ORIENTED CONCEPTS

Knowledge Management and
Research Organization
Pune

Editorial Panel

Author
Er. Nishit Mathur

Language Editor
Prof. Jaipal Gaikwad

Graphic and Creative Panel
Ms. K. Jamdal
Ms. Lata Dawange
Ms. Pinaz Driver
Ms. Tejashree Bhosale
Mr. Kiran Shinde
Mr. Akshay Mirajkar

Acknowledgment
Every attempt has been made to trace the copyright holders of material reproduced in this book. Should an infringement have occurred, we apologize for the same and will be pleased to make necessary correction/amendment in future edition of this book.
The content is developed by taking reference of online and print publications that are mentioned in Bibliography. The content developed represents the breadth of research excellence in this multidisciplinary academic field. Some of the information, illustrations and examples are taken "as is" and as available in the references mentioned in Bibliography for academic purpose and better understanding by learner.'

ROLE OF SELF INSTRUCTIONAL MATERIAL IN DISTANCE LEARNING

The need to plan effective instruction is imperative for a successful distance teaching repertoire. This is due to the fact that the instructional designer, the tutor, the author (s) and the student are often separated by distance and may never meet in person. This is an increasingly common scenario in distance education instruction. As much as possible, teaching by distance should stimulate the student's intellectual involvement and contain all the necessary learning instructional activities that are capable of guiding the student through the course objectives. Therefore, the course / self-instructional material are completely equipped with everything that the syllabus prescribes.

To ensure effective instruction, a number of instructional design ideas are used and these help students to acquire knowledge, intellectual skills, motor skills and necessary attitudinal changes. In this respect, students' assessment and course evaluation are incorporated in the text.

The nature of instructional activities used in distance education self-instructional materials depends on the domain of learning that they reinforce in the text, that is, the cognitive, psychomotor and affective. These are further interpreted in the acquisition of knowledge, intellectual skills and motor skills. Students may be encouraged to gain, apply and communicate (orally or in writing) the knowledge acquired. Intellectual-skills objectives may be met by designing instructions that make use of students' prior knowledge and experiences in the discourse as the foundation on which newly acquired knowledge is built.

The provision of exercises in the form of assignments, projects and tutorial feedback is necessary. Instructional activities that teach motor skills need to be graphically demonstrated and the correct practices provided during tutorials. Instructional activities for inculcating change in attitude and behavior should create interest and demonstrate need and benefits gained by adopting the required change. Information on the adoption and procedures for practice of new attitudes may then be introduced.

Teaching and learning at a distance eliminates interactive communication cues, such as pauses, intonation and gestures, associated with the face-to-face method of teaching. This is particularly so with the exclusive use of print media. Instructional activities built into the instructional repertoire provide this missing interaction between the student and the teacher. Therefore, the use of instructional activities to affect better distance teaching is not optional, but mandatory.

Our team of successful writers and authors has tried to reduce this.

Divide and to bring this Self Instructional Material as the best teaching and communication tool. Instructional activities are varied in order to assess the different facets of the domains of learning.

Distance education teaching repertoire involves extensive use of self-instructional materials, be they print or otherwise. These materials are designed to achieve certain pre-determined learning outcomes, namely goals and objectives that are contained in an instructional plan. Since the teaching process is affected over a distance, there is need to ensure that students actively participate in their learning by performing specific tasks that help them to understand the relevant concepts. Therefore, a set of exercises is built into the teaching repertoire in order to link what students and tutors do in the framework of the course outline. These could be in the form of students' assignments, a research project or a science practical exercise. Examples of instructional activities in distance education are too numerous to list. Instructional activities, when used in this context, help to motivate students, guide and measure students' performance (continuous assessment)

PREFACE

We have put in lots of hard work to make this book as user-friendly as possible, but we have not sacrificed quality. Experts were involved in preparing the materials. However, concepts are explained in easy language for you. We have included may tables and examples for easy understanding.

We sincerely hope this book will help you in every way you expect.

All the best for your studies from our team!

OBJECT ORIENTED CONCEPTS

Contents

BLOCK 1: INTRODUCTION TO PROGRAMMING WITH JAVA

UNIT 1 PROBLEM SOLVING WITH COMPUTERS

Need of programming, Algorithms, Flowcharts, Pseudo-Code, Programming Languages Overview, and Overview of Object Oriented Concepts, Skills check, Exercises

UNIT 2 INTRODUCTION TO JAVA

History, Overview of JVM & Bytecode, Java versions (J2me etc.), Java class library, JDK, Your First Java Application, Skills check, Exercises

UNIT 2 BEGINNING WITH JAVA PROGRAMMING

Java Keywords, Comments in Java, Variables and Assignments, Strings and Characters, Arithmetic Operators and Expressions, Type Conversion in Expressions, Type Conversion, Skills check, Exercises

BLOCK 2: JAVA CONTROL STATEMENTS, OPERATORS AND CLASSES

UNIT 1 JAVA CONTROL STATEMENTS

The if statement, The if-else statement, Blocks of Code, The for statement, Increment & Decrement operators, Backslash codes, Relational and Boolean logical operators, Ternary operators, Skills check, Exercises

UNIT 2 MORE ABOUT CONTROL STATEMENTS AND OPERATORS

Nested if statement, Variations of the for loop, The while loop, The do loop, Nested loops, The break statement, The continue statement, The switch statement, The bitwise operators, Skills check, Exercises

UNIT 3 **CREATING CLASSES**

The General form of a class, Creating Simple Classes, Adding Constructors, Constructor Overloading, The this keyword, Instance variables and methods, Static variables and methods, Local variables and its scope, Method overloading, Argument Passing, Wrapper Classes, System Class, Garbage Collection, Skills check, Exercises

---

BLOCK 3: **INHERITANCE, INTERFACE, PACKAGES AND EXCEPTIONS IN JAVA**

UNIT 1 **INHERITANCE**

Overview of Re-usability, Subclasses, Inheritance and Variables, Method Overriding, Inheritance and Methods, Inheritance and Constructors, Class Modifiers, Variable Modifiers, Constructor Modifiers, Method Modifiers, Object and Class classes, Skills Check, Exercises

UNIT 2 **INTERFACES AND PACKAGES**

Interfaces, Interface References, Interface Inheritance, The instanceof Operator, Packages, The import statement, Access control and Packages, Skills Check, Exercises

UNIT 3 **EXCEPTIONS**

Overview of Exceptions, Exception Handling, Catch Block and searches, The throw statement, Exception and Error classes, The throws clause, Custom exceptions, Skills check, Exercises

BLOCK 4: **JAVA CLASS LIBRARY, FILE HANDLING AND GUI**

UNIT 1 **INTRODUCTION TO JAVA CLASS LIBRARY**

Classes of java.util package, Classes of java.net package, Classes of java.lang package, Overview of Collection Framework, Skills check, Exercises

UNIT 2 **FILE HANDLING IN JAVA**

Overview of File Handling, File Class in Java, Using Character Stream Classes, Using Byte Stream Classes, Using Scanner and Console Classes in Java, Skills check, Exercises

UNIT 3 **BUILDING GRAPHICAL USER INTERFACE (SWING)**

Building applications using classes related to Graphical Interface, Creating Layouts in GUI, Using Events in GUI applications, Skills check, Exercises

Dr. Babasaheb
Ambedkar
Open University

PGDCA-102

OBJECT ORIENTED CONCEPTS

BLOCK 2: JAVA CONTROL STATEMENTS, OPERATORS AND CLASSES

UNIT 1
JAVA CONTROL STATEMENTS 03

UNIT 2
MORE ABOUT CONTROL STATEMENTS AND OPERATORS 22

UNIT 3
CREATING CLASSES 43

BLOCK 2: JAVA CONTROL STATEMENTS, OPERATORS AND CLASSES

Block Introduction

In Java, when any statement happens to skip or executed more than once, then it needs a control. The operators are used to carry out primitive data types which can be categorised as unary, binary or ternary.

In this block we will learn and study about concept involved in studying and working of control structures with stress on their syntaxes. The outline of these structures with their programming techniques is well explained. The knowledge about different types of operators with their features are helps the students to grasp more. The detailed about loops and statements with several illustrations are provided which could be of any future use to students.

After completing this block, students will be able to know more about control structures and its features with basic operations about operators and statements. The concept of constructor and garbage collector will allow him to work on certain Java control platforms. After this block, students will get knowledge and confidence about working with loops, operators, structures and statements.

Block Objective

After learning this block, you will be able to understand:

- Control structures
- Study the syntax of IF structure
- Basic of IF-Else statement
- Concept of FOR statement
- Idea about prefix increment/decrement operators
- Detail about newline character
- Idea about ternary operator
- Structure of Nested IF statement

- Idea about variation in FOR loop
- Detail about condition in while loop
- Idea about working of do..while loop
- Knowledge about processing of Nested loop
- Study the functions about Break Statement
- Idea about continue statement
- Basic about working of Switch statement
- The basic of three classes
- Knowledge about role of constructor
- Idea about constructor overloading
- Features of garbage collector

Block Structure

Unit 1: Java Control Statements

Unit 2: More about Control Statements and Operators

Unit 3: Creating Classes

UNIT 1: JAVA CONTROL STATEMENTS

Unit Structure

1.0 Learning Objectives

1.1 Introduction

1.2 The if Statement

1.3 The if-else Statement

1.4 Blocks of Code

1.5 The for Statement

1.6 Increment and Decrement Operators

1.7 Backslash Codes

1.8 Relational and Boolean Logical Operators

1.9 Ternary Operators

1.10 Skills Check

1.11 Exercises

1.12 Let Us Sum Up

1.13 Answers for Check Your Progress

1.14 Glossary

1.15 Assignment

1.16 Activities

1.17 Case Study

1.18 Further Readings

1.0 Learning Objectives

After learning this unit, you will be able to understand:

- Study of if statement
- Study of if-else statement
- Study of Blocks of Code
- Study of Increment & Decrement operators

- Study of Backslash codes

1.1 Introduction

The programs that were written and designed up till now, carries a sequential flow of control. It means that the statements initially were executed line by line from top to bottom in a particular order. It is found that when any statement happens to skip or executed more than once, then it needs a control. In this unit we will see that such can be done using control structures. The control structure carries three groups:

- Decision making statements
- Repetition statements
- Branching statements

Decision making is type of control statements that will decide whether or not a given statement or group of statements be worked out or not. Repetition statements are such where the statements are carried out more than ones. Branching statements on the other hand transfers the control to another line in a given code.

1.2 The if Statement

It is seen that the if statement is applied to particular statement or block of statements to carried out. The if statement carries a condition that will judge whether the statements will worked out or not. It is found that when the condition in parentheses works to be true, then statements gets executed, otherwise, it will be skipped off. Conditions are created using conditional and relational operators. Following is syntax of if statement as:

```
if (condition) {
// code
}
```

There is another code shown which displays the message "Mohit" when the value of boolean variable becomes true.

```
if (wish == true) {
        System.out.println("Nishit");
}
```

In the code, the braces get erased. These are not used in case of block consists of only single statement. Also, the condition 'wish==true' can be described as 'wish'.

Check your progress 1

1. Which among the following acts as a selective statement?

a. if	c. for
b. goto	d. else if

1.3 The if-else Statement

It is studied that if statement is called as single selection structure where only single statement can be worked out. Under such condition, no alternative statement will work when the condition appears to be false. For this, we need to have if else statement which is a double selection structure statement. In this when the condition becomes true, and then if block will worked out, otherwise, else block will work. The syntax of if else structure is shown:

```
If (condition ){
        // code
} else {
        //code
}
```

In an example showing whether the number is even an odd number, it is found that an even number is that which is divisible by 0 and leaves a remainder of 0. The remainder when divides number by 2 can be obtained with the help of modulo (%) mathematical operator. Consider the syntax:

```
If (num % 2 == 0 ){
        System.out.println("Number is even");
} else {
        System.out.println("Number is odd");
}
```

In this, we see that an if else statement is nested where block of if or else carries another if else structure. Here, an else is always linked with previous else. Such type of concepts forms a chained if else structure where only single several alternatives worked. Consider the code shown where if else structures have been nested to obtain number stored in int num:

```
if(num=1){
System.out.println("One");
} else {
if (num = 2) {
System.out.println("Two"); } else {
    if (num = 3) {
        System.out.println("Three"); } else {
    System.out.println("Numbers greater than three cannot be processed
    by this code");
        }
    }
}
```

It is found that the complete block is attached with else even if no braces are used. On removing the braces, we are left with as shown below. We have also written the words if and else on the same line.

```
 if (num = 1) {
 System.out.println("One");
 } else if (num = 2) {
 System.outprintln(Two");
 } else if (num = 3) {
 System.out.println("Tbree");
 } else {
     System.out.println("Numbers greater than three cannot be processed
by this code");
     }
```

The above code shows the chained if else structure which is obtained using a number of if and else statements.

Check your progress 2

1. What will be value obtained, if the program is executed?

```
Public class ControlStatement {
Public static void main (string [] args){
int a = 25;
if (~a>25)
    a++;
    a+=a;
system.out.print(a);
      }
}
```

a. -49 c. 48

b. 50 d. 65

1.4 Blocks of Code

In Java, there exist two or more statements that are grouped in bocks of code which is often called as code blocks that can be seen in enclosed statements among opening and closing curly braces. On forming such block of code, it becomes logical unit which are applied in case of single statement. Consider this if statement:

```
if (x < y){
x = y;
y = 0;
}
```

In the above code we see that if x is < y, then both statements inside block will get executed. So the two statements in block generate a logical unit where statement cannot be worked without others. Consider an example where a block of code as the target of for loop is shown:

```
/* Demonstrate a block of code.
Call this file "Test.java"
*/
class Test {
public static void main (String args[]) {
int x, y;
y=20;
for ( x= 0; x<10; x++) {
System.out.piintln("This is x: " x);
System.outprintln("This is y: " y);
Y = y-2;
            }
      }
}
```

The above program will generate an output as shown:

```
This is x: 0
This is y: 20
This is x: 1
This is y: 18
This is x: 2
This is y: 16
This is x: 3
This is y: 14
This is x: 4
This is y: 12
This is x: 5
This is y: 10
This is x: 6

This is y: 8

This is x: 7

This is y: 6
```

Here, we see that the target of for loop is block of code and not simply a single statement. So every time the loop will repeat three statements present inside block.

Check your progress 3

1. It is seen that a compound statement in Java is enclosed in:

a. parenthesis c. square brackets

b. braces d. all of them

1.5 The for Statement

In Java, it is seen that a for loop runs with group of Java statements till boolean condition results as true. Such loop will combine three elements such as initialization statement, boolean expression and increment or decrement statement. The syntax of for loop is:

```
for (<initialization> ; <condition> ; <statement>) {

        <Block of statements>;

}
```

In this, we find that the initialization statement will get worked out before loop begins. Normally, it is applied to initialize loop variable. Here the condition statement will run prior to block of statements. In such case, the block of statement will work only when boolean condition becomes true. Hence, this is used to increment or decrement loop variable as shown in example.

```
for (int i = 0; i < 5 ; i++)

{

        System.out.println("i is : "+i);

}
```

Here we find that it is possible to initialize various variables in initialization block of for loop with the help of comma shown below:

for (int i = 0, j = 5; i++)

Here we can have more than one increment or decrement portion as shown:

for (int i = 0; i < 5; i++, j++)

The flow chart of working of for statement in Java is shown in fig 1.1

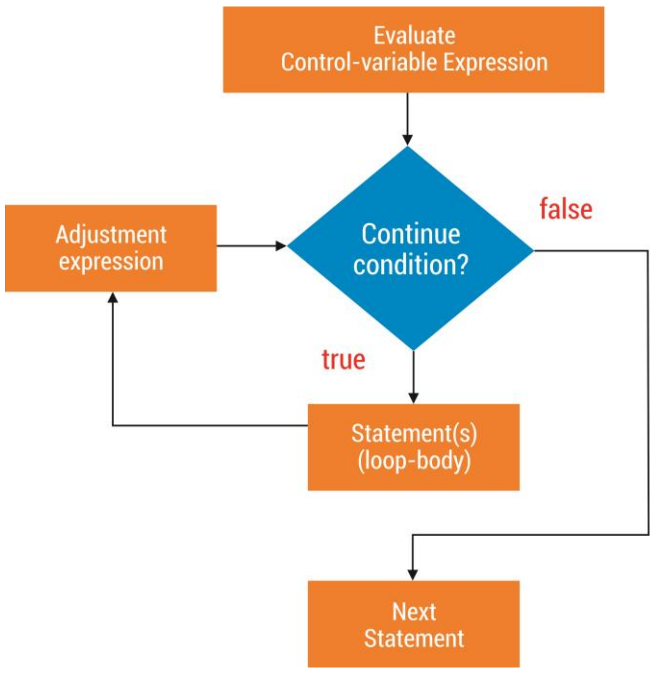


Fig 1.1 Flow chart of for statement

Check your progress 4

1. What will the value obtained from the program?

```
Public class loop {
    Public static void main (String[] args){
        Integer a = 012, b;
        for (b=0;b<a;b++)
        System.out.print(b);
        }
}
```

a. 9 c. 15

b. 11 d. 18

1.6 Increment and Decrement Operators

In Java, we have noticed that there exists two type of Increment or decrement operators as ++ and --. These operators are exceptional as they can be written both prior to operand as applied and is known as prefix increment/decrement or when used after will know as postfix increment/decrement. Consider an example:

```
x = 1;

y = ++x;

System.out.println(y);

prints 2, but

x = 1;

y = x++;

System.out.println(y);

prints 1

Source Code

//Count to ten

class UptoTen {

        public static void main (String args[]) {

        int i;

        for (i=1; i 0; i++)

        System.out.println(i);

                }

        }

}
```

If we write i++, then it means i = i + 1 and when i--, then i = i - 1. Here we will add and subtract one from number which is common operations which could be done by special increment and decrement operators. Also, for add and assign operation, we can use +=. Normally it is seen that we write as i += 11. It means that we want to count from 0 to 15 by two's as:

```
class CountToFifteen {

public static void main (String args[]) {
```

```
int i;
for (i=0; i =15; i += 2) { //Note Increment Operator by 2
System.out.println(i);
          }
     } //main ends here
}
```

Consider an operator -=, if we want to count from ten to zero by twos, then will write this as:

```
class CountToZero {
public static void main (String args[]) {
int I;
for (i=10; i>= 0;i==2) { //Note Decrement Operator by 2
System.out.println(i);
          }
     }
}
```

Check your progress 5

1. What do you mean by variable++?

   a. Adding 1 to variable.

   b. Adding 1 to variable after using its current value.

   c. Adding 1 to variable before using its value.

   d. Double the value of variable.

1.7 Backslash Codes

These are characters which cannot be applied from keyboard and cannot be printed. These can be used anywhere as normal character and are termed as escape sequences. It is seen that a single most important backslash code is \n, which is called as newline character. We see that backslash codes are character constants which can be assigned to character variable inside single quotes. It can be written as:

char = ch;

ch = “\”;

Backslash types

Backslash Code	Description
\t	tab character ('\u0009')
\n	new line or line feed character ('\u000A')
\r	carriage-return character ('\u000D')
\f	form-feed character ('\u000C')
\a	alert or bell character ('\u0007')
\e	escape character ('\u001B')
\cx	control character corresponding to x
\\	backslash character
\0n	character with octal value 0n (0 <= n <= 7)
\0nn	character with octal value 0nn (0 <= n <= 7)
\0mnn	character with octal value 0mnn (0 <= m <= 3, 0 <= n <= 7)
\xhh	character with hexadecimal value 0xhh
\uhhhh	character with hexadecimal value 0xhhhh

Check your progress 6

1. What is \\?

   a. character with hexadecimal value

   b. escape character

   c. backslash character

   d. alert character

1.8 Relational and Boolean Logical Operators

Relational operators

There are many types of relational operators such as <, <=, >, >=, !=, ==. These operators can be used and checked with the if statements as:

```
if (x == 2)
{
        if (y!=2)
        {
        System.out.println("Both conditions are true.");
        }
}
```

In Java, it is easier to handle multiple logic conditions with logic operators like &&, || and !.

It is found that && is logical and && combines two boolean values that returns boolean as true when only both of operands are true as shown:

```
boolean b;
b = 3 > 2 && 5 < 7; // b is true
b = 2 > 3 && 5 < 7; // b is now false
```

|| is logical or. || combines two boolean variables or expressions that returns result as true when either or both of operands are true as shown:

```
boolean b;
b = 3 > 2 || 5 < 7; // b is true
b = 2 > 3 || 5 < 7; // b is still true
b = 2 > 3 || 5 > 7; // now b is false
```

Here we find that the last logic operator is ! which states not. This will reverse the value of boolean expression. Now if b is true, then !b is false and if b is false, then !b is true as shown:

```
boolean b;
```

```
b = !(3 > 2); // b is false

b = !(2 > 3); // b is true
```

Such operators will test for multiple conditions easily as shown in earlier example which can now be written as:

```
 if (x == 2 && y != 2)

 {

   System.out.println("Both conditions are true.");

 }
```

Boolean logical operators

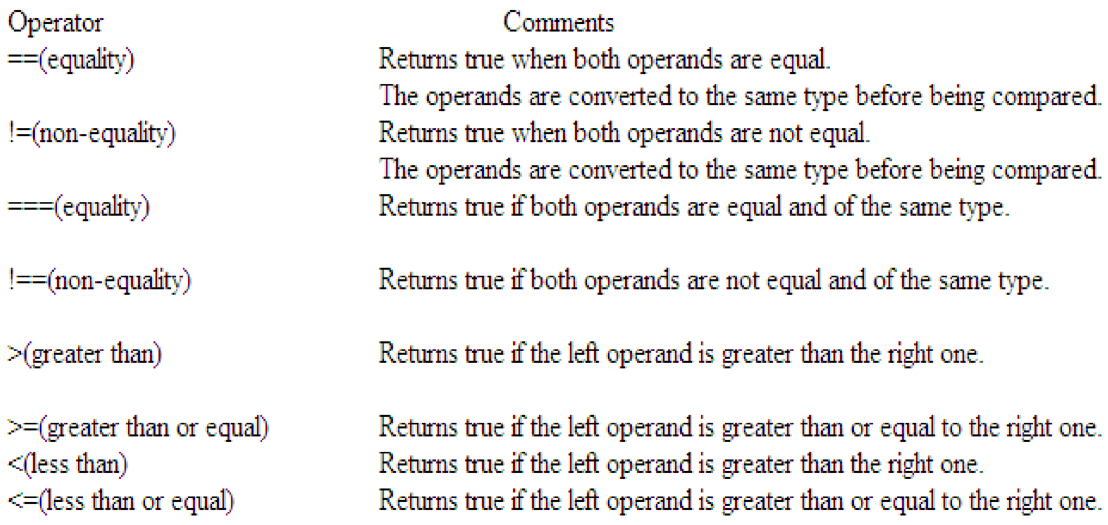
There are also many types of Boolean logical operators such as | , & , ^ , ! , || , && , == , != . These operators will take the value to be true or false, and have default value as false. The main use of Boolean facilities is to work with expressions that controls if decisions and while loops. Such operators act on Boolean operands as per the table shown:

Boolean operands

A	B	A|B	A&B	A^B	!A
false	false	false	false	false	true
true	false	true	false	true	false
false	true	true	false	true	true
true	true	true	true	false	false

| the OR operator
& the AND operator
^ the XOR operator
! the NOT operator
|| the short-circuit OR operator
&& the short-circuit AND operator
== the EQUAL TO operator
!= the NOT EQUAL TO operator

Boolean operators



Operator	Comments
==(equality)	Returns true when both operands are equal. The operands are converted to the same type before being compared.
!=(non-equality)	Returns true when both operands are not equal. The operands are converted to the same type before being compared.
===(equality)	Returns true if both operands are equal and of the same type.
!==(non-equality)	Returns true if both operands are not equal and of the same type.
>(greater than)	Returns true if the left operand is greater than the right one.
>=(greater than or equal)	Returns true if the left operand is greater than or equal to the right one.
<(less than)	Returns true if the left operand is greater than the right one.
<=(less than or equal)	Returns true if the left operand is greater than or equal to the right one.

Check your progress 7

1. A relational operation returns ____ value.

   a. int c. float

   b. char d. boolean

2. Which among the following returned by greater than, <, and equal to, ==, operator?

   a. Integers c. Boolean

   b. Floating point numbers d. None of these

1.9 Ternary operators

In Java, the ternary operator also shown as ?, it is a type of if else statement which can be used to find an expression and return one of two operands as per the result of an expression.

```
boolean b = true;

String s = ( b == true ? "True" : "False" );
```

In this we see that the value of set of String s is set as per the value of boolean b, which can be represented by if else statement as:

```
boolean b = true;
String s;
if(b == true)
{
    s = "True";
}else{
    s = "False";
}
```

Such type of operator is useful in lowering number of lines of code. It can also be useful to test null values.

Check your progress 8

1. What will be highest value among three integers as per ternary operations?

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int 12, 6, 15, big ;
    clrscr() ;
    printf("Enter three numbers : ") ;
    scanf("%d %d %d", &a, &b, &c) ;
    big = a > b ? (a > c ? a : c) : (b > c ? b : c) ;
    printf("\nThe biggest number is : %d", big) ;
    getch() ;
}
```

a. 12　　c. 15

b. 6　　d. none

1.10 Skills Check

In Java, there are certain skills involved while analysing control structures. There are three such types of structures as Decision making, Repetition and Branching. The basic of control statement is if statement which is the single selection structure having one statement. You need to practice certain examples based on if statement in order to go for else if and nested if. There are two or more statements that are grouped in bocks of code which is often called as code blocks. Skills are involved in evaluating certain ternary operations which will clear the logic involve.

Check your progress 9

1. Which is not a part of control structure?

   a. Decision making

   b. Repetition

   c. Branching

   d. Boolean

1.11 Exercises

Example 1:

Consider a program which will show use of newline character.

```
class NewLineCharDemo
{
    public static void main (String args[])
    {
        System.out.print(“This is first line. \n”);
        System.out.print(“This is second line.”);
        System.out.print(“This is third line.”);
    }
}
```

Output

```
This is first line.
This is second line. This is third line.
```

1.12 Let Us Sum Up

In this unit we have learnt that, the control structure in Java carries three group statements such as Decision making statements, Repetition statements and Branching statements.

It is studied that if statement is called as single selection structure where only single statement can be worked out.

In Java, there exist two or more statements that are grouped in bocks of code which is often called as code blocks which is seen in enclosed statements among opening and closing curly braces.

In Java, the ternary operator also shown in shape of Question mark (?) is a type of if else statement which can be used to find an expression and return one of two operands as per the result of an expression

1.13 Answers for Check Your Progress

Check your progress 1

Answers: (1-a)

Check your progress 2

Answers: (1-b)

Check your progress 3

Answers: (1-b)

Check your progress 4

Answers: (1-b)

Check your progress 5

Answers: (1-b)

Check your progress 6

Answers: (1-c)

Check your progress 7

Answers: (1-d), (2-c)

Check your progress 8

Answers: (1-c)

Check your progress 9

Answers: (1-d)

1.14 Glossary

1. **Modulus operator -** It is an operator which is shown with percent sign (%) that works on integers and gives a remainder when a number is divided by another number.
2. **Boolean expression -** It is an expression which shows the result as true or false.
3. **Comparison operators -** These are operators such as ==, !=, >, <, >=, and <= which compares two values.
4. **Logical operators -** These are type of operators like and, or, and not which combines with boolean expressions.
5. **Block -** It is a group of consecutive statements having similar indentation.

1.15 Assignment

Discuss the different types of Control structure statements in Java?

1.16 Activities

Calculate the output of the following program?

```
for(i=1,j=0;i<10;i++)
        j+=I;
System.out.println(i);
```

1.17 Case Study

Discuss various control structures in Java?

1.18 Further Readings

1. Learning Programming by Peter Norvig's
2. Approach programming with a more positive by P.Brian.Mackey

UNIT 2: MORE ABOUT CONTROL STATEMENTS AND OPERATORS

Unit Structure

2.0 Learning Objectives

2.1 Introduction

2.2 Nested if Statement

2.3 Variations of for Loop

2.4 The While Loop

2.5 The Do Loop

2.6 Nested Loops

2.7 The Break Statement

2.8 The Continue Statement

2.9 The Switch Statement

2.10 The Bitwise Operators

2.11 Skills Check

2.12 Exercises

2.13 Let Us Sum Up

2.14 Answers for Check Your Progress

2.15 Glossary

2.16 Assignment

2.17 Activities

2.18 Case Study

2.19 Further Readings

2.0 Learning Objectives

After learning this unit, you will be able to understand:

- Nested IF statement
- While do loops

- Switch and break statements
- Bitwise operators

2.1 Introduction

In Java operators are used to carry out primitive data types which can be categorised as unary, binary or ternary. These operators take one, two or three arguments respectively. It is found that a unary operator exists before or after an argument. As seen a binary or ternary operator will exist in between the arguments. Apart from these there are several divisions of such operators:

- assignment
- arithmetic
- relational
- logical
- bitwise
- compound assignment
- conditional
- type

2.2 Nested if Statement

When an if statement is having a target of another if or else statement, then it is said to be nested inside the outer if. This statement will put one IF Statement inside another.

The syntax of Nested IF statement is shown:

```
if (expression) statement;
if (expression) statement;
        |
        |
        |
        if (expression) statement;
```

Consider another form of an IF statement:

```
if (expression)  statement;
else
        if (expression)  statement;
else
        if (expression)
|
|
|
else statement;
```

To know more about how this statement works in Java consider an example shown. Write a program using if statement which will check number as zero, positive or negative.

```
class CheckSignNumberDemo
{
public static void main(String args[])
        {
        int x = 10;
        if(x > -1)
        if(x != 0)
        if(x > 0)
        System.out.println("x is a positive number having value " + x);
        }
}
```

Output

```
        x is a positive number having value 10
```

Check your progress 1

1. While executing a program having nested loops, which loop will be executed maximum number of times?

   a. outermost loop   c. all loops

   b. innermost loop   d. cannot determined

2.3 Variations of for Loop

In Java, it is seen that for is one of the most versatile statement as it allows a broad range of variations. Consider an example:

```
public class for loop
{
        public static void main (String [] args)
        {
                for (int i=5;i<10;i++)
                {
                        System.out.println("i="+i);
                }
        }
}
```

If we run the above program, we will see that it will generate an output as:

```
i=5
i=6
i=7
i=8
i=9
```

In this program, we see that an initialization statement, condition statement and iteration statement are not compulsory and can be kept empty. If we include a variation, then the above program can be written as:

```
Public class for loop
```

```
{
public static void main (String [] args)
{
         int i=5;
         for (;i<10;)
         {
                  System.out.println(“i=”+i);
                  i++;
         }
         System.out.println(“After for loop”);
         }
}
```

It is seen in the above program, that initialization and iteration part remains empty. If we write an initialization statement before for loop, then increment the counter inside body of loop.

Check your progress 2

1. What will be value of i in the program on execution?

```
public class Main {
  public static void main(String args[]) {
    int i = 0;
    boolean done = false;
    for (; !done;) {
      System.out.println("i is " + i);
      if (i == 10)
        done = true;
      i++;
    }
  }
}
```

a. 0　　　c. 2

b. 1　　　d. all of these

2.4 The While Loop

In Java, while loop is applied to carry out statement(s) till condition remains to be true. First we see about the syntax of loop:

```
while (condition(s))
{
// Body of loop
}
```

In the above we see that if condition remains true then body of loop will carry out soon after execution of loop body condition which is checked again. When the condition is true then body of loop gets carried out again and process will continue till the condition becomes false. In looping, condition should always worked out first which can be true or false, and when it is constant as while (c) {…} where c is constant, then any non-zero value of c will be true and zero remains false.

Also, through testing of multiple conditions like:

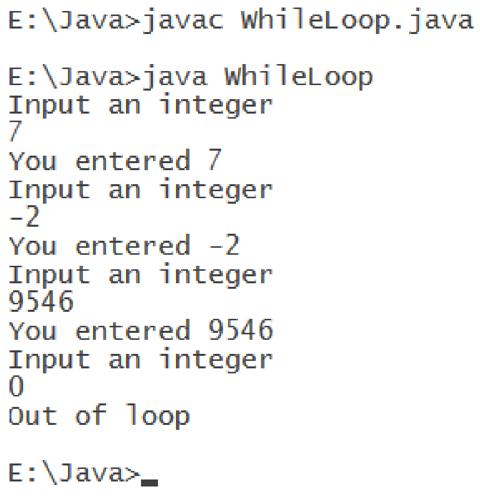
```
While (a > b && c !=0)
{
// Loop body
}
```

The loop body gets worked out till value of a is more than value of b where c is not equal to zero. It is seen that body of loop carries more than single statement. In case of multiple statements, it is required to put them in block having {} and when the body of loop has only single statement, then you can optionally use such bracket {}. To know more, consider an example where a program asks the user to input an integer and show it until user enters 0.

```
import java.util.Scanner;
class WhileLoop {
public static void main(String[] args) {
int n;
Scanner input = new Scanner(System.in);
System.out.println("Input an integer");
while ((n = input.nextInt()) != 0) {
```

```
System.out.println("You entered " + n);
System.out.println("Input an integer");
}
Svstem.out.println("Out of loop");
}
}
```

You will find that the output of such program will show:



```
E:\Java>javac WhileLoop.java

E:\Java>java WhileLoop
Input an integer
7
You entered 7
Input an integer
-2
You entered -2
Input an integer
9546
You entered 9546
Input an integer
0
Out of loop

E:\Java>_
```

Check your progress 3

1. In Java, the while loop is applied till condition remains:

a. false

b. true

c. imaginary

d. all of these

2.5 The Do Loop

A do...while loop is similar to a while loop, except that a do...while loop is guaranteed to execute at least one time.

Syntax:

The syntax of a do...while loop is:

```
do
{
  //Statements
}while(Boolean_expression);
```

Notice that the Boolean expression appears at the end of the loop, so the statements in the loop execute once before the Boolean is tested.

If the Boolean expression is true, the flow of control jumps back up to do, and the statements in the loop execute again. This process repeats until the Boolean expression is false.

Example:

```
public class Test {
  public static void main(String args[]){
    int x = 10;
    do{
      System.out.print("value of x : " + x );
      x++;
      System.out.print("\n");
    }while( x < 20 );
  }
}
```

This would produce the following result:

```
value of x : 10
value of x : 11
value of x : 12
value of x : 13
value of x : 14
value of x : 15
value of x : 16
value of x : 17
value of x : 18
value of x : 19
```

Check your progress 4

1. In Java, do...while loop will execute:

a. at least one time	c. repeated twice
b. every time	d. none of these

2.6 Nested loops

In Java, Nested loops are very helpful. The nested loops can be better understandable by considering example of pattern problems. In this, the integer value is taken as input from user with the result that following pattern gets printed based on input. Now, if we have input as 7, then we see that following 7 star pattern gets printed as:

```
*
**
***
****
*****
******
```

From this pattern we see that loops are used to print such pattern as we are not aware in advance of number of *'s that to be printed. For doing this we require nested loops which is loop within another loop. Here the outer FOR loop keeps track of the line number while the inner FOR loop is used to keep track of the number of stars we are printing. It is analysed that number of stars on a particular line is equal to line number which are related to outer loop control variable. Keeping these things in mind, we can write the following code which prints this pattern. The number n is taken as an input from the user:

```
for (int i = 1; i <= n; i++) {
        for (int j = 1; j<=1; j++) {
                System.out.print("*");
                }
System.out.println();
}
```

It is seen that while using print() method and not println() to show star patterns similarly like *'s as displayed on same line. Once the inner loop is executed, we can move to next line with the help of println() statement.

Check your progress 5

1. In Nested loop, the inner loop is:

a. while loop	c. do..while loop
b. for loop	d. none of these

2.7 The Break Statement

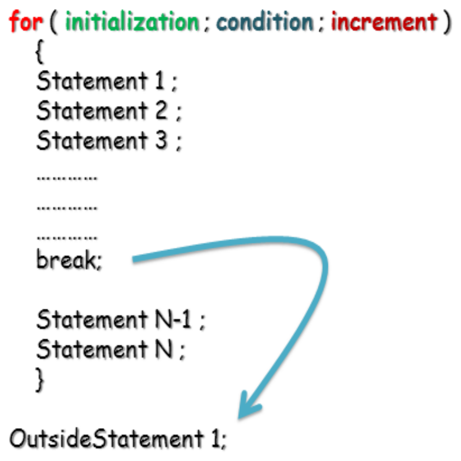
In Java, the Break Statement applies to bring about loop control statements. Such type statement will break for Loop, while Loop and do-while loop. This statement will skip the remaining statements and will execute that statement which is after loop.

The Break statement is used when condition is satisfied inside loop. It is applied to make looping statements more flexible and powerful. The syntax of the Break Statement is break;

Different Ways of Using Break Statement in Java:

Break Statement used in For Loop

The break statement will take control out of the loop where all statements gets carried out. As shown, statement 1, statement 2…. will go on till it reach to break statement where loop will break and takes the control outside the loop.



```
for ( initialization ; condition ; increment )
  {
  Statement 1 ;
  Statement 2 ;
  Statement 3 ;
  ...........
  ...........
  ...........
  break;

  Statement N-1 ;
  Statement N ;
  }
OutsideStatement 1;
```

Fig 2.1 Break statement used in For loop

It is seen that in above arrangement of break statement, it is found that the control will take place outside the loop.

Break Statement used in While Loop

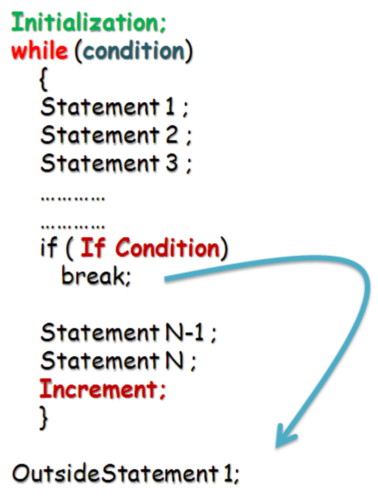


Fig 2.2 Break Statement used in While loop

Here we see that the break statement is present inside the while loop. When such statement will work out and break, then it will move the control outside the while loop.

Check your progress 6

1. The Break Statement in Java will break:

a. For loop c. Do while loop

b. While loop d. all of these

2.8 The Continue Statement

In Java, it is seen that the Continue Statement will skip the part of loop. Similar like break statement, continue statement will not terminate the loop, but will skip the left over part of loop where the control goes back again to check the conditions. The syntax of Continue Statement is:

```
{
//loop body
…………
```

…………

…………

…………

continue;

…………

…………

}

It is studied that Continue Statement is also called as Jumping Statement which skips the Loop and Re-Executes Loop by considering new condition. This statement applies only in Loop Control Statements like For Loop, While Loop and do-While Loop. We can use such statements in many ways:

Continue Statement Written Inside For Loop

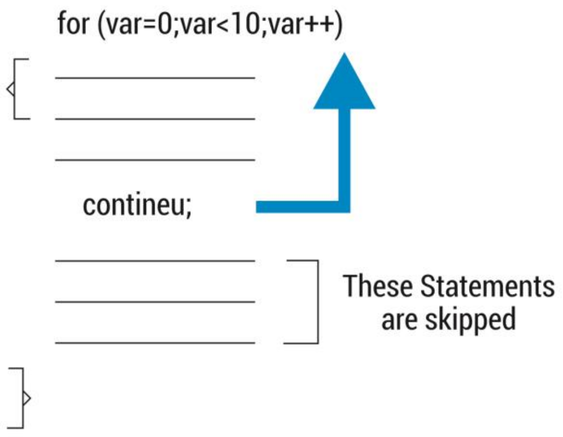


Fig 2.3 Continue statement written inside for loop

Continue Statement Written Inside While Loop

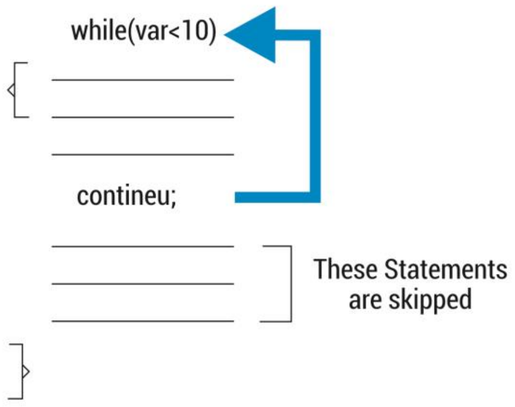


Fig 2.4 Continue Statement written inside while loop

Check your progress 7

1. Which is not a Loop Control Statement?

a. Nested loop	c. While Loop
b. For Loop	d. Do-While Loop

2.9 The Switch Statement

Switch statement is used in Java where you require selecting one of several alternatives that is based on value of an integer, character, or string variable. The syntax of switch statement is:

```
switch (expression)
{
case constant;
        statements;
        break;
[ case constant – 2;
        statement;
        break; ] …
[ default;
        Statements;
        Break; ] …
}
```

It is found that in a switch statement, the expression must evaluate to int, short, byte, or char. In this, each grouping of code lines begins with case keyword and ends with break statement. Here, coding of many case groups can be done where group starts with word case followed by constant and colon.

In such case, coding of more than one statement is possible when the value of switch expression equals to the value of constant. The last line of each case group is a break statement, which causes the entire switch statement to end. Consider an example:

```
Double commissionRate;
```

```
Switch (salesClass)
{
        Case 1:
                commissionRate  = 0.02;
                break;
        Case 2:
                commissionRate  = 0.035;
                break;
        Case 3:
                commissionRate  = 0.05;
                break;
        default:
                commissionRate  = 0.0;
                break;
}
```

It is found that a switch statement can find char data. In example shown, a char variable named sales Category is calculated for commission rates having sales categories as A, B or C.

```
double commissionRate;
switch (salesCategory)
{ case 'A': case 'a:
        commissionRate  = 0.02;
        break;
case 'B':
case 'b':
        commissionRate  = 0.035;
        break;
case 'C':
case c:
        commissionRate  = 0.05;
        break;
default:
        commissionRate  = 0.0;
        break;
        }
```

Check your progress 8

1. In switch statement, an expression evaluates to:

a. int	c. byte
b. short	d. all

2.10 The Bitwise Operators

In Java, it is seen that bitwise operators works on individual bits of integer that can be int value or long value. It is seen that when an operand is shorter than int, it is promoted to int before performing operations.

These operators allow us to know how integers are shown in terms of binary. In case of decimal number 3, it can be shown as 11 in binary, while in case of 5, it can be shown as 101 in binary. It is found that negative integers are stored in two's complement form.

Operator	**Name**	**Example**	**Result**	**Description**
a. & b	and	3 & 5	1	1 if both bits are 1.
a | b	or	3|5	7	1 if either bit is 1.
a ^ b	xor	3 ^ 5	6	1 if both bits are different.
~a	not	~3	-4	Inverts the bits.
n<<p	left shift	3<<<2	12	Shifts bits of n left p positions.
n>>p	right shift	5>> 2	1	Shifts bits of n right p positions.
n>>>p	right shift	-4 >>> 28	15	Shifts bits of n right p

Check your progress 9

1. Which among the following is not a bitwise operator?

a. &	c. |=
b. &=	d. <=

2.11 Skills Check

In Java skills are applied to work and solve problems related to looping. There are certain loop statements such as FOR loop, While loop, Do..While loop where knowledge is used to judge which loop will be executed first.

In programming, looping is common which will make us to reach on certain results. There is several loop statement which continuously works till reaches to particular solution.

Check your progress 10

1. In programming, looping is common which will make us to reach on certain results.

   a. True

   b. False

2.12 Exercises

Example 1:

Program to display triangle of * using nested for loop

```
class NestedForLoopDemo
{
  public static void main(String args[])
  {
for(int i = 1; i <=5 ; i++)
{
for(int j = 1; j <= i; j++)
{
System.out.println("*  ");
}
System.out.println("");
}
```

```
  }
}
```

Output

```
*
* *
* * *
* * * *
* * * * *
```

Example 2 :

Program to print tables

```
class NestedWhileLoop
{
  public static void main(String args[])
  {
int i=1, j=1;
System.out.println("Tables");

while(i <= 2) // change to 2 to 10 or 20 as many tables user want
{
while(j <= 10)
{
System.out.println(i + " * " + j + " = " + (i*j));
j++;
}
i++;
System.out.println("");
System.out.println("");
}
```

```
      }
    }
```

Output

```
Tables
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
1 * 4 = 4
1 * 5 = 5
1 * 6 = 6
1 * 7 = 7
1 * 8 = 8
1 * 9 = 9
1 * 10 = 10

2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20
```

2.13 Let Us Sum Up

In this unit we have learnt that Java operators are used to carry out primitive data types that can be categorised as unary, binary or ternary.

It is seen that in case of looping, condition becomes worked out first which can be true or false, and when it is constant then any non-zero value of c will be true and zero remains false.

If the Boolean expression is true, the flow of control jumps back up to do, and the statements in the loop execute again.

In Java, the Break Statement applies to bring about loop control statements. Such type statement will break for Loop, while Loop and do-while loop.

Switch statement is used in Java where you require selecting one of several alternatives that is based on value of an integer, character, or string variable.

2.14 Answers for Check Your Progress

Check your progress 1

Answers: (1-b)

Check your progress 2

Answers: (1-d)

Check your progress 3

Answers: (1-b)

Check your progress 4

Answers: (1-a)

Check your progress 5

Answers: (1-b)

Check your progress 6

Answers: (1-d)

Check your progress 7

Answers: (1-a)

Check your progress 8

Answers: (1-d)

Check your progress 9

Answers: (1-d)

Check your progress 10

Answers: (1-a)

2.15 Glossary

1. **Looping -** it is a length of line which goes again and again over making an opening
2. **Programming -** It is a technique of writing codes in order to work on computer.

2.16 Assignment

Compute the output of the program?

```
public class Test
{
        public static void main(String args[])
        {
                int = 1, j = 0;
                switch(i)
                {
                        case 2: j += 6;
                        case 4: j += 1;
                        default: j += 2;
                        case 0: j += 4;
                }
                System.out.printlnej = " + );
        }
}
```

2.17 Activities

Compare FOR and While..do loop according to their syntax.

2.18 Case Study

Design a nested if-else statement which will show character grade to percentage of mark as:

Grade A	70 or above
Grade B	60-69
Grade C	50-59
Grade D	40-49
Grade E	30-39
Grade F	less than 30

2.19 Further Readings

1. Learning Programming by Peter Norvig's
2. Approach programming with a more positive by P.Brian.Mackey

UNIT 3: CREATING CLASSES

Unit Structure

3.0 Learning Objectives

3.1 Introduction

3.2 The General form of a Class

3.3 Creating Simple Classes

3.4 Adding Constructors

3.5 Constructor Overloading

3.6 The this Keyword

3.7 Instance Variables and Methods

3.8 Static Variables and Methods

3.9 Local Variables and its Scope

3.10 Method Overloading

3.11 Argument Passing

3.12 Wrapper Classes

3.13 System Class

3.14 Garbage Collection

3.15 Skills Check

3.16 Exercises

3.17 Let Us Sum Up

3.18 Answers for Check Your Progress

3.19 Glossary

3.20 Assignment

3.21 Activities

3.22 Case Study

3.23 Further Readings

3.0 Learning Objectives

After learning this unit, you will be able to understand:

- Adding Constructors
- Constructor Overloading
- This keyword
- Instance variables methods
- Static variables methods
- Local variables

3.1 Introduction

After having basic knowledge about Java, you can now have basics of Java programming experience which you can apply to write your own classes. In this unit, you will receive detailed information about how you can write and define classes which could be in terms of declaring member variables, methods as well as constructors. Also this unit will make you learn to how to apply your classes to design and develop objects along with using certain objects what you can develop.

3.2 General form of a Class

In Java, classes are created using keyword class in which you can define objects. A class contains three types of items:

- Variables
- Methods
- Constructors

It is found that variables describe its state, whereas class carries static and instance variables. In this, method shows the logic that describes the behaviour of a class. As seen, class contains static and instance methods. Where it is seen that a constructor will initialize the state of new instance of class. We see that the general form of a Class in Java can be:

```
class clsName
{
```

```
// instance variable declaration
typel varNamel = valuel;
type2 varName2 = value2;
:
:
typeN varNameN = valueN;
class clsName
{
// instance variable declaration
typel varNamel = valuel;
type2 varName2 = value2;
:
:
typeN varNameN = valueN;
// Constructors
clsName (cparam 1)
{
// body of constructor
}
:
:
clsName(cparamN)
{
// body of constuctor
}
// Methods
rType1 methodName1(mParams1)
{
// body of method
}
}
```

In the above syntax:

- Class shows declared class named clsName which follows java naming conventions for identifiers.
- Instance variables named varNamel by varNameN as normal variable declaration syntax.
- Constructors carry similar name as class having no return values.
- Methods named mthNamel declares mthNameN having return type methods as rtypel by rTypeN and mParamN.

Check your progress 1

1. In the example, the Stock is called as:

```
Class Stock {
Public commodity;
Public price;
Public void buy (int no_of commodity) {}
Public boolean sale () {}
}
```

a. Class c. Methods

b. Fields d. None of above

3.3 Creating Simple Classes

We can write simple class in Java. Consider an example of simple class Java program which will calculate Area of Rectangle as shown:

```
Class Rectangle {

double length;

double breadth;

}

// This class declares an object of type Rectangle.
```

```
Class RectangleDemo {
Public static void main (String args[]) {
Rectangle myrect = new Rectangle();
double area;
// assign values to myrect’s instance variables
        myrect.length = 20;
        myrect.breadth = 10;
// Compute Area of Rectangle
Area = myrect.length * myrect.breadth;
System.out.println(“Area is ”+area);
        }
}
```

If we run this program we will see that the output will be:

```
Area is 200.0
```

While designing such program, we have followed following steps:

Step 1: Class Creation / Declaration:

Consider following class –

```
Class Rectangle {
double length;
double breadth;
}
```

- Class shows new type of data.
- Here rectangle is new data type.
- Rectangle is used to declare objects of type Rectangle.
- Class declaration creates template not actual object.

Step 2: Creation of Object

```
Rectangle myrect = new Rectangle();
```

Actually, the memory is allotted to object as soon as execution of statement that will create instance of class “Rectangle” having instance name as “myrect“. In fig 3.1, we see the use of Constructor

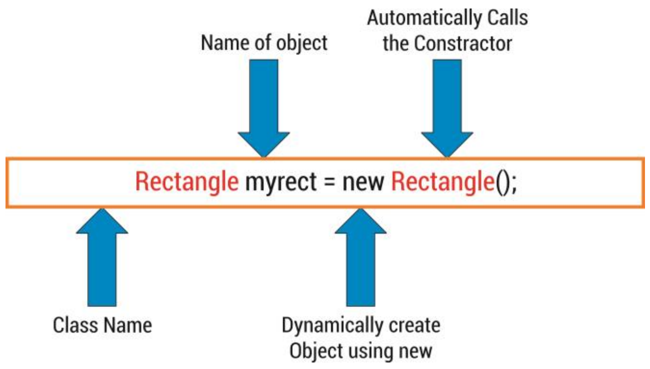

Name of object
Automatically Calls the Constructor
Rectangle myrect = new Rectangle();
Class Name
Dynamically create Object using new

Fig 3.1 Creating Objects

Step 3: Accessing Instance Variables of Object/Instance with DOT Operator

```
myrect.length = 20;
myrect.breadth = 10;
```

It is found that each Instance/Object have its own copy of instance variables that can be length and breadth.

Check your progress 2

1. While creating a simple class program, you have to consider:

a. Class creating	c. Instance variable accessing
b. Object creation	d. all of above

3.4 Adding Constructors

In Java, while discussing about classes, you need to involve with constructor which is present in a class. If you fail to write a constructor for class, then Java compiler will automatically create a default constructor for such class. It is seen that each time a new object is created. Constructor carries similar name as class, so a class can have more than one constructor. It is studied that constructors are methods in Java that are initialises objects. It has similar name as class they called when an object of class is created. Consider simple Java constructor class:

```
Class Programming {
        // constructor method
        Programming() {
                System.out.println("Constructor method called.");
        }
        Public static void main (String[] args) {
                Programming object = new Programming(); // creating object
                }
        }
```

On execution of this program we will see its output as:

```
E:\ Java>javac Programming.java
E:\ Java > java Programming
Constructor method called.
E:\ Java>
```

A simple example of a constructor is shown below:

```
        Public class Puppy{
        Public Puppy() {
        }
Public Puppy (String name) {
//This constructor has one parameter, name.
        }
}
```

Check your progress 3

1. Identify the return type of Constructor from the following.

   a. int

   b. float

   c. void

   d. None of above

3.5 Constructor Overloading

Constructor overloading is a mechanism in Java where a class can have n number of constructors having different parameter lists. To distinguish, it is the duty of compiler to differentiate the constructors by considering number of parameters in list and its type. Similarly like other methods, java constructor also overloads as it develops several constructors in desired class. Consider a constructor overloading example:

```
Class Language {
String name;
Language() {
        System.out.println("constructor method called.");
}
Language (String t) {
        Name = t;
}
Public static void main (String[] args) {
Language cpp = new Language ();
Language java = new Language ("Java");
Cpp.setName("C++");
Java.getName();
Cpp.getName();
}
        Void setName(String t) {
        name = t;
}
        Void getName() {
        System.out.println("Language name: " +name);
}
}
```

The output of this program will be:

```
E: \Java>javac Language.java
E: \ Java > java Language
Constuctor method called.
Language name: Java
Language name: C++
E: \Java
Language name:
```

Consider another overloading program where you will find an overloading of constructor that generates an output.

```
class Student5{
        int id;
        String name;
        int age;
        Student5(int i:String n){
        id = i;
        name = n;
        }
        Student5(int i:String n:int a){
        id = i;
        name = n;
        age=a;
        }
        void display0{System.out.println(id+" "+name+" "=age);
        }
public static void main(String arg[]){
Student5 s1 = new Student5(111,"Nishit");
Student5 s2 = new Student5(222,"Rohit",25);
s1.display();
s2.display();
        }
}
```

If we run the above program we will get an output as:

```
        111   Mohit      0
        222   Rohit      25
```

Check your progress 4

1. Which among the following can be overloaded?

a. Methods	c. Constructors
b. Class	d. Fields

3.6 The this Keyword

In Java, this keyword serves as a reference variable which mostly refers to present or current object. There are many usage of java this keyword as:

- It refers current class instance variable.
- This () calls upon current class constructor.
- It calls current class method.
- It passes an argument in method call.
- It passes an argument in constructor call.
- It returns current class instance.

Consider a program where you will find difference in outputs when this keyword is not used as shown:

```
class Studentl0 {
int id;
String name;
Student 10(int id, String name) {
id = id;
name = name;
}
void display() {System.out.println(id+" "-name);
}
public static void main(String args[]){
Studentl0 s1 = new Student10(111,”Nishit");
Student10 s2 = new Student10(321,”Rohit”);
s1.display();
s2.display();
        }
}
```

The output will be as shown, when no "this keyword" are used:

```
0 null
```

```
0 null
```

The output will be as shown, when we use "this keyword":

```
111 Mohit

222 Rohit
```

Check your progress 5

1. What will be the correct option, when "these keywords" are displayed?

```
Class Rectangle {
int length;
int breadth;
void setDiamentions (int ln, int br)
{
this.length = ln;
this.breadth = br;
}
}
Class RectangleDemo {
Public static void main(String args[]) {
Rectangle r1 = new Rectangle();
r1.setDiamentions(20,10);
System.out.println(“Length of Rectangle :” +r1.length);
System.out.println(“Breadth of Rectangle : ”+r1.breadth);
}
}
```

a. Length of Rectangle: 20 c. Both a and b

b. Breadth of Rectangle: 10 d. Neither a nor b

3.7 Instance Variables and Methods

Instance Variables in Java are variables that are defined without using static keyword. They are object specific and present outside any method declaration as their values are occurrence of any particular and not shared among instances. Consider an example shown:

```
Class Page
```

```
{
Public String page Name;
// instance variable with public access
Private int pageNumber;
// instance variable with private access
}
```

Rules for Instance variable

- Applies to any of four access level
- Marked as final
- Marked as transient
- Cant marked as abstract
- Cant marked as synchronized
- Cant marked as strict
- Cant marked as native
- Cant marked as static

Instance Method

In Java, an instance method is linked with and works upon an object. With this result, it is necessary to develop an instance of that class so as to call upon such method. The syntax of instance method is shown:

```
reType funcName(paramList)
{
    //Body of this method
}
```

Syntax to call instance method

```
objRef.funcName(args);
```

Here:

objRef: object reference variable

funcName: is name of method

args: optional arguments

For example,

```
int length()

String substring(int start)
```

Consider an example of instance method and instance variable as shown:

```
Class Circle
{
        //Instance Variables
        double x;
        double y;
        double radius;
        //Instance method
        void scale(double a)
        {
                radius *=a;
        }
}
```

In the above example, class circle will show instance variables and instance method.

Check your progress 6

1. The instant variables can be marked as:

a. synchronized	c. native
b. strict	d. transient

3.8 Static Variables and Methods

Static Variables

Static variable is a variable which belongs to class and not to instance. It gets initialized first at the start before initialization of any instance variables. In this type of variable, only a single copy will be shared by every instances of class and is not accessed directly by class name and requires no object. The syntax of the Static variable is:

<class-name>.<variable-name>

Consider an example of static variable:

```
Public class MyStatic
{
        int var;
        static int var2;

        public static int methodStatic()
        {
                ..................
        }
}
```

In the above program, 2 variables are present, 1 is instance and other is static. To obtain multiple objects of 'MyStatic' class, we see that instance variables are placed in separate space for storage whereas static variable remains common to every objects of MyStatic class with no separate space given. It is found that the java object belongs to MyStatic class will alter static variable, as there is no use to creating object of a class to modify static variables. It is found that you can alter the class name directly.

In java, every instance methods can directly be accessed using such static variables that are initialized first and then instance variables.

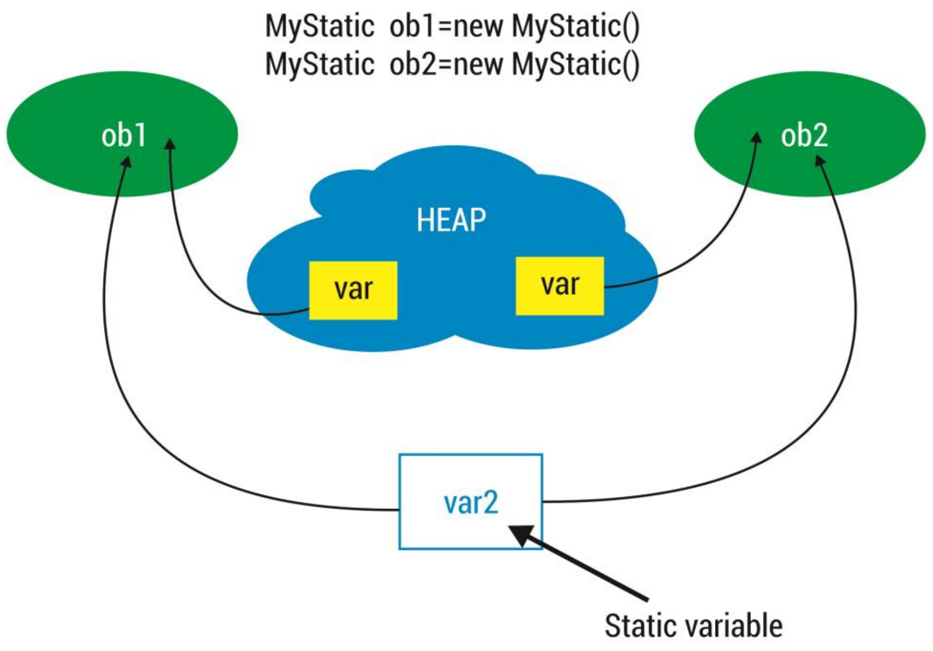


Fig 3.2 Static variable

Static Methods

Static method is that which belongs to class and not to object and can access only static data without the need of instance variables. This method can only call other static methods and not non-static methods. Such type of method can access directly by class name and requires no object. The syntax of static method is:

```
<class-name>.<method-name>
```

In Static methods, you can call all static methods of java class by class name and

```
MyStatic.methodStatic();
```

It is seen that there is an requirement to create object for MyStatic class as you can call directly as shown in static variable. Generally, these static methods access static variables. It is seen that every instance methods of a class gets access such static methods as compared to static methods, that can't be access any instance types directly.

Check your progress 7

1. Which is incorrect about static methods?

a. It belongs to class	c. It can access static data
b. It belongs to an object	d. It uses instance variables

3.9 Local Variables and its Scope

In explaining about a class, we see that a local variable is a technique in which the parameters obtained gets worked out that are used to store intermediate results. These variables are declared in a method and will be out at the time of its working. Such variables use different methods with similar name and have no relationship with each other. The syntax of Local Variable is:

```
Type varName = value;
```

Consider an example of a local variable which will describe its scope

```
Class Sample
{
    Static ctr = 0;
int i=100;
void display1()
{
        System.out.println(“Before”);
        System.out.println(“ctr = ” + ctr);
        System.out.println(“i=”+i);

        int ctr = 2, I =200;

        System.out.println(“ ”);
        System.out.println(“After”);
        System.out,println(“ctr = ”+ctr)
}
void display2()
{
System.out.println("Another method");
System.out.println("ctr = " +ctr);|
System.out.println(“i = "+i );
}
}
class LocalVariables
{
        public static void main(String args[])
        {
        Sample sObj = new Sample();
        sObj.display1();
        System.out.println("");
        sObj.display2();
        }
}
```

From the output of this program we see that there are two results once before execution and second after execution.

Before

```
ctr = 0

i = 100
```

After

```
ctr = 2
```

i = 200

Check your progress 8

1. Which is correct about local variables?

   a. It is declared inside a method

   b. It exist at the time of its execution

   c. It carries no relationship among each other

   d. All of these

3.10 Method Overloading

Method overload exists when a class having many methods with similar name but having different parameters, then under such situation the method will overload. Under this situation, if you have to do only single operation with similar name, then you will find that the method increases the readability of a program. If you have to perform addition with required numbers but you have n number of arguments and when you write method as a=int,int in case of two parameters and b=int,int,int in case of three parameters, then you will find difficulty and also programmers can't understand the behaviour of such method. This will happen because its name differs. Under such scenario, we do method overloading to have program quickly.

There are two ways in which you can overload the method:

- It can be done by changing number of arguments
- It can be done by changing data type

Consider two examples shown below where the Method of Overloading is explained by changing the number of arguments and by changing data type of argument.

Example of Method Overloading by changing the no. of arguments

Here you will find two overloaded methods, first sum method performing addition of two numbers and another is sum method performing addition of three numbers.

```
class Calculation{
void sum(int a,int b){System.out.println(a+b);}
void sum(int a,int bint c){System.out.println(a+b+c);}
public static void main(String args(){
Calculation obj=new Calculation();
obj.sum(10,10,10);
obj.sum(20,20);
    }
}
```

If you run the above program you will find that its output will be:

```
30

40
```

Example of Method Overloading by changing data type of argument

Here, you will see that two overloaded methods exists that differs in data type. First will be the sum method which gets two integer arguments and another will be the sum method that gets two double arguments.

```
class Calculation2{
void sum(int a,int b){System.out.println(a+b);}
void sum(double a,double b){System.out.println(a+b);}
public static void main(String args()){
Calculation2 obj=new Calculation2();
obj.sum(10.5,10.5);
obj.sum(20,20);
}
}
```

When you run this program, you will find the output as:

```
21.0

40
```

Check your progress 9

1. What will be the output?

```
class OverloadingCalculation3 {
void sum(int a,long b){System.out.println("a method invoked"):}
void sum(long a,int b){System.out.println(“b method invoked"):}
public static void main(String args[]){
OverloadingCalculation3 obj=new OverloadingCalcula±ion3();
obj.sum(20,20);//now ambiguity
      }
}
```

a. 20 c. 10

b. 40 d. Compile Time Error

3.11 Argument Passing

In Java, arguments are the list of parameters which gets passed in Java Programme during the start of any programme. In this, if arguments.length > 0, then it will checks if any arguments has been provided.

It is found that when you have pass-by-value parameter, then a copy of argument is kept safe in memory location which is given for formal parameter. Under such circumstances, any changes made to formal parameter in method have no effect on the value of argument which is applied back in calling method.

Also, when a parameter is pass-by-reference, then the memory address of an argument is passed to method which makes formal parameter for argument. Here the change made to formal parameter within the method gets reflected in value of an argument in case when a control is back to calling function.

Normally, it is found that all primitives are pass-by-value period into a method making a copy of primitive that carried out in such method. Hence, the value of copy will alter inside the method keeping the original value unchanged. Consider an example of the program:

```
public class TestPassPrimitive {
static void doSomething(int m) {
m=m+2;
System.out.println("The new value is " +m ".");
}
public static void main(String[] args) {
int m=5;
System.out.println("Before doSomething, m is " +m ".");
doSomething(m);
System.out.println("After doSomething, m is "+m+ ".");
        }
}
```

If we run such program, we see that the output will be shown as:

```
Before doSomething, m is 5.
The new value is 7.
After doSomething, m is 5.
```

Check your progress 10

1. Arguments is a list of:

a. Parameters
c. Array
b. Programs
d. None of above

3.12 Wrapper Classes

The wrapper classes are part of java.lang package, which is imported by default into all Java programs. The wrapper classes in java serves two main functions:

- Providing mechanism to ‘wrap’ primitive values in an object to provide activities for objects.
- Providing assortment of utility functions for primitives such as converting primitive types to bases as binary, octal or hexadecimal

It is found that these statements will show difference among primitive data type and object of wrapper class:

```
int x = 35;

Integer y = new Integer(44);
```

Here, the statement will declares int variable named x and initializes value 35. Also, the object gets initialized with value 44 with reference to object assigned to object variable y.

It is seen that in Java, there appears to be a wrapper class for every primitive date type. Such class will summarize particular value for primitive data type. In wrapper class we see that int is for Integer, float is for Float and so on.

Types

Primitive data type	**Wrapper class**
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
char	Character
boolean	Boolean

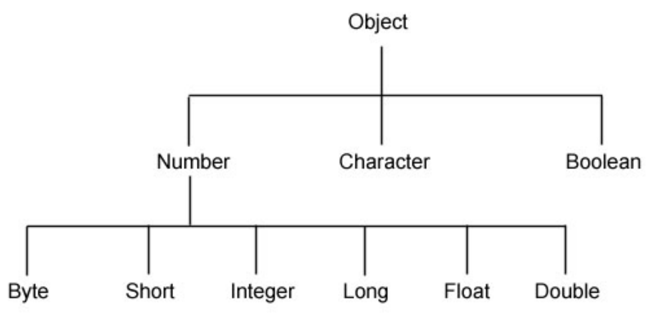


Fig 3.3 Wrapper Classes

Check your progress 11

1. The wrapper classes in java are:

a. private

b. serializable

c. immutable

d. finale

3.13 System Class

In Java, we see that a System Class:-

- Is a core class.
- Shows number of method and class fields.
- Is available in java.lang package.

System class carries Standard input, Standard output, error stream output, loading files and libraries that can be access to external properties and environment variables with utility method for quickly copying array.

Unique Features in System Class:-

- It can directly interact with device
- It has all library in regards to input and output Stream
- It takes input and out specific buffer from Operating System.
- It carries reference of input and output stream which is static when system class loads into memory
- It interacts automatically with operating system

The internal structure of System class with method and variables:-

```
public final class System extends Object
{
public static void void registerNative();
static
{
registerNative();
}
public final static InputpuStream in=null;
public final static PrintStream out=null;
public final static PrintStream err = null;
public static void setOut(PrintStream out){}
public static void setErr(PrintStream err){}
public static void setIn(InputStream in){}

public static Console console() {}
public static native long nanoTtme();
}
```

During the class loading, the system class is loaded initially in static block where call register Natives() method will call to JNI(java Native Interaction). The JNI having inbuilt code will interact with Operating System and loads all library specified by JVM. Further, the JVM will receive all static reference variable such as public final static Print Stream out=null; The JVM will read the corresponding code and will instruct JNI to get Output Buffer from Operating System, With this, the JNI run its method and shows Output Stream Buffer

Check your progress 12

1. Which is not a feature of Java System Class?

   a. It will not interact directly with device

   b. It carries all library having input and output Stream

   c. It takes input/output specific buffer from Operating System

   d. All of above

3.14 Garbage Collection

In Java, garbage collection refers to mechanism which describes about heap memory that identifies which object is in use and which not and further will delete unused objects. In used object, some part of program maintains a pointer to that object, whereas in case of unused object, there will be no referenced with any part of a program. Hence the memory used by unreferenced object gets broken.

In Java, it is seen that the mechanism of de-allocating of memory is handled directly by garbage collector who has certain process steps as shown:

Step 1: Marking

The initial step is marking where garbage collector will decide which pieces of memory are in use and which are not.

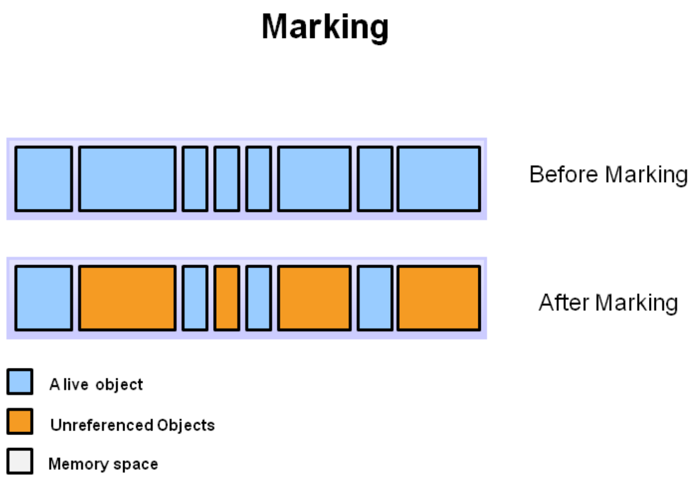


Fig 3.4 Marking process in Garbage

Step 2: Normal Deletion

Normal deletion will remove unreferenced objects and leaves behind referenced objects and pointers to free space.

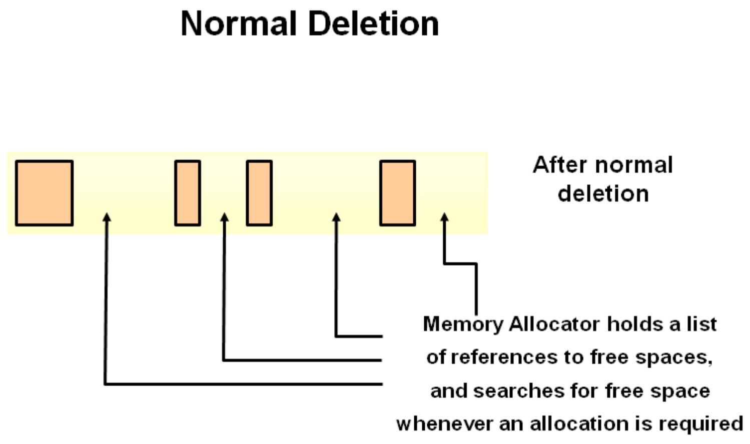

Normal Deletion
After normal deletion
Memory Allocator holds a list of references to free spaces, and searches for free space whenever an allocation is required

Fig 3.5 memory allocator

To improve the performance apart from adding deleting unreferenced objects, you can also compact left over part of referenced objects. With this, the new memory allocation becomes easier and faster as shown in fig 3.6.

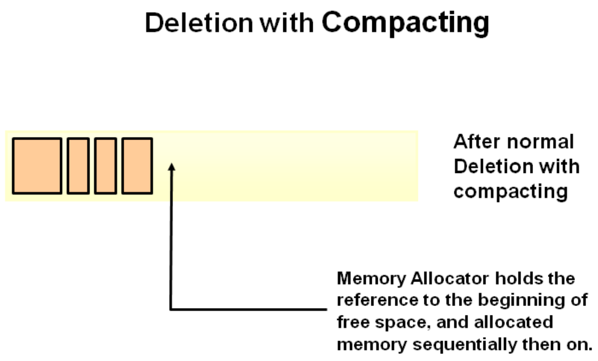

Deletion with Compacting
After normal Deletion with compacting
Memory Allocator holds the reference to the beginning of free space, and allocated memory sequentially then on.

Fig 3.5 memory deletion

Now, we see that any new objects are allotted to eden space where both survivor spaces start out empty.

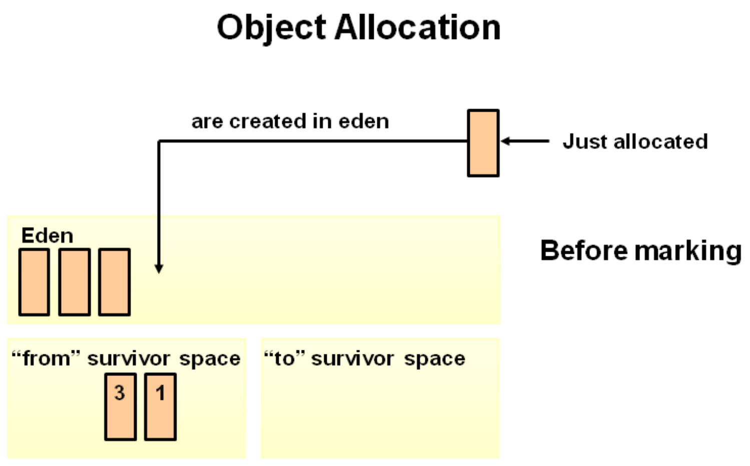


Fig 3.6 Object allocation

When the eden space fills up, a minor garbage collection is triggered.

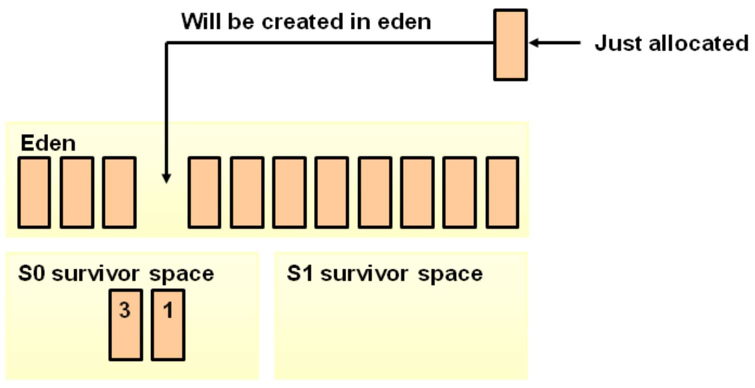


Fig 3.7 Filling of eden space

Now it is seen that the referenced objects are shifted to first survivor space. Here the unreferenced objects get deleted on clearing of eden space.

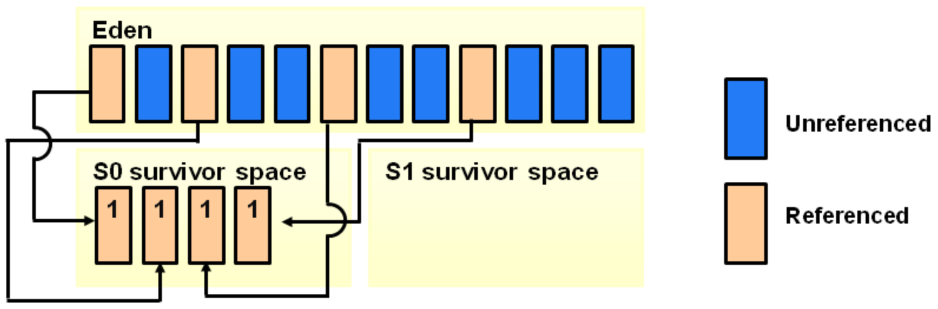


Fig 3.8 Copying of referenced objects

We see that after such operation, the unreferenced objects are deleted and referenced objects are moved to survivor space. Hence, they are moved to the second survivor space (S1). It is seen that objects from last minor GC on first survivor space (S0) carries age incremented and moves toward S1.

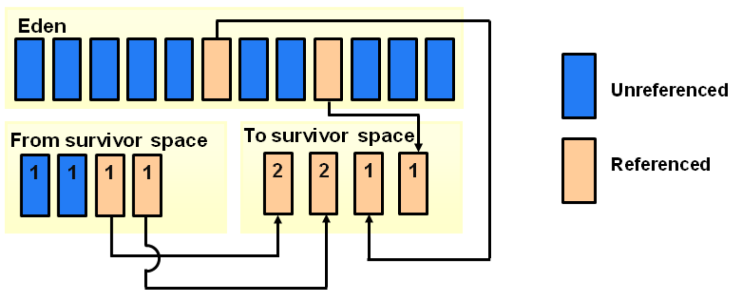


Fig 3.9 Object Aging

It is found that the survivor spaces switch and referenced objects are moved to S0. Eden and S1 are cleared.

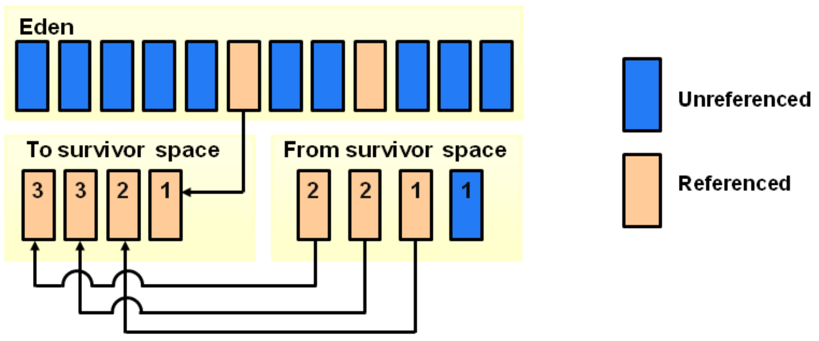


Fig 3.10 adding aging

After a minor GC, when aged objects reach a certain age threshold they are promoted from young generation to old generation.

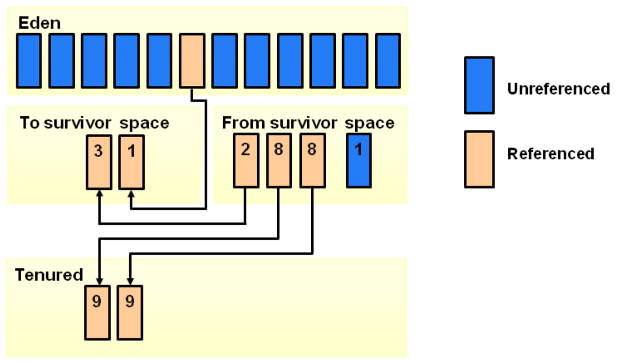


Fig 3.11 Promotion

With working of minor GCs to occur objects that continues to promote old generation space.

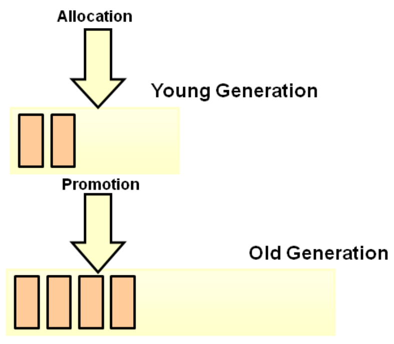


Fig 3.12 Promotion

Check your progress 13

1. Which is correct in case of garbage collector?

   a. It describes about heap memory

   b. It describes about the status of object

   c. It deletes unused objects

   d. All of these

3.15 Skills Check

In Java lots of skills are used to study about Java classes which gets created using keyword. The concept of three classes as Variables, Methods and Constructors are easy to understand.

The skills used in studying simple class program can be applied to highlight class name, name of object, object creation using new and creating call using constructor.

The skills involved in solving and studying various constructor classes has similar name as class where class have more than one constructor. The skills in solving constructor overloading mechanism explains about number of constructors having different parameter lists which gets distinguishes by considering number of parameters in list and its type.

Check your progress 14

1. In Java lots of skills are used to study about Java classes which gets created using keyword

   a. True

   b. False

3.16 Exercises

Example 1:

Program to create a class named as Square having variables height and width, overloaded constructors assigning value to variables

```
class Square
{
       int height;
       int width;

       Square()
       {
               height  = 0;
               width = 0;
        }
        Square(int side)
        {
               height  = width  = side;
         }

         Square(int sideh, int sidew)
         {
                height  = sideh;
                width = sidew;
          }
}

class ImplSquare
{
    public static void main(String args[])
    {
        Square sObj1 = new Square();
        Square sObj2 = new Square(5);
        Square sObj3 = new Square(2,3);

        System.out.println("Variable values of object1 : ");
        System.out.println("Object1 height = " + sObj1.height);
        System.out.println("Object1 width = " + sObj1.width);
        System.out.println("");

        System.out.println("Variable values of object2 : ");
        System.out.println("Object2 height = " + sObj2.height);
        System.out.println("Object2 width = " + sObj2.width);
        System.out.println("");

        System.out.println("Variable values of object3 : ");
        System.out.println("Object3 height = " + sObj3.height);
        System.out.println("Object3 width = " + sObj3.width);
     }
}
```

Output

```
Variable values of object1:
Object1 height = 0
Object1 width = 0


Variable values of object2:
Object height = 5
Object width = 5

Variable values of object3:
Object height = 2
Object width = 3
```

3.17 Let Us Sum Up

In this unit we have seen that Java classes are created using keyword class where you can define objects. It is seen that a class contains three types of items Variables, Methods and Constructors.

It is studied that a simple class program can be designed where we will highlight class name, name of object, object creation using new and creating call using constructor.

It is found that constructor in class carries similar name as class, so class can have more than one constructor. It is basically methods in Java which initialises objects.

It is studied that constructor overloading is a mechanism where a class can have many number of constructors having different parameter lists that gets distinguishes constructors by considering number of parameters in list and its type.

3.18 Answers for Check Your Progress

Check your progress 1

Answers: (1-a)

Check your progress 2

Answers: (1-d)

Check your progress 3

Answers: (1-d)

Check your progress 4

Answers: (1-c)

Check your progress 5

Answers: (1-c)

Check your progress 6

Answers: (1-d)

Check your progress 7

Answers: (1-b)

Check your progress 8

Answers: (1-d)

Check your progress 9

Answers: (1-d)

Check your progress 10

Answers: (1-a)

Check your progress 11

Answers: (1-d)

Check your progress 12

Answers: (1-a)

Check your progress 13

Answers: (1-d)

Check your progress 14

Answers: (1-a)

3.19 Glossary

1. **Block -** It is a group of consecutive statements having similar indentation.
2. **Body -** It is a block which is present in a compound statement that follows the header.

3. **Nesting -** In programming, it is the location of one program structure inside another

4. **Boolean expression -** It is an expression which shows the result as true or false.

5. **Comparison operators -** These are operators such as ==, !=, >, <, >=, and <= which compares two values.

3.20 Assignment

Explain the principle of constructor overloading?

3.21 Activities

Discuss in your group about garbage collection.

3.22 Case Study

In the program shown, can the program be there in any.java file? Comment.

```
class Test {
static int x;
int k;
// constructor with 2 args
public Test( int n, int m) {
x = n;
k = m;
        }
public static void main(String[] args) {
Test t1 = new Test(10, 20);
Test t2 = new Test(30, 40);
System.out.print(t1.x +" ");
System.out.print(t1.k  +" ");
System.out.print(t2.x +" ");
System.out.println(t2.k);
        }
}
```

3.23 Further Readings

1. Learning Programming by Peter Norvig's
2. Approach programming with a more positive by P.Brian.Mackey

Block Summary

The block gives detailed knowledge to user or students in the areas of control structures and its features. The framing of control structures and statements with use of operators and loops will help to gain extra knowledge. Each topic is explained with examples and diagrams which made clear to students and gives lot of understanding about programming skills. The knowledge about different types of operators with their features and applications of control loop structures really allow students to work.

After completing this block, students will be able to know more about control structures and its features with basic operations about operators and statements. The concept of constructor and garbage collector will allow him to work on certain Java control platforms. After this block, students will get knowledge and confidence about working with loops, operators, structures and statements.

Block Assignment

Short Answer Questions

1. Write short note on Java classes?
2. What is the syntax of While...do loop?
3. What is the use of this class in Java?
4. What is the idea of break statement?
5. What is FOR loop?

Long Answer Questions

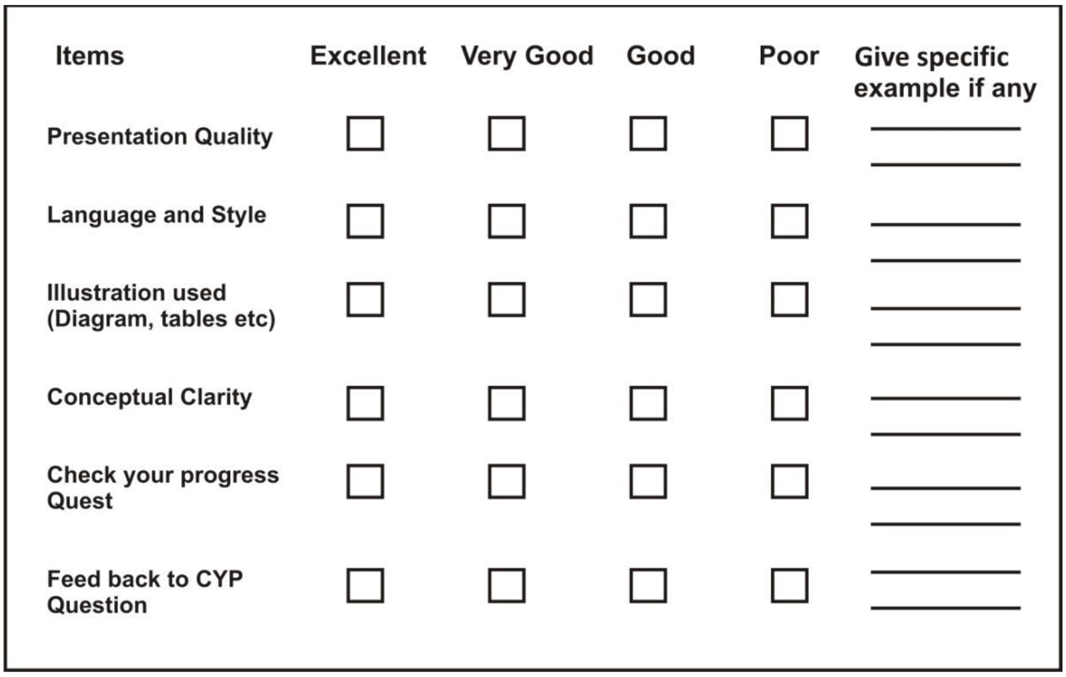
1. How to write a simple class program in Java?
2. Explain the features of garbage collector?
3. Classify different types of operators in Java?

Enrolment No. ☐

1. How many hours did you need for studying the units?

Unit No	**1**	**2**	**3**	**4**
Nos of Hrs				

2. Please give your reactions to the following items based on your reading of the block:



Items	Excellent	Very Good	Good	Poor	Give specific example if any
Presentation Quality	☐	☐	☐	☐	
Language and Style	☐	☐	☐	☐	
Illustration used (Diagram, tables etc)	☐	☐	☐	☐	
Conceptual Clarity	☐	☐	☐	☐	
Check your progress Quest	☐	☐	☐	☐	
Feed back to CYP Question	☐	☐	☐	☐	

3. Any Other Comments

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

“Education is something which ought to be brought within the reach of every one.”

- Dr. B. R. Ambedkar

Dr. Babasaheb Ambedkar Open University
Jyotirmay' Parisar, Opp. Shri Balaji Temple, Sarkhej-Gandhinagar Highway, Chharodi, Ahmedabad-382 481.

OBJECT ORIENTED CONCEPTS

PGDCA-102

BLOCK 3:

INHERITANCE,
INTERFACE, PACKAGES
AND EXCEPTIONS IN JAVA

Dr. Babasaheb Ambedkar Open University
Ahmedabad

OBJECT ORIENTED CONCEPTS

Knowledge Management and
Research Organization
Pune

Editorial Panel

Author
Er. Nishit Mathur

Language Editor
Prof. Jaipal Gaikwad

Graphic and Creative Panel
Ms. K. Jamdal
Ms. Lata Dawange
Ms. Pinaz Driver
Ms. Tejashree Bhosale
Mr. Kiran Shinde
Mr. Akshay Mirajkar

Acknowledgment
Every attempt has been made to trace the copyright holders of material reproduced in this book. Should an infringement have occurred, we apologize for the same and will be pleased to make necessary correction/amendment in future edition of this book.
The content is developed by taking reference of online and print publications that are mentioned in Bibliography. The content developed represents the breadth of research excellence in this multidisciplinary academic field. Some of the information, illustrations and examples are taken "as is" and as available in the references mentioned in Bibliography for academic purpose and better understanding by learner.'

ROLE OF SELF INSTRUCTIONAL MATERIAL IN DISTANCE LEARNING

The need to plan effective instruction is imperative for a successful distance teaching repertoire. This is due to the fact that the instructional designer, the tutor, the author (s) and the student are often separated by distance and may never meet in person. This is an increasingly common scenario in distance education instruction. As much as possible, teaching by distance should stimulate the student's intellectual involvement and contain all the necessary learning instructional activities that are capable of guiding the student through the course objectives. Therefore, the course / self-instructional material are completely equipped with everything that the syllabus prescribes.

To ensure effective instruction, a number of instructional design ideas are used and these help students to acquire knowledge, intellectual skills, motor skills and necessary attitudinal changes. In this respect, students' assessment and course evaluation are incorporated in the text.

The nature of instructional activities used in distance education self-instructional materials depends on the domain of learning that they reinforce in the text, that is, the cognitive, psychomotor and affective. These are further interpreted in the acquisition of knowledge, intellectual skills and motor skills. Students may be encouraged to gain, apply and communicate (orally or in writing) the knowledge acquired. Intellectual-skills objectives may be met by designing instructions that make use of students' prior knowledge and experiences in the discourse as the foundation on which newly acquired knowledge is built.

The provision of exercises in the form of assignments, projects and tutorial feedback is necessary. Instructional activities that teach motor skills need to be graphically demonstrated and the correct practices provided during tutorials. Instructional activities for inculcating change in attitude and behavior should create interest and demonstrate need and benefits gained by adopting the required change. Information on the adoption and procedures for practice of new attitudes may then be introduced.

Teaching and learning at a distance eliminates interactive communication cues, such as pauses, intonation and gestures, associated with the face-to-face method of teaching. This is particularly so with the exclusive use of print media. Instructional activities built into the instructional repertoire provide this missing interaction between the student and the teacher. Therefore, the use of instructional activities to affect better distance teaching is not optional, but mandatory.

Our team of successful writers and authors has tried to reduce this.

Divide and to bring this Self Instructional Material as the best teaching and communication tool. Instructional activities are varied in order to assess the different facets of the domains of learning.

Distance education teaching repertoire involves extensive use of self-instructional materials, be they print or otherwise. These materials are designed to achieve certain pre-determined learning outcomes, namely goals and objectives that are contained in an instructional plan. Since the teaching process is affected over a distance, there is need to ensure that students actively participate in their learning by performing specific tasks that help them to understand the relevant concepts. Therefore, a set of exercises is built into the teaching repertoire in order to link what students and tutors do in the framework of the course outline. These could be in the form of students' assignments, a research project or a science practical exercise. Examples of instructional activities in distance education are too numerous to list. Instructional activities, when used in this context, help to motivate students, guide and measure students' performance (continuous assessment)

PREFACE

We have put in lots of hard work to make this book as user-friendly as possible, but we have not sacrificed quality. Experts were involved in preparing the materials. However, concepts are explained in easy language for you. We have included may tables and examples for easy understanding.

We sincerely hope this book will help you in every way you expect.

All the best for your studies from our team!

Contents

BLOCK 1: INTRODUCTION TO PROGRAMMING WITH JAVA

UNIT 1 PROBLEM SOLVING WITH COMPUTERS
Need of programming, Algorithms, Flowcharts, Pseudo-Code, Programming Languages Overview, and Overview of Object Oriented Concepts, Skills check, Exercises

UNIT 2 INTRODUCTION TO JAVA
History, Overview of JVM & Bytecode, Java versions (J2me etc.), Java class library, JDK, Your First Java Application, Skills check, Exercises

UNIT 2 BEGINNING WITH JAVA PROGRAMMING
Java Keywords, Comments in Java, Variables and Assignments, Strings and Characters, Arithmetic Operators and Expressions, Type Conversion in Expressions, Type Conversion, Skills check, Exercises

BLOCK 2: JAVA CONTROL STATEMENTS, OPERATORS AND CLASSES

UNIT 1 JAVA CONTROL STATEMENTS
The if statement, The if-else statement, Blocks of Code, The for statement, Increment & Decrement operators, Backslash codes, Relational and Boolean logical operators, Ternary operators, Skills check, Exercises

UNIT 2 MORE ABOUT CONTROL STATEMENTS AND OPERATORS
Nested if statement, Variations of the for loop, The while loop, The do loop, Nested loops, The break statement, The continue statement, The switch statement, The bitwise operators, Skills check, Exercises

UNIT 3	**CREATING CLASSES**
	The General form of a class, Creating Simple Classes, Adding Constructors, Constructor Overloading, The this keyword, Instance variables and methods, Static variables and methods, Local variables and its scope, Method overloading, Argument Passing, Wrapper Classes, System Class, Garbage Collection, Skills check, Exercises

BLOCK 3:	**INHERITANCE, INTERFACE, PACKAGES AND EXCEPTIONS IN JAVA**
UNIT 1	**INHERITANCE**
	Overview of Re-usability, Subclasses, Inheritance and Variables, Method Overriding, Inheritance and Methods, Inheritance and Constructors, Class Modifiers, Variable Modifiers, Constructor Modifiers, Method Modifiers, Object and Class classes, Skills Check, Exercises
UNIT 2	**INTERFACES AND PACKAGES**
	Interfaces, Interface References, Interface Inheritance, The instanceof Operator, Packages, The import statement, Access control and Packages, Skills Check, Exercises
UNIT 3	**EXCEPTIONS**
	Overview of Exceptions, Exception Handling, Catch Block and searches, The throw statement, Exception and Error classes, The throws clause, Custom exceptions, Skills check, Exercises

BLOCK 4:	**JAVA CLASS LIBRARY, FILE HANDLING AND GUI**
UNIT 1	**INTRODUCTION TO JAVA CLASS LIBRARY**
	Classes of java.util package, Classes of java.net package, Classes of java.lang package, Overview of Collection Framework, Skills check, Exercises

UNIT 2 **FILE HANDLING IN JAVA**

Overview of File Handling, File Class in Java, Using Character Stream Classes, Using Byte Stream Classes, Using Scanner and Console Classes in Java, Skills check, Exercises

UNIT 3 **BUILDING GRAPHICAL USER INTERFACE (SWING)**

Building applications using classes related to Graphical Interface, Creating Layouts in GUI, Using Events in GUI applications, Skills check, Exercises

Dr. Babasaheb
Ambedkar
Open University

PGDCA-102

OBJECT ORIENTED CONCEPTS

BLOCK 3: INHERITANCE, INTERFACE, PACKAGES AND EXCEPTIONS IN JAVA

UNIT 1
INHERITANCE 03

UNIT 2
INTERFACES AND PACKAGES 28

UNIT 3
EXCEPTIONS 47

BLOCK 3: INHERITANCE, INTERFACE, PACKAGES AND EXCEPTIONS IN JAVA

Block Introduction

Inheritance can be defined as the process where one object acquires the properties of another. With the use of inheritance the information is made manageable in a hierarchical order. Subclass in Java is a method of inheritance which appears from Java superclass. It is seen that an interface is mixture of abstract methods where a class implements an interface that will inherit abstract methods of interface. We can declare a reference variable of interface type where interface reference variable can be created which shows object that will implements interface. It is found that exception handling is considered as one of the powerful arrangement to handle runtime errors in certain conditions such that normal flow of application can be maintained.

In this block students will be able to know about knowledge related to Inheritance and its relationship with subclass. The types of variables with their occurrence are illustrated with examples. Students in this block will be given idea on packages and interface with knowledge about inherit abstract methods of interface. The information related to declaring a reference variable of interface is well explained with examples.

After completing this block, students will be aware about abnormal, unexpected or extraordinary events or conditions that occur at the runtime of a program with knowledge about throw statement and clauses. With such detailed knowledge on Inheritance, Interface, Packages and Exceptions in Java, any student can able to work on simple Java platform.

Block Objective

After learning this block, you will be able to understand:

- The concept of Inheritance
- Idea related to Subclass
- Knowledge about different variables
- Detailed about constructor and its use

- Ide about packages
- Concept of reference variable
- Features of instance of operator
- Idea about exception handling

Block Structure

Unit 1: Inheritance

Unit 2: Interfaces and Packages

Unit 3: Exceptions

UNIT 1: INHERITANCE

Unit Structure

1.0 Learning Objectives

1.1 Introduction

1.2 Overview of Re-usability

1.3 Subclasses

1.4 Inheritance and Variables

1.5 Method Overriding

1.6 Inheritance and Methods

1.7 Inheritance and Constructors

1.8 Class Modifiers

1.9 Variable Modifiers

1.10 Constructor Modifiers

1.11 Method Modifiers

1.12 Object and Class Classes

1.13 Skills Check

1.14 Exercises

1.15 Let Us Sum Up

1.16 Answers for Check Your Progress

1.17 Glossary

1.18 Assignment

1.19 Activities

1.20 Case Study

1.21 Further Readings

1.0 Learning Objectives

After learning this unit, you will be able to understand:

- Subclasses

- Inheritance and Variables
- Method Overriding
- Inheritance and Methods
- Constructors

1.1 Introduction

Inheritance can be defined as the process where one object acquires the properties of another. With the use of inheritance the information is made manageable in a hierarchical order. When we talk about inheritance, the most commonly used keyword would be extends and implements. These words would determine whether one object IS-A type of another. By using these keywords we can make one object acquire the properties of another object.

Inheritance is a compile-time mechanism. A super-class can have any number of subclasses. But a subclass can have only one superclass. This is because Java does not support multiple inheritance.

1.2 Overview of Re-usability

Inheritance serves as pillar of object-oriented programming which is actually one of various techniques for getting broader goal of reusability. With reusability, creation of new class exists that uses features of existing class without recoding those features. There are actually several ways to achieve reusability in object-oriented programming.

Inheritance means reusing code in a hierarchical structure. For instance, a Basic Programmer is a Programmer is a Worker is a Person is an Animal. All Animals have heads, and therefore the Head property of a Basic Programmer should inherit all the general features of Animal heads plus all the features of Person heads plus all the features of Worker heads plus all the features of Programmer heads. When creating a Head property for a Basic Programmer object, you should need to write only the head code unique to Basic Programmers.

Check your progress 1

1. Inheritance means:

   a. reusing code in hierarchical structure

   b. writing code in hierarchical structure

   c. examining code in hierarchical structure

   d. all of above

1.3 Subclasses

Subclass in Java is a method of inheritance which appears from Java superclass. Class in Java can be subclass, superclass or both. Consider a Cat class in the example shown:

```
public class Animal {
public static void hide() {
        System.out.println("The hide method in Animal.”);
        }
public void override() {
System.out.println("The override method in Animal.”);
        }
}
public class Cat extends Animal {
public static void hide() {
        System.out.println("The hide method in Cat.");
        }
public void override() {
System.out.println("The override method in Cat.");
}
public static void main(String[] args) {
        Cat myCat = new Cat();
        Animal myAnimal = (Animal)myCat;
        myAnimallide();
        myAnimal.override();
        }
}
```

Creating Subclasses

You can declare a class which is subclass of another class that is present inside Class Declaration as shown. The example below creates a subclass named SubClass which is of another class named SuperClass.

You would write:

```
        Class SubClass extends SuperClass {
        …
        }
```

The above example will declare SubClass as subclass of Superclass class. It shows that SuperClass is superclass of SubClass which inherits variables and methods from its superclass's and so on. A Java class can have only one direct superclass. Java does not support multiple inheritances.

Check your progress 2

1. Subclass in Java occurs from:

a. inheritance	c. subclass
b. superclass	d. none of above

1.4 Inheritance and Variables

Inheritance is one of the important features of Object Oriented Programming which allows a class to use properties and methods of another class. In this, the derived class is called as subclass and base class is called as super-class. It is seen that a derived class adds additional variables and methods which will differentiate derived class from base class. The syntax is shown below:

```
        Public class ChildClass extends BaseClass {

                // derived class methods extend and possibly override

        }
```

Consider an example:

```
//A class to display the attributes of the vehicle
class Vehicle {
        String color;
        int speed;
        int size;
        void attributes() {
        System.out.println("Color: " +color);
        System.out.println("Speed: " + speed);
        System.out.println("Size : " + size);
                }
        }
// A subclass which extends for vehicle
Class Car extends Vehicle {
        int CC;
        int gears;
        Void attributescar() {
        // The subclass refers to the members of the superclass
                System.out.println("Color of Car: " +color);
                System.out.println("Speed of Car: " +speed);
                System.out.println("Size of Car: " +size);
                System.out.println("CC of Car: " +CC);
                System.out.println("No of gears of Car: " +gears);
        }
}
Public class Test {
        Public static void main (String args[]) {
        Car b1 = new car();
        b1.color = "Blue";
        b1.speed = 200;
        b1.size = 22;
        b1.CC = 1000;
        b1.gears = 5;
        b1.attributescar();
        }
}
```

If we run the above program, we get:

```
        Color of Car: Blue
        Speed of Car: 200
        Size of Car: 22
        CC of Car: 1000
        No of gears of Car: 5
```

Types of inheritance

In Java, inheritance is of 5 types:

Single Inheritance

Single inheritance is simple to understand as when a class extends another class and then we call it as single inheritance. In fig 1.1, class B extends only one single class which is A. Here class A is parent class of B and class B is child class of A.

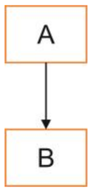


Fig 1.1 Single inheritance

Multiple Inheritance

Multiple Inheritance is a technique where one class extends more than one base class.

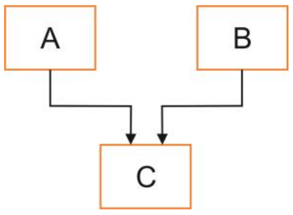


Fig 1.2 Multiple inheritance

Multilevel Inheritance

Multilevel inheritance is a mechanism in object oriented where one can inherit from derived class making base class for new class. In the fig 1.3, C is subclass of B and B is child class of A as shown.

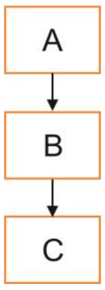


Fig 1.3 Multilevel inheritance

Hierarchical Inheritance

In such type of inheritance, one class is inherited by several sub classes. Fig 1.4 shows such arrangement where class B, C and D inherits similar class A, where A is parent class of classes B,C & D.

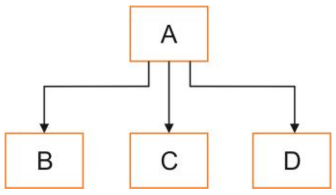


Fig 1.4 Hierarchical inheritance

Hybrid Inheritance

Hybrid inheritance is a mixture of both Single and Multiple inheritance. Fig 1.5 shows, an arrangement where hybrid inheritance is achieved in java in similar way as multiple inheritance.

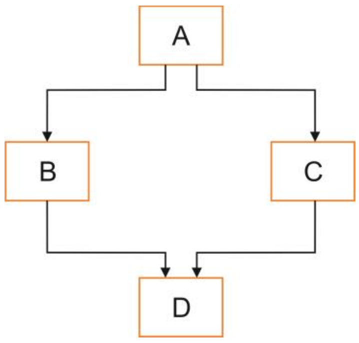


Fig 1.5 Hybrid inheritance

Variable

By Variable we mean that reserved area which is demarked inside the memory.

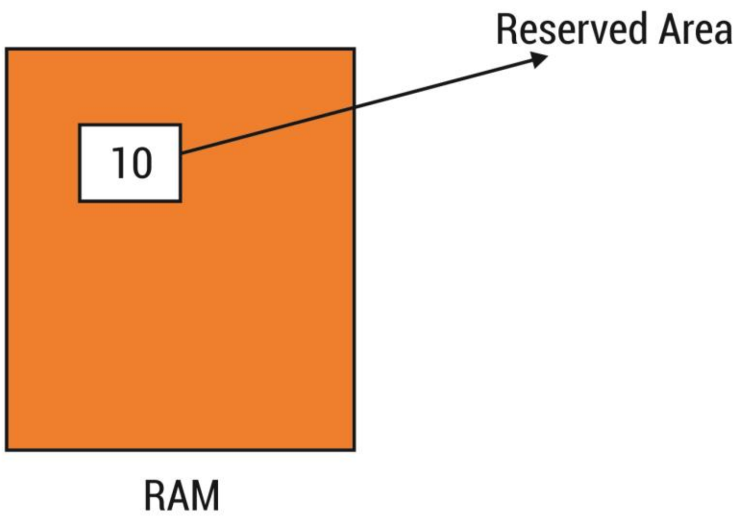


Fig 1.6 Variable

Types of Variable

There are three types of variables in java

- local variable
- instance variable
- static variable

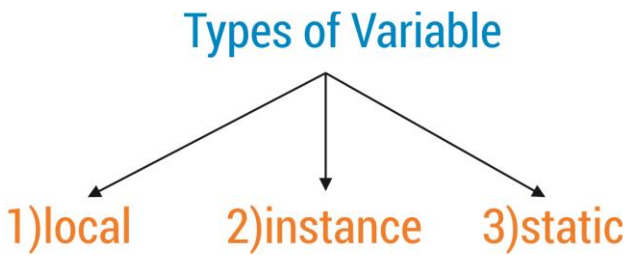


Fig 1.7 Types of Variable

Local Variable: It is a variable which is declared within method itself.

Instance Variable: It is a variable which is declared within the class but outside the method.

Static variable: It is a variable which is declared as static is called static variable.

Check your progress 3

1. In Object Oriented Programming, Inheritance uses class to use:

a. properties of another class c. both a and b

b. methods of another class d. neither a nor b

2. __________ is a variable declared inside class but outside a method.

a. Local variable c. Static variable

b. Instance variable d. None of these

1.5 Method Overriding

In a class hierarchy, when a method in a subclass has similar name along with type signature as method in its superclass, then such method in subclass gets override in superclass. In case of overridden method from a subclass, it refers to as version of particular method as explained by subclass. Consider the following:

```
// Method overriding.
Class A {
int i,j;
A(int a, int b) {
i = a;
j = b;
}
// display i and j
void show() {
System.out.println("i and j: " +i+'"'+j);
}
}
class B extends A {
int k;
B(int a, int b, int c) {
super(a, b);
k = c;
}
// display k — this overrides show() in A
void show() {
System.out.println("k: " +k);
}
}
Class Override {
Public static void main(String args[]) {
B subOb = new B(1, 2, 3);
subOb.show();     //this calls show() in B
}
}
```

The output of above program will give K:3

In this, when show( ) is call upon an object of type B, then version of show() defined inside B is used. So versio n of show( ) in B overrides version as in A. We see that version of B, superclass version of show( ) calls up in subclass version by allowing every instance variables to be shown.

```
class B extends A {
int k;
B(int a, int b, int c) {
super(a, b);
k = c;
}
void show() {
super.show();       // this calls A's show()
System.out.println("k:  " +k);
}
}
```

If this is version is substituted of A in previous program, then the output will be:

```
i and j: 1 2

k: 3
```

Super.show( ) calls superclass version of show( ). For example:

```
// Methods with differing type signatures are overloaded – not

// overridden.

Class A {
int i, j;
A(int a, int b) {
i = a;
J = b;
}
//display i and j
void show () {
System.out.println("i and j: " +i+ "" +j);
}
}
// Create a subclass by extending class A.
class B extends A {
int k;
B(int a, int b, int c) {
super (a, b);
k = c;
}
// overload show()
void show (String msg) {
System.out.println(msg + k);
}
}
Class Override {
Public static void main (String args[]) {
B subOb = new B(1, 2, 3);
subOb.show("This is k:");     //this calls show() in B
subOb.show();     //this calls show() in A
}
}
```

The output produced by this is:

```
This is k: 3
i and j: 1 2
```

Here we see that a version of show( ) in B will use string parameter which makes this as type signature which is different from one in A having no parameters. With this result, no overriding takes place.

Check your progress 4

1. _________keyword is used in subclass to call constructor of superclass.

   a. super

   b. this

   c. extent

   d. extends

1.6 Inheritance and Methods

Inheritance results as an effective method which will share code among various classes having some traits in common by allowing classes to have different parts. Fig 3.7 called as Vehicle class which carries two subclasses as Car and Truck.

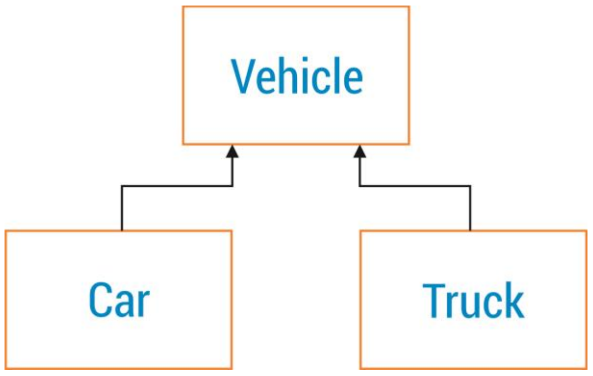


Fig 1.8 Vehicle class

It is seen that Vehicle class belongs to superclass of Car and Truck that are subclasses of Vehicle. Here, the Vehicle class contain those fields and methods which are Vehicles needed whereas Car and Truck have fields and methods particular to Cars and Trucks.

It is seen that many people declare that inheritance is a way to categorize particular class. It is studied that a Car is a Vehicle where a Truck is also a Vehicle. So, it is not how you determine superclasses and subclasses in your application. It simply shows how you need to work with them.

As seen, when a subclass extends a superclass, then all protected and public fields and methods of it gets inherited by subclass. In this, the fields and methods are part of subclass, as if subclass declared itself.

Check your progress 5

1. Select the correct option if class A is inherited by class B.

   a. class B + class A

   b. class B inherits class A

   c. class B extends A

   d. class B extends class A

1.7 Inheritance and Constructors

In Java, it is seen that constructor can be inherited as the constructor name is based on class name. While inheriting methods, the method signature should remain same. It is found that the constructors of child class will not have similar method signature of parent class. In such case, the default constructor of parent class is called. When the parent class having its own parent class then, such constructor is called and so forth. Java allows you to call other constructors beside the default one as shown:

```
public class ColorRectangle extends Rectangle
{
        Color color;
        Public ColorRectangle(int initWidth, int initHeight, Color initColor)
        {
                super ( initWidth, initHeight);
                color = initColor;
        }
}
```

This method calls a constructor of parent class that should have a public constructor which takes two int parameters.

Check your progress 6

1. A default constructor _________.

a. has no return type

b. has no argument

c. has one argument

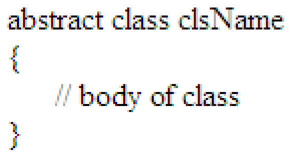
d. has one argument but no return type

1.8 Class Modifiers

In Java, class modifiers carry three possible modifiers which lead to class keyword as shown:

Keyword	**Description**
Abstract	Cannot be instantiated
Final	Cannot be extended
Public	Can be accessed by any other class Without this, access to class is limited

The syntax of class with abstract modifier is shown as:



```
abstract class clsName
{
    // body of class
}
```

The syntax of class with final modifier is shown:

```
Final class clsName
{
        //body of class
}
```

The syntax of class with public modifier is shown as:

```
Public class clsName
{
        //body of class
}
```

Check your progress 7

1. Which keyword in class modifier will limited the scope of class access?

a. abstract	c. public
b. final	d. none of these

1.9 Variable Modifiers

It is found that there exist seven possible modifiers which lead the declaration of variable as shown:

Keyword	**Description**
Final	It is constant.
Private	Access only by code in the same class
Protected	Access only by code in a subclass or same package
Public	Access by any other class
Static	It is not an instance variable.
Transient	It is not part of persistent state of this class
Volatile	Can change unexpectedly

Final variable

In Java, final variable is constant as you cannot alter its value once you create and initialize it. The syntax of Final Variable is:

```
final type varName = value;
```

Here we see that:

type: data type in java.

varName: valid identifier or variable name.

value: it is optional.

Private variable:

It is a type of variable which accessed only by code in same class. With the help of privileged method, you can set value of particular variable. The syntax of Private Variable is:

```
private type varName = value;
```

Here:

type: data type in java.

varName: valid identifier or variable name.

value: to initialize variable by given a value.

Protected variable:

It is a variable which gets access by code only present in subclass or same package. The syntax of Protected Variable is:

protected type varName = value;

Here:

type: data type in java.

varName: valid identifier or variable name.

value: to initialize variable by given a value.

Public variable:

It is a type of variable which gets accessed by any other class in any package. Its syntax is:

public type varName = value;

Here:

type: data type in java.

varName: valid identifier or variable name.

value: to initialize variable by given a value.

Static variable:

It is a type of class variable which is not an object variable. In this, number of objects created from same class having copies of class variables. The syntax is:

static type varName = value;

Here:

type: data type in java.

varName: valid identifier or variable name.

value: to initialize variable by given a value.

Transient variable:

It is a variable which is not serialized. It uses transient keyword to show Java virtual machine where the shown variable is not part of persistent state of object. Its syntax is:

transient type varName = value;

Here:

type: data type in java.

varName: valid identifier or variable name.

value: to initialize variable by given a value.

Volatile variable:

It is a variable which is treated different from other variables. In case of volatile behaviour, it will warn compiler to have fresh copies of variable instead of holding variable in registers. Its syntax is:

volatile type varName = value;

Here:

type: data type in java.

varName: valid identifier or variable name.

value: to initialize variable by given a value.

Check your progress 8

1. Which keyword is access by code that lies in same class?

a. final	c. protected
b. private	d. public

1.10 Constructor Modifiers

Constructor in Java creates an instance of a class which signifies it as creating an object. Constructor carries certain modifiers such as public, protected, private, or none. They can take only access modifiers and cannot be abstract, final, native, static, or synchronized.

It is found that there is no return type, not even void type values in it. It carries same name as their class; by convention, methods and uses name other than class name. In case of normal program convention, method will start with lowercase letter and constructor with uppercase letter. The syntax of constructor modifier is:

ConstructorModifier

ConstructorModifiers ConstructorModifier

ConstructorModifier: one of

Public protected private

It is found that a compile time error occurs if same modifier appears more than once in constructor declaration, or when constructor declares more than one of access modifiers public, protected, and private.

Check your progress 9

1. In constructor, compilation error occurs when declares more than one of _________modifiers

a. public	c. private
b. protected	d. all

1.11 Method Modifiers

In Java, method modifier is an easy feature that calls a method. Method Modifiers provides three modifiers as before, around and after. It is seen that before and after modifiers run just before and after the method without affecting original method. The around modifier will run in place of original method, with a hook to easily call original method. The benefit of using such modifier is that it helps in defining multiple modifiers in single namespace. These separate modifiers don't need to know about each other. This makes top-down design easy. Have a base class that provides the skeleton methods of each operation, and have plugins modify those methods to flesh out the specifics.

Check your progress 10

1. Which is not a basic method modifier?

a. before	c. below
b. around	d. after

1.12 Object and Class Classes

Object Class

In Java, an object class is the best class for all classes available. There are certain object class methods such as:

- equals
- toString()
- wait()
- notify()
- notifyAll()
- hashcode()
- clone

It is seen that object is an instance of a class which is produced with new operator. This new operator will return reference to new instance of class. Such reference assigns reference variable of class. Such type of process is called as instantiation.

It is seen that an object reference will show a handle to object which is developed and stored in memory. The objects in Java can carry out with references that get stored in a variable. Now to create a variable of your class type is equivalent to creating variables of primitive data types like integer or float. It is found that every time you create an object, you get a new set of instance variables comes which comes into existence that defines characteristics of particular object. To create an object of class with reference variable associated with it, and then you should allot memory for object with the help of new operator. Such type of process is called instantiating an object or creating an object instance.

Once the object is created, you can use new operator to instantiate the object. This new operator will return location of object that assigns a reference type. Consider an example where creations of Cubical objects are applied with the help of new operator.

```
public class Cube {
        int length = 10;
        int breadth = 10;
        int height = 10;
        public int getVolume() {
```

```
        return (length  * breadth * height);
        }
        public static void main(String[] args) {
        Cube cubeObj; // Creates a Cube Reference
        cubeObj = new Cube(); //Creates an Object of Cube
        System.out.println("Volume of Cube is : " +cubeObj.getVohtme());
        }
    }
```

Class

In Java, a class is a group of objects having certain common properties. It is a sort of template or blueprint from where objects are created. So a class in java contains:

- data member
- method
- constructor
- block
- class and interface

The syntax to declare a class is:

```
class <class_name> {
    data member;
    method;
}
```

Consider an example of Object and Class

In this example, a Student class is created having two data members' id and name. Now the object of Student class is created with the help of new keyword and printing objects value as:

```
class Studentl {
int id; //data member (also instance variable)
String name; //data member(also instance variable)
public static void main(String args[]){
Studentl sl=new Studentl (); //creating an object of Student
System.out.println(s1.id);
System.out.println(sl.name);
    }
}
```

If we run the above program, we find: Output: 0 null

Check your progress 11

1. Which among the following method of Object class is used to get class of an object at run time?

   a. get() c. class getclass()

   b. void getclass() d. none of above

1.13 Skills Check

In Java various skills are applied to get the result of programs that uses modifiers. Certain short cuts are required in order to handle such problems. The methods of interfaced with class, inheritance and modifiers needs special attention to solve, which requires special skills check.

Check your progress 12

1. In Java various skills are applied to get the result of programs that uses modifiers.

   a. True

   b. False

1.14 Exercises

Example 1:

Program that illustrates final variable use in java using day In Week variable

```
class FinalVariableDemo
{
    public static void main(String[] args)
    {
        //Once created and initialized, its value cannot be changed.
         final int dayInWeek = 7;
         //Below statement will not compile. you cannot change value of
dayInWeek variable.
```

```
        //dayInWeek  = 6;
        System.out.println("Number  of days in a week = " + dayInWeek);
       }
}
```

On running the above program, you will find output as:

```
Number of days in a week = 7
```

1.15 Let Us Sum Up

In this unit we have learnt that Inheritance can be defined as the process where one object acquires the properties of another. This inheritance makes the information to manage in a hierarchical order.

It is studied that a Subclass in Java is a method of inheritance which appears from Java superclass.

There exist three types of variables in Java as Local, Instance and Static. It is studied that local variable is a variable which is declared within method itself, instance variable is a variable which is declared within class but outside method and static variable is a variable which is declared as static is called static variable.

In Java, it is seen that constructor can be inherited as the constructor name is based on class name. While inheriting methods, the method signature should remain same.

1.16 Answers for Check Your Progress

Check your progress 1

Answers: (1-a)

Check your progress 2

Answers: (1-b)

Check your progress 3

Answers: (1-c)

Check your progress 4

Answers: (1-a)

Check your progress 5

Answers: (1-c)

Check your progress 6

Answers: (1-b)

Check your progress 7

Answers: (1-c)

Check your progress 8

Answers: (1-b)

Check your progress 9

Answers: (1-d)

Check your progress 10

Answers: (1-c)

Check your progress 11

Answers: (1-c)

Check your progress 12

Answers: (1-a)

1.17 Glossary

1. **Local Variable -** It is a variable which is declared within method itself.
2. **Instance Variable -** It is a variable which is declared within the class but outside the method.

3. **Static variable -** It is a variable which is declared as static is called static variable
4. **Class -** These are group of objects with common properties and are like a blueprint from where objects are created.

1.18 Assignment

How a class is initialized in java?

1.19 Activities

Discuss the types of modifiers in Java used.

1.20 Case Study

Discuss the output of this program?

```
class A {
    int i;
    int j;
    A() {
        i = 1;
        j = 2;
    }
}
class Output {
    public static void main(String args[])
    {
        A obj1 = new A();
        A obj2 = new A();
            System.out.print(obj1.equals(obj2));
    }
}
```

1.21 Further Readings

1. A Primordial Interface by Wm. Paul Rogers.
2. Java Diamonds Forever by Tony Sintes.
3. Java Routines by John D. Mitchell.
4. Object-Oriented Design and Programming by Tony Sintes.

UNIT 2: INTERFACES AND PACKAGES

Unit Structure

2.0 Learning Objectives

2.1 Introduction

2.2 Interfaces

2.3 Interface References

2.4 Interface Inheritance

2.5 Instance of Operator

2.6 Packages

2.7 The Import Statement

2.8 Access Control and Packages

2.9 Skills Check

2.10 Exercises

2.11 Let Us Sum Up

2.12 Answers for Check Your Progress

2.13 Glossary

2.14 Assignment

2.15 Activities

2.16 Case Study

2.17 Further Readings

2.0 Learning Objectives

After learning this unit, you will be able to understand:

- Interface References.
- Interface Inheritance.
- Instance of Operator.
- Packages.
- Import statement.

- Access control.

2.1 Introduction

A package is a collection of types which gives access protection and name space management. We see that there are types which are called as classes, interfaces, enumerations and annotation types. It is seen that an enumerations and annotation types exists as particular kinds of classes and interfaces. It is correctly said that types are simply classes and interfaces.

An interface is mixture of abstract methods. A class implements an interface, and will inherit abstract methods of interface. It is not a class, but if we write it, it is equivalent to writing a class. Further it is seen that a class describes itself as an attributes and behaviours of an object. An interface contains behaviours that a class implements.

2.2 Interfaces

Interface is gathering of abstract methods in which a class implements it that will interface abstract methods. It is similar to writing a class where it describes an attributes and behaviours of particular object. An interface contains behaviours that a class implements. It is similar to a class as:

- It has n number of methods.
- It is written in file having .java extension where interference name matches with file name.
- It carries a bytecode whose interface is seen .class file.
- It exists in a package where bytecode file exists in directory structure which will matches with the package name.

It is seen that an interface is different from a class as:

- It cannot be instantiate.
- It carries no constructor.
- It contains abstract methods.
- It has no instance fields.
- It is not extended but is implemented by a class.

- It can extend multiple interfaces.

Declaring Interfaces:

While declaring an interface, you use interface keyword as shown in example below:

```
/* File name: NameOfInterface.java */
Import java.lang. *;
//Any number of import statements
Public interface NameOfInterface
{
        //Any number of final, static fields
        //Any number of abstract method declarations\
}
```

Properties of Interface:

- An interface requires no abstract keyword while declaration.
- Methods in it are implicitly abstract hence abstract keyword is not required.
- Methods being implicitly public.

Example:

```
/* File name: Animal.java */
Interface Animal {
        Public void eat();
        Public void travel();
}
```

Implementing Interfaces:

While implementing class as an interface, then it seems that it is bound and agrees to do certain behaviours. When a class is not able to do every behaviour of interface, then under such case, it has to declare itself as abstract. While doing this, a class makes use of implements keyword in order to implement an interface. It is found that an implements keyword will appear in class declaration as shown:

```
/* File name: MammalInt.java */
Public class MammalInt implements Animal {
        Public void eat() {
```

```
        System.out.println("Mammal eats");
    }
    Public void travel() {
        System.out.println("Mammal travels");
    }
    Public int noOfLegs(){
    return 0;
    }
    Public static void main(String args[]){
        MammalInt m = new MammalInt();
    m.eat();
    m.travel();
        }
    }
```

If we run the above program, we will get an output as:

```
Mammal eats
Mammal travels
```

Rule of defining an interface with overriding methods employs:

- Checked exceptions not to be declared on implementation methods other than declared by interface method of declared by interface method.
- Signature of interface method and subtype should maintain at the time of overriding the methods.
- Implementation class remains as abstract and if so, then interface methods need not be implemented.

Roles of implementing interfaces are:

- Class can put into operation with more than one interface at particular time.
- Class extends only single class that implements several interfaces.
- Interface extends another interface as class extends another class.

Extending Interfaces:

Interface extends another interface where a class extends another class. For extending an interface, an extend keyword is applied where a child interface inherits methods of parent interface. This is explained with the help of an example where sports interface extends by Hockey and Football interfaces as:

```
//Filename: Sports.java
```

```
        Public interface Sports
        {
                Public void setHomeTeam(String Name);
                Public void setVisitingTeam(String name);
        }
//Filename: Football.java
Public interface Football extends Sports
{
        Public void homeTeamScored(int points);
        Public void visitingTeamScored(int points);
        Public void endOfQuarter(int quarter);
}
//Filename: Hockey.java
Public interface Hockey extends Sports
{
        Public void homeGoalScored();
        Public void visitingGoalScored();
        Public void endOfPeriod(int period);
        Public void overtimePeriod(int ot);
}
```

In this, we see that, Hockey interface carries 4 methods where it inherits 2 from Sports. So a class which implements Hockey requires implementing 6 methods. In the same way, a class which implements Football requires to explain 3 methods from Football and 2 methods from Sports.

Check your progress 1

1. Which of these keywords is used to define interfaces in Java?

a. interface	c. intf
b. Interface	d. Intf

2.3 Interface References

It is seen that in Java, you can declare a reference variable of interface type. We can create an interface reference variable which shows object which will implements the interface. On calling object method by interface reference, we find that it is a version of method that gets implemented through object to work.

We find that a method gets executed at the time of dynamically run time which allows classes to create in the code which calls methods on them.

```
Test f = new InterfaceTest();

f.call();
```

In this we see that a variable f is declared to be of interface type Test that was assigned as an instance of InterfaceTest. Here, f can be used to access call( ) method where other members cannot execute InterfaceTest class. An interface reference variable only can declare interface declaration.

Consider a program:

```
Interface Test {
        void call();
}
Class IntercafeTest implements Test {
        Public void call ()
        {
                System.out.println("call method called");
        }
}
Public class Javaapp {
        Public static void main (String[] args) {
        Test f = new InterfaceTest ();
        f.call();
        }
}
```

The output of such program will be:

```
        Call method called
```

Check your progress 2

1. Which of these access specifiers can be used for an interface?

a. Public	c. private
b. Protected	d. All of above

2.4 Interface Inheritance

Inheritance and interfaces have in common that they both have the ability to define an abstract "contract" or "protocol" that a group of concrete classes following this "contract" have to implement. This gives you the ability to handle a number of similar classes in a uniform way, which allows you to write more compact, abstract code, which again allows code reuse.

Consider a class implements interface example where one interface extends another interface as:

```
interface Printable{
void print();
}
interface Showable extends Printable{
void show();
}
class Testinterface2 implements Showable
{
    public void print(){System.out.println("Nishit");
    }
    public void show(){
    System.out.println("Mathur");
    }
    public static void main(String args[]){
    Testinterface2 obj = new Testinterface2();
    obj.print();
```

```
        obj.show();
        }
    }
```

If we run this program we get an output as:

```
Nishit
Mathur
```

Check your progress 3

1. Which keyword is used to inherit a class?

a. super	c. extent
b. this	d. extends

2.5 Instance of Operator

In Java, an instance of operator is applied to check object instance of specific type that can be class or subclass or interface. It is seen that the instance of in java is called as comparison operator as it compares instance with type which returns true or false. On using instance of operator with any variable having null value will return false.

Consider an example of java instance of operator where it will check current class as:

```
Class Simple1 {
        Public static void main(String args[]) {
        Simple1 s = new Simple1();
System.out.println(s instance of Simple); //true
        }
}
```

If we run the program, then the output will be true

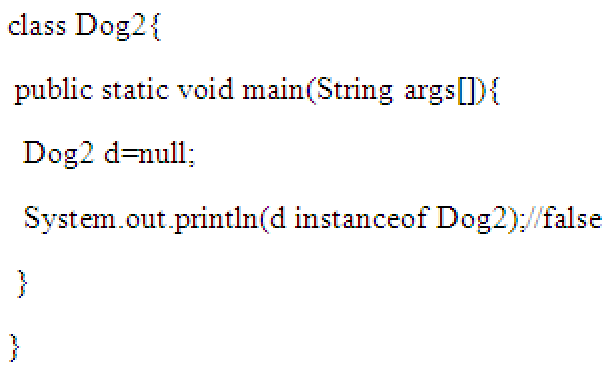
An object of subclass type is a type of parent class. In an example, a Dog extends Animal where an object of Dog called as Dog or Animal class.

```
Class Animal {}

Class Dog1 extends Animal {

//Dog inherits Animal

Public static void main(String args[]) {

Dog1 d = new Dog1();

System.out.println(d instance of Animal) //true

        }

}
```

If we run the program, then the output will be true

Instance of in java with variable having null value

On applying instance of operator having variable with null value it shows the returns false. Consider an example shown where we use instance of operator with variable having null value.



```
class Dog2{

 public static void main(String args[]){

  Dog2 d=null;

  System.out.println(d instanceof Dog2);//false

 }

}
```

If we run the program, then the output will be false

Check your progress 4

1. What will be the output?

```
Class simple1 {
Public static void main(String args[]) {
Simple1 s=new Simple1();
System.out.println(s instanceof Simple;) //true
}
}
```

a. new   c. print

b. true   d. compilation error

2.6 Packages

A java package is a group of similar types of classes, interfaces and sub-packages. Package in java can be categorized in two form, built-in package and user-defined package. There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc. Here, we will have the detailed learning of creating and using user-defined packages.

Advantage of Java Package

1) Java package is used to categorize the classes and interfaces so that they can be easily maintained.

2) Java package provides access protection.

3) Java package removes naming collision.

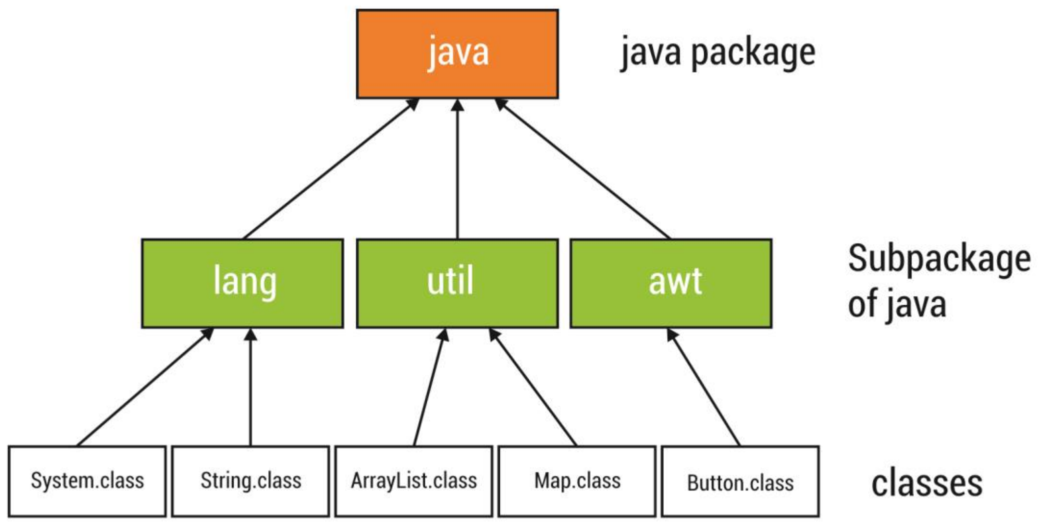


Fig 2.1 Java Package

Creating a Package

The package keyword is used to create a package in java.

```
// save as Simple.java
Package mypackl;
Public class Simple {
Public static void main(String args[]) {
      System.out.println("Welcome to package");
      }
}
```

Compiling a java package

To compile a java package and if not applying IDE, then you have to follow the syntax:

```
javac -d directory javafilename
```

In an example:

```
javac -d . Simple.java
```

We see that a -d switch will specify destination where you need to put your generated class file. You can also use any directory name like /home, d:/abc etc. In order to keep the package in same directory, you have to use . (dot).

Running a java package program

To run a java package, you should use qualified name as mypack.Simple etc in order to run the class.

To compile: javac –d . Simple.java

To Run: java mypack.Simple

Output: Welcome to package

In this we see that a -d switch tells the compiler where to put class file. The . (dot) will show the current folder.

Check your progress 5

1. Which of these keywords is used to define packages in Java?

a. pkg	c. package
b. Pkg	d. Package

2.7 The Import Statement

Import statement or keyword is applied where you want to import built-in and user-defined packages in java source file. This is done so that a class can refer to class with another package simply by using its name. It is fund that there are 3 different ways to refer to class which is available in different package.

Using fully qualified name

```
class MyDate extends java.util.Date
{
//statement;
}
```

Import class you want to use

```
import java.util.Date;
class MyDate extends Date
{
//statement.
}
```

Import classes from particular package

```
import java.util.*;
class MyDate extends Date
{
//statement;
}
```

Import statement is kept after package statement.

```
package mypack;
import java.util.*;
```

If you are not creating any package, then import statement will be the first statement of your java source file.

Static import

Static import is a feature which expands capabilities of import keyword. It will normally import static member of a class. We see that static member is called as the association where class name is outside the class. With this import, it is possible to refer to static member directly without having its class name.

It is seen that there exists two general form of static import statement:

First Form: The first form of static import statement will import single static member of a class having syntax as:

```
Import static package.class-name.static-member-name;
```

In an example:

```
import static java.lang.Math.sqrt;

//importing static method sqrt of Math class
```

Second Form: In second form of static import statement, an import of all static member of class is carried having syntax as:

```
import static package.class-type-name.*;
```

In an example

```
import static java.lang.Math.*;

//importing all static member of Math class
```

In case of not using static import, the program can be written as:

```
public class Test

{

    public static void main(String[] args)

    {

        System.out.println(Math.sqrt(144));

    }

}
```

If you run the above program, then the output will be 12.

Consider a program where static import is present:

```
Import static java.lang.Math.*;

Public class Test

{

    Public static void main(String[] args)

    {
```

```
                System.out.println(sqrt(144));
            }
        }
```

If we run this program, then the output will be 12.

Check your progress 6

1. What is the use of import statement in Java?

a. to import built-in packages	c. both a and b
b. to import user-defined packages	d. neither a nor b

2.8 Access Control and Packages

In Java we see that access control is performed by using access modifiers with each member in class definition. Java will work with four different access modifiers which are applied to any individual member of class such as:

- public
- private
- protected
- package

There are certain levels in Java where:

- Public class member access with other code
- Private class member access in its own class
- Default access class member has no access specifiers
- Protected feature of class is present to every class in same package and subclasses.
- Protected features are accessible as compared to default features.

Consider an example, where the effects of public and private access are shown as:

```
    Class test {
        Int a;              //default access
        Public int b;       //public access
```

```
          Private int c;        //private access
          // methods to access c
          Void setc(int i) {
          c = i;
          }
          Int getc() {
                 return c;
          }
   }
   Public class Main {
          Public static void main(String args[]){
          Test ob = new Test();
          ob.a = 1;
          ob.b = 2;
   //This is not OK and will cause an error
   //ob.c = 100;       //Error!
   //You must access c through its methods
   Ob.setc(100);       //OK
   System.out.println("a, b, and c: "+ob.a+
          " " +ob.b+ " " +ob.getc());
          }
   }
```

On running the above program, we will get an output as 1 2 100

Check your progress 7

1. Which is the access modifier that defines class in Java?

a. public	c. protected
b. private	d. all of these

2.9 Skills Check

There are lots of skills used to define a class in Java. There are certain packages in Java that needs skill based knowledge in order to solve. The layout of identifiers allows you to select classes in Java.

Check your progress 8

1. The layout of identifiers allows you to select classes in Java.

   a. True

   b. False

2.10 Exercises

Exercise 1:

In an example shown, an instance of operator will check whether Two-Wheeler is actually a Vehicle or Four-wheeler is actually a Vehicle.

Solution:

Invalid Use of Instance Of Operator :

```
System.out.println(v3 instance of TwoWheeler);
```

Example : IS-A Relationship Verification

```
class FourWheeler extends Vehicle{}
class TwoWheeler extends Vehicle{}
class WagonR extends FourWheeler{}
public class Vehicle
{
public static void main( String[] args )
{
        FourWheeler vl = new FourWheeler ();
        TwoWheeler v2 = new TwoWheeler();
        WagonR v3 = new WagonR();
        System.out.println(v1 instanceof Vehicle);
```

```
            System.out.println(v2  instanceof  Vehicle);
            System.out.println(v3  instanceof  Vehicle);
            System.out.println(v3  instanceof  FourWheeler);
            }
        }
```

If you run this program, the output will be:

```
true
true
true
true
```

2.11 Let Us Sum Up

In this unit we have learnt that a package is a collection of types which gives access protection and name space management which called as classes, interfaces, enumerations and annotation types.

It is seen that an interface is mixture of abstract methods where a class implements an interface that will inherit abstract methods of interface.

We can declare a reference variable of interface type where interface reference variable can be created which shows object that will implements interface.

It is seen that inheritance and interfaces both have ability to define an abstract "contract" or "protocol" that a group of concrete classes follows "contract" have to implement.

An instance of operator is applied to check object instance of specific type that can be class or subclass or interface. It is seen that the instance of in java is called as comparison operator as it compares instance with type which returns true or false.

2.12 Answers for Check Your Progress

Check your progress 1

Answers: (1-a)

Check your progress 2

Answers: (1-a)

Check your progress 3

Answers: (1-d)

Check your progress 4

Answers: (1-b)

Check your progress 5

Answers: (1-c)

Check your progress 6

Answers: (1-c)

Check your progress 7

Answers: (1-d)

Check your progress 8

Answers: (1-a)

2.13 Glossary

1. **Package -** It is a collection of types that gives access protection and name space management in Java.
2. **Interface -** In programming, interface is a mixture of abstract methods where class implements an interface.

2.14 Assignment

Compare interface and reference with examples.

2.15 Activities

Collect some information about interference and use certain code to interface it.

2.16 Case Study

Study the callback( ) method of interface reference variable and defines its output.

```
interface MyIntetface {
void callback(int param);
}
class Client implements MyInterface{
// Implement Callback's interface
public void callback(int p) {
System.out.println("callback called with " + p);
        }
}
public class Main {
public static void main(String args[]) {
MyInterface c = new Client();
c.callback(42);
        }
}
```

2.17 Further Readings

1. A Primordial Interface by Wm. Paul Rogers.
2. Java Diamonds Forever by Tony Sintes.
3. Java Routines by John D. Mitchell.
4. Object-Oriented Design and Programming by Tony Sintes.

UNIT 3: EXCEPTIONS

Unit Structure

3.0 Learning Objectives

3.1 Introduction

3.2 Overview of Exceptions

3.3 Exception Handling

3.4 Catch Block and Searches

3.5 The Throw Statement

3.6 Exception and Error Classes

3.7 The Throws Clause

3.8 Custom Exceptions

3.9 Skills Check

3.10 Exercises

3.11 Let Us Sum Up

3.12 Answers for Check Your Progress

3.13 Glossary

3.14 Assignment

3.15 Activities

3.16 Case Study

3.17 Further Readings

3.0 Learning Objectives

After learning this unit, you will be able to understand:

- Exception Handling
- Catch Block and Searches
- Throw Statement
- Exception and Error Classes
- The Throws Clause

3.1 Introduction

In Java, exceptions are abnormal, unexpected or extraordinary events or conditions which occurs during the runtime which could be file not found exception or unable to get connection. In such conditions, java will try to throw exception object.

Whereas Java exception handling is employed in order to take care of error conditions that exists in program in a sequence through necessary action. The exception handlers under such conditions can be framed to hold a particular exception which is the case of Number Format exception or group exceptions by generic exception handlers.

3.2 Overview of Exceptions

In java, exception is an event which will disturb normal flow of a program. It is an object which is thrown at runtime. In this, when you throw an exception then an object is thrown. For throwing an object as an exception, you need only those objects whose classes descend from Throwable. It means that Throwable behaves as a base class for a complete family of classes that are declared in java.lang. A small part of such family is shown in Figure 3.1.

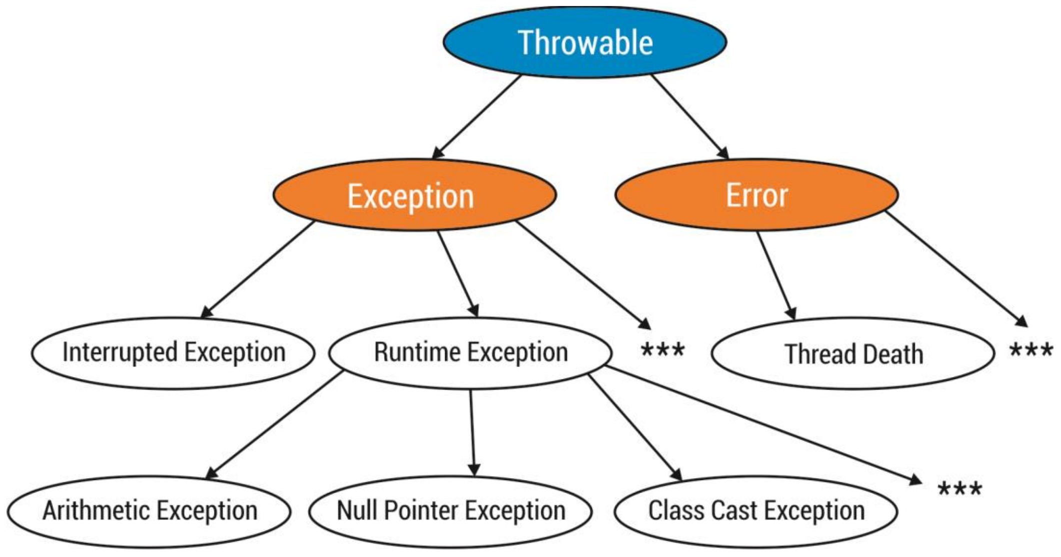


Fig 3.1 Throwable subclasses

From the fig 3.1, Throwable carries two direct subclasses as Exception and Error.

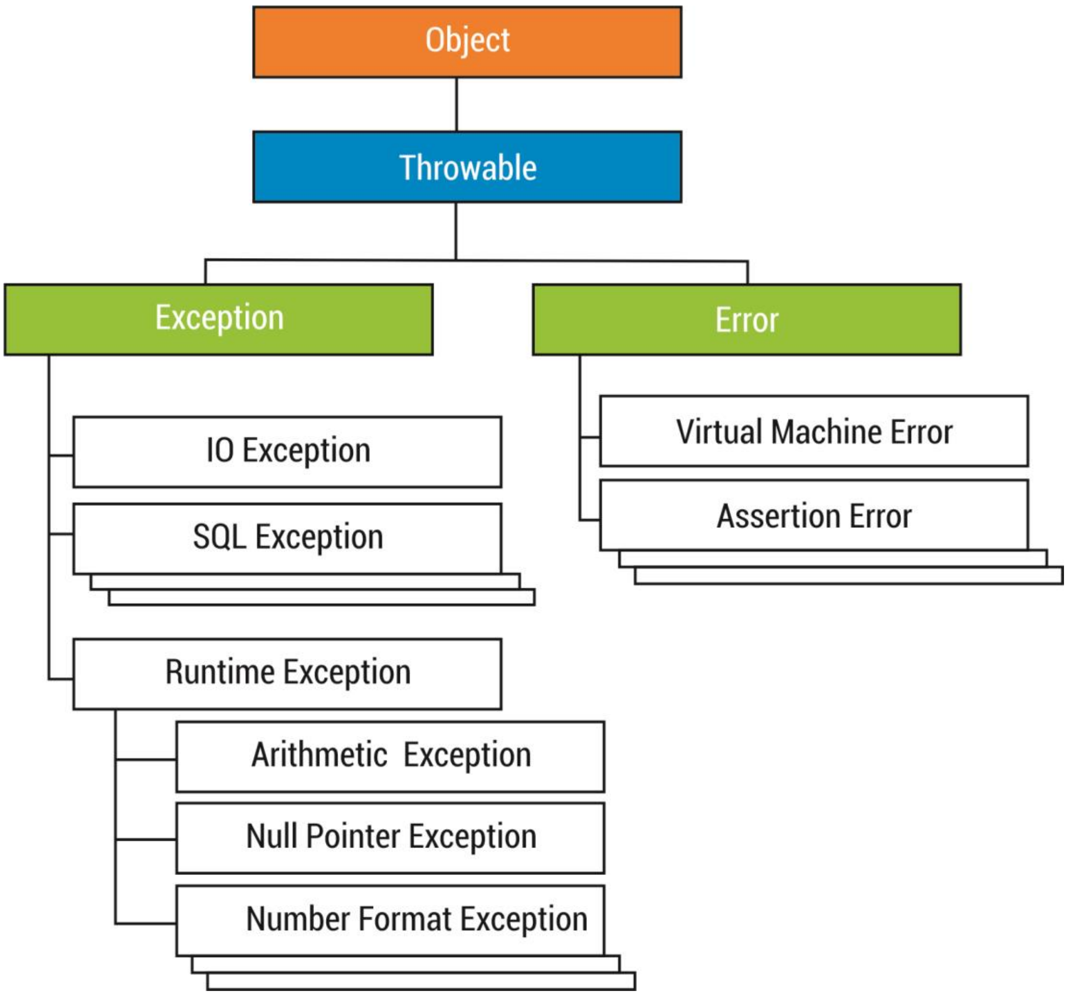


Fig 3.2 Exception Classes

It is found that an exception is thrown to signal abnormal conditions which can be handled by some catcher that ultimately results in dead thread. On the other end errors are thrown if the problems are very serious like a case of out of Memory Error. Generally, code to be written should throw only exceptions and not errors.

Apart from throwing objects with classes declared in java.lang, you can also throw objects as per your design. In order to create your own class of throwable objects, you need to declare subclass of member of Throwable family. Normally it is seen that throwable classes extends class Exception.

In case of using existing exception class from java.lang, it simply depends on certain circumstances. Many times, a class from java.lang will work perfect as if your methods is call up with invalid argument, then you throw Illegal Argument Exception which is a subclass of Runtime Exception in java.lang.

Check your progress 1

1. Which keyword is used to throw an exception in a program?

a. try	c. throw
b. finally	d. catch

3.3 Exception Handling

In Java, exception handling is considered as one of the powerful arrangement to handle runtime errors in certain conditions such that normal flow of application can be maintained. This process handles the runtime errors such as Class Not Found, IO, SQL, Remote etc. There are 5 types of keywords that are used in java exception handling.

- Try
- Catch
- Finally
- Throw
- Throws

Advantage of Exception Handling

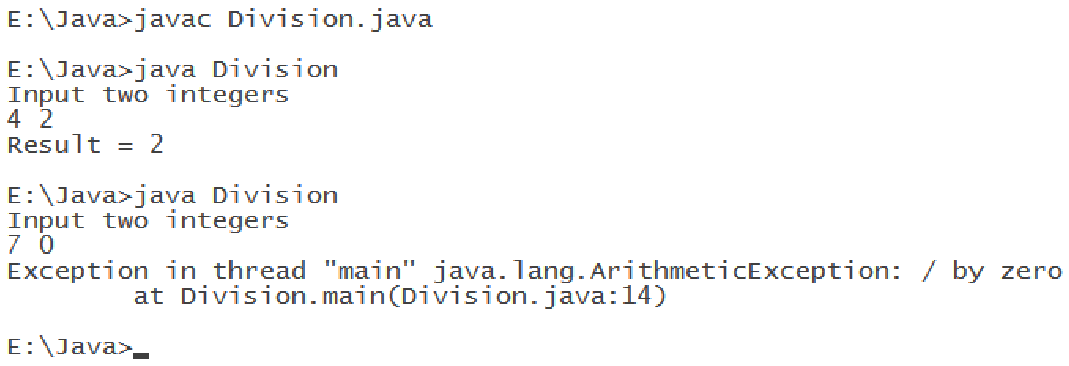
There are certain advantages of exception handling such as maintaining normal flow of application. It is seen that exceptions normally disturb flow of application for which we have to use the procedure of exception handling. Consider a simple program where there occur 10 statements and exception results in statement 5 as shown:

```
statement 1;
statement 2;
statement 3;
statement 4;
statement 5;//exception occurs
statement 6;
statement 7;
statement 8;
statement 9;
statement 10;
```

Out of 10 statements in a program, we find that an exception occurs at statement 5. So with this, the remaining code will stop working as statement 6 to 10 will not run. Now if we apply exception handling mechanism, we see that rest of the statement will start working. Now consider a simple Java program which divides two integers.

```
import java.util.Scanner;
class Division {
        public static void main(String[] args) {

        int a, b, result;

        Scanner input = new Scanner(System.in);
        System.out.primln("Input two integers");

        a = input.nextInt();
        b = input.nextInt();

        result = a / b;

        System.out.println("Result = " + result);
                }
        }
```

If we club and run the above code twice, then we have an output as:



```
E:\Java>javac Division.java

E:\Java>java Division
Input two integers
4 2
Result = 2

E:\Java>java Division
Input two integers
7 0
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at Division.main(Division.java:14)

E:\Java>_
```

Check your progress 2

1. Which keyword is not a part of exception handling?

a. try

b. finally

c. thrown

d. catch

3.4 Catch Block and Searches

In Java, it is seen that a try statement carries one catch block. In this there is no code which is written at the end of try block and beginning of first catch block.

```
try {
} catch (ExceptionType name) {
} catch (ExceptionType name) {
}
```

In this we see that every catch block is an exception handler which will be able to handles type of exception as shown in an argument. Here the argument type, Exception Type will declare type of exception which and handler has to handle which should comprise of name of class which inherits it from Throwable class. Here the handler is related to the exception with name.

It is found that a catch block contains code which is carried out at the time when an exception handler calls upon. In such case a runtime system will call upon exception handler when handler is at first in call stack with Exception Type matches with type of exception thrown. In such case the system will consider it as match when thrown object is assigned legally to exception handler's argument.

Consider exception handlers for write List method shown:

```
try {


} catch (IndexOutOfBoundsException e) {
    System.err.println("IndexOutOfBoundsException: " + e.getMessage());
} catch (IOException e) {
    System.err.println("Caught IOException: " + e.getMessage());
}
```

It is found that an exception handler can simply print error messages or make the program to stop. It can do error recovery, ask user to make decision or can spread the error to higher level handler with the help of chained exceptions.

Catching More Exceptions with Single Exception Handler

We see that the two exception handlers that is used by writeList() method is very particular. The two handlers will handle one type of exception. In Java, we can write general exception handlers which can handle multiple types of exceptions. The Java exceptions are Throwable objects. The Java packages have many classes which are obtained from Throwable which builds a hierarchy of Throwable classes. As seen, in the catch clause, particular types of exceptions which a block can handle and separate exception type with a vertical bar (|) is shown:

```
catch (IOException|SQLException ex) {
    logger.log(ex);
    throw ex;
}
```

In this, we see that if a catch block handles more than one exception type, then catch parameter will totally become final. Here, the catch parameter is final and hence no value is assign to it in the catch block.

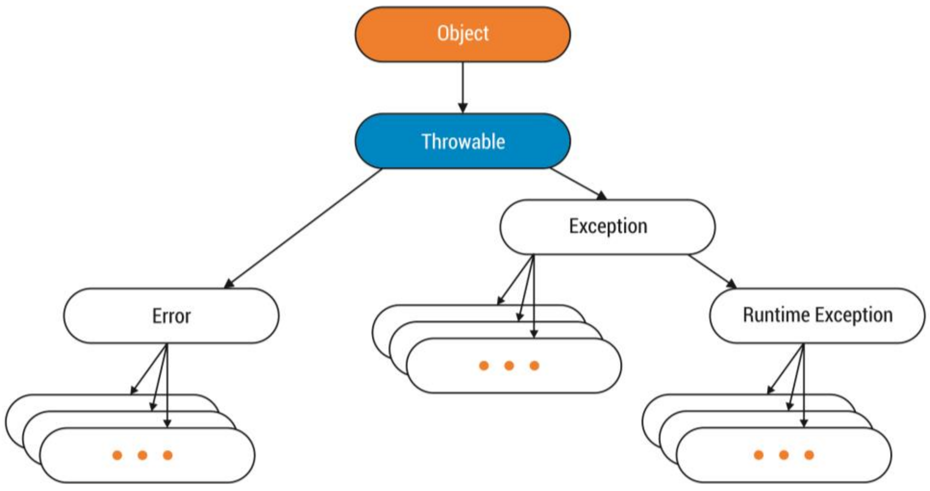


Fig 3.3 Exceptional block

If we alter the writeList() method by writing it so as to handles both IO Exceptions and Array Index Out Of Bounds Exceptions. We see that a close ancestor of IO Exception and Array Index out Of Bounds Exception is Exception class. So an exception handler which will be able to handle both types of exceptions will look like:

```
try {
    . . .
} catch (Exception e) {
    System.err.println("Exception caught: " + e.getMessage());
}
```

It is seen that an exception class in this is very high in Throwable class hierarchy. Hence along with IO Exception and Array Index Out Of Bounds Exception types, such exception handler will about to catch many other types.

Check your progress 3

1. Which among the following keyword is used for block to find for exceptions?

a. try	c. throw
b. catch	d. check

3.5 The Throw Statement

In Java, it is seen that a throw statement is used to throw an exception. It requires a single argument which is a throwable object. The throwable objects are instances of any subclass of Throwable class. In a throw statement shown:

```
throw someThrowableObject;
```

In this, if we try to throw an object which is not throwable, then under such cases a compiler will refuse to accumulate the program and will show error message which will look like:

```
testing.java:10: Cannot throw class java.lang.Integer; it must be a subclass
of class java.lang.Throwable.
                    throw new Integer(4);
```

In reference to throw statement, a method is taken from class which implements a common stack object. Here the pop method will remove top element from stack and returns it

```
public Object pop() throws EmptyStackException {
    Object obj;
    if (size == 0)
        throw new EmptyStackException();
        obj = objectAt(size - 1);
        setObjectAt(size - 1, null);
        size--;
        return obj;
    }
```

We see a pop method will check for any elements on stack. If stack is empty, then pop will instantiates a new Empty Stack Exception object and throws it. Here, an Empty Stack Exception class is defined in java.util package.

The throws Clause

The declaration of pop method has clause like:

```
throws EmptyStackException
```

In this, a throw clause will tell that method can throw an Empty Stack Exception. As known, Java requires particular methods which can either catch or specify all checked exceptions which are thrown in scope of such method.

Check your progress 4

1. Which statement is correct which will describe object thrown by throw statement?

   a. assignable to Throwable type

   b. assignable to Error type

   c. assignable to Exception type

   d. assignable to String type

3.6 Exception and Error Classes

It is seen that in Java, exception classes are organised in hierarchy. There is basic exception class which is known as Exception. So a basic hierarchy begins with Throwable class and not with Exception which is further subclassed as Exception and Error as shown in figure 3.4.

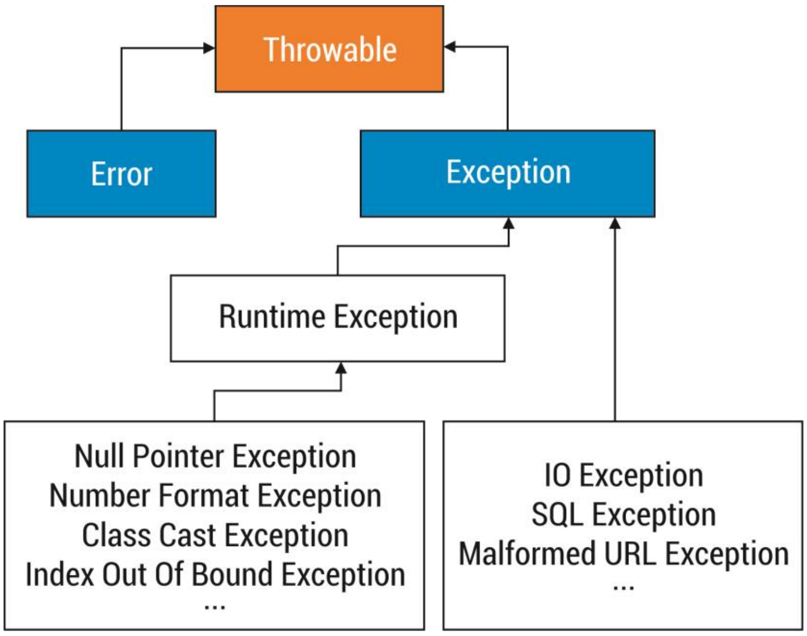


Fig 3.4 Exception hierarchy

In this figure we see that Exception subclasses shows errors which program recovered from. Apart from RuntimeException and its subclasses, they are normally shown errors which is expected by a program during normal working.

The Error subclasses shows serious errors which a program shouldn't expect to catch and recover from. This will cover conditions like expected class file being missing, or an Out Of Memory Error.

Runtime Exception is again subclass of Exception and its subclasses are different since they show exceptions which a program shouldn't expect. They represent what are likely to be programming errors rather than errors due to invalid user input or a badly configured environment.

Check your progress 5

1. Which among the following is a super class of exceptional type classes?

a. String	c. Throwable
b. Runtime Exceptions	d. Cachable

3.7 The Throws Clause

In Java, throws keyword is applied in method declaration to explicit particular exceptions which is thrown by particular method. While declaring a method, one or more exceptions are defined with throws clause. In defining a method, throws clause is included to declare exceptions which are thrown but doesn't get caught in method. Now, if a method with throws clause having few exceptions, then it will tell other methods as if you need to call, then you should know how to handle such exceptions while throwing. The syntax of Throws in java is:

```
void MethodName() throws ExceptionName{
    Statement1
    ...
    ...
}
```

In an example:

```
public void sample() throws IOException{
    //Statements
    //if (somethingWrong)
    IOException e = new IOException();
    throw e;
    //More Statements
}
```

We see if a method throws more exceptions then all of them should be listed in throws clause as shown:

```
public void sample() throws IOException, SQLException
{
    //Statements
}
```

Check your progress 6

1. In ______, throws keyword is applied in method declaration to explicit particular exceptions which is thrown by particular method.

   a. Java c. c++

   b. c d. none of the above

3.8 Custom Exceptions

In Java, you can create your own exceptions. For creating as per your needs, the Exception of such type is known as custom exception or can be a user-defined exception. With such custom exception, you can design and have your own exception and message. Consider an example shown:

```
class InvalidAgeException extends Exception{
 InvalidAgeException(String s){
  super(s);
 }
}

class TestCustomException1{
   static void validate(int age)throws InvalidAgeException{
     if(age<18)
      throw new InvalidAgeException("not valid");
     else
      System.out.println("welcome to vote");
   }
   public static void main(String args[]){
      try{
      validate(13);
      }catch(Exception m){System.out.println("Exception occured: "+m);}
```

```
        System.out.println("rest of the code...");
      }
    }
```

If we run the above program, we have output as:

```
Exception occurred: Invalid Age Exception:not valid
rest of the code...
```

Check your progress 7

1. Which of these classes is used to define exceptions?

a. Exception	c. Abstract
b. Trowable	d. System

3.9 Skills check

In Java, it is seen that there arises exceptions and errors in a program. There are necessary skills used in order to solve certain exceptional problems. You can create custom exceptions in Java which involves great skills to calculate and program.

Check your progress 8

1. In Java, it is seen that there arises exceptions and errors in a program

a. True

b. False

3.10 Exercises

Exercise 1:

Consider an example program with an exception class as:

```
class Exceptions {
 public static void main(String[] args) {

 String languages[] = { "C", "C++", "Java", "Perl", "Python" };

 try {
  for (int c = 1; c <= 5; c++) {
    System.out.println(languages[c]);
  }
 }
 catch (Exception e) {
  System.out.println(e);
 }
 }
}
```

If we compile and run, then we have an output as:

```
C++
Java
Perl
Python
java.lang.ArrayIndexOutOfBoundsException: 5
```

3.11 Let Us Sum Up

In this unit we have learnt that, exceptions are abnormal, unexpected or extraordinary events or conditions which occurs during the runtime which could be file not found exception or unable to get connection. In such conditions, java will try to throw exception object.

It is studied that exception is an event which will disturb normal flow of a program. It is an object which is thrown at runtime. In this, when you throw an exception then an object is thrown

It is found that exception handling is considered as one of the powerful arrangement to handle runtime errors in certain conditions such that normal flow of application can be maintained.

3.12 Answers for Check Your Progress

Check your progress 1

Answers: (1-c)

Check your progress 2

Answers: (1-c)

Check your progress 3

Answers: (1-a)

Check your progress 4

Answers: (1-a)

Check your progress 5

Answers: (1-c)

Check your progress 6

Answers: (1-a)

Check your progress 7

Answers: (1-a)

Check your progress 8

Answers: (1-a)

3.13 Glossary

1. **Checked exception -** It is subclass of Exception excluding class Runtime Exception and its subclasses.

2. **Runtime exceptions -** These are exceptions which are thrown at runtime.

3.14 Assignment

What is the basic difference between the 2 approaches to exception handling?

3.15 Activities

How to create custom exceptions?

3.16 Case Study

Discuss the output of the program.

```
public class Foo
{
   public static void main(String[] args)
   {
      try
      {
         return;

      }
      finally
      {
         System.out.println( "Finally" );
      }
   }
}
```

3.17 Further Readings

1. A Primordial Interface by Wm. Paul Rogers.
2. Java Diamonds Forever by Tony Sintes.
3. Java Routines by John D. Mitchell.
4. Object-Oriented Design and Programming by Tony Sintes.

Block Summary

In this block, students will be detailed with knowledge about Inheritance and its relationship with subclass and interface block. The concepts about types of variables with illustrations are explained in step by step manner. The block stress on information related to packages and interface with examples about inherit abstract methods of interface. The information related to declaring a reference variable of interface is well explained with examples.

After studying this block, students will feel self-confident while working on simple Java platform and can enhance their knowledge by studying certain examples and illustrations mentioned in this block. With such detailed knowledge on Inheritance, Interface, Packages and Exceptions in Java, student can be benefitted in future.

Block Assignment

Short Answer Questions

1. What is Subclass in Java?
2. What is a package in Java?
3. Define instance of operator?
4. What if the main method is declared as private?
5. What are runtime exceptions?

Long Answer Questions

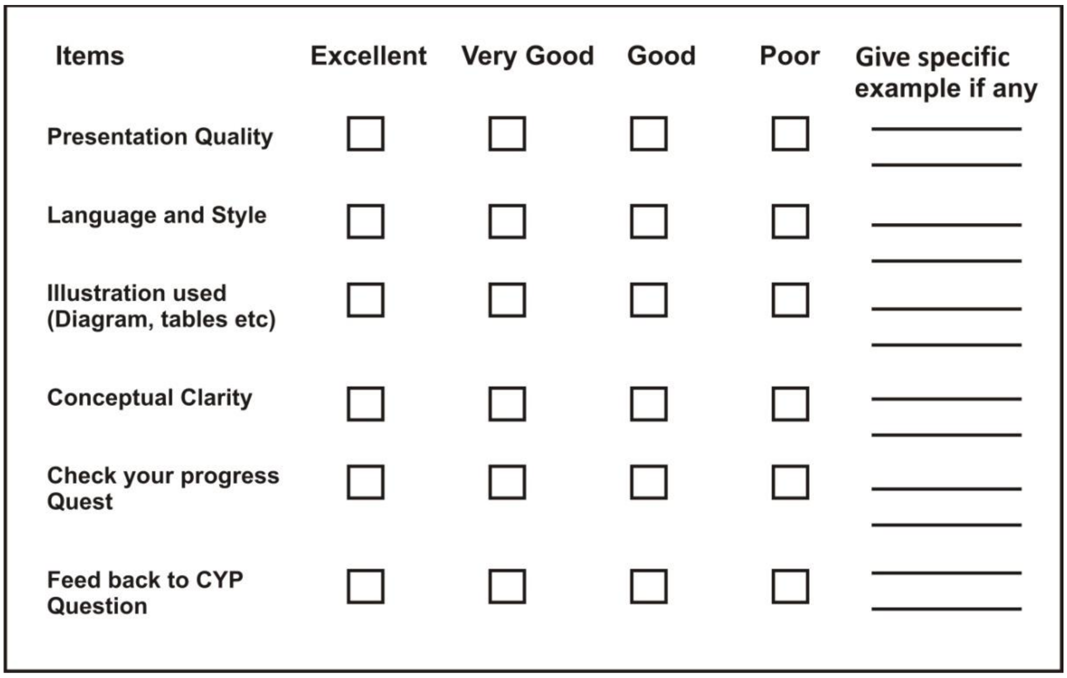
1. What are the different ways to handle exceptions?
2. Explain types of variables in Java?
3. Explain about throw statement and clauses?

Enrolment No.

1. How many hours did you need for studying the units?

Unit No	1	2	3	4
Nos of Hrs				

2. Please give your reactions to the following items based on your reading of the block:



Items	Excellent	Very Good	Good	Poor	Give specific example if any
Presentation Quality	☐	☐	☐	☐	
Language and Style	☐	☐	☐	☐	
Illustration used (Diagram, tables etc)	☐	☐	☐	☐	
Conceptual Clarity	☐	☐	☐	☐	
Check your progress Quest	☐	☐	☐	☐	
Feed back to CYP Question	☐	☐	☐	☐	

3. Any Other Comments

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

Education is something which ought to be brought within the reach of every one.

- Dr. B. R. Ambedkar

Dr. Babasaheb Ambedkar Open University

Jyotirmay' Parisar, Opp. Shri Balaji Temple, Sarkhej-Gandhinagar Highway, Chharodi, Ahmedabad-382 481.

OBJECT ORIENTED CONCEPTS

PGDCA-102

BLOCK 4:

JAVA CLASS LIBRARY, FILE HANDLING AND GUI

Dr. Babasaheb Ambedkar Open University
Ahmedabad

OBJECT ORIENTED CONCEPTS

Knowledge Management and
Research Organization
Pune

Editorial Panel

Author
Er. Nishit Mathur

Language Editor
Prof. Jaipal Gaikwad

Graphic and Creative Panel
Ms. K. Jamdal
Ms. Lata Dawange
Ms. Pinaz Driver
Ms. Tejashree Bhosale
Mr. Kiran Shinde
Mr. Akshay Mirajkar

Acknowledgment
Every attempt has been made to trace the copyright holders of material reproduced in this book. Should an infringement have occurred, we apologize for the same and will be pleased to make necessary correction/amendment in future edition of this book.
The content is developed by taking reference of online and print publications that are mentioned in Bibliography. The content developed represents the breadth of research excellence in this multidisciplinary academic field. Some of the information, illustrations and examples are taken "as is" and as available in the references mentioned in Bibliography for academic purpose and better understanding by learner.'

ROLE OF SELF INSTRUCTIONAL MATERIAL IN DISTANCE LEARNING

The need to plan effective instruction is imperative for a successful distance teaching repertoire. This is due to the fact that the instructional designer, the tutor, the author (s) and the student are often separated by distance and may never meet in person. This is an increasingly common scenario in distance education instruction. As much as possible, teaching by distance should stimulate the student's intellectual involvement and contain all the necessary learning instructional activities that are capable of guiding the student through the course objectives. Therefore, the course / self-instructional material are completely equipped with everything that the syllabus prescribes.

To ensure effective instruction, a number of instructional design ideas are used and these help students to acquire knowledge, intellectual skills, motor skills and necessary attitudinal changes. In this respect, students' assessment and course evaluation are incorporated in the text.

The nature of instructional activities used in distance education self-instructional materials depends on the domain of learning that they reinforce in the text, that is, the cognitive, psychomotor and affective. These are further interpreted in the acquisition of knowledge, intellectual skills and motor skills. Students may be encouraged to gain, apply and communicate (orally or in writing) the knowledge acquired. Intellectual-skills objectives may be met by designing instructions that make use of students' prior knowledge and experiences in the discourse as the foundation on which newly acquired knowledge is built.

The provision of exercises in the form of assignments, projects and tutorial feedback is necessary. Instructional activities that teach motor skills need to be graphically demonstrated and the correct practices provided during tutorials. Instructional activities for inculcating change in attitude and behavior should create interest and demonstrate need and benefits gained by adopting the required change. Information on the adoption and procedures for practice of new attitudes may then be introduced.

Teaching and learning at a distance eliminates interactive communication cues, such as pauses, intonation and gestures, associated with the face-to-face method of teaching. This is particularly so with the exclusive use of print media. Instructional activities built into the instructional repertoire provide this missing interaction between the student and the teacher. Therefore, the use of instructional activities to affect better distance teaching is not optional, but mandatory.

Our team of successful writers and authors has tried to reduce this.

Divide and to bring this Self Instructional Material as the best teaching and communication tool. Instructional activities are varied in order to assess the different facets of the domains of learning.

Distance education teaching repertoire involves extensive use of self-instructional materials, be they print or otherwise. These materials are designed to achieve certain pre-determined learning outcomes, namely goals and objectives that are contained in an instructional plan. Since the teaching process is affected over a distance, there is need to ensure that students actively participate in their learning by performing specific tasks that help them to understand the relevant concepts. Therefore, a set of exercises is built into the teaching repertoire in order to link what students and tutors do in the framework of the course outline. These could be in the form of students' assignments, a research project or a science practical exercise. Examples of instructional activities in distance education are too numerous to list. Instructional activities, when used in this context, help to motivate students, guide and measure students' performance (continuous assessment)

PREFACE

We have put in lots of hard work to make this book as user-friendly as possible, but we have not sacrificed quality. Experts were involved in preparing the materials. However, concepts are explained in easy language for you. We have included may tables and examples for easy understanding.

We sincerely hope this book will help you in every way you expect.

All the best for your studies from our team!

Contents

BLOCK 1: INTRODUCTION TO PROGRAMMING WITH JAVA

UNIT 1 PROBLEM SOLVING WITH COMPUTERS

Need of programming, Algorithms, Flowcharts, Pseudo-Code, Programming Languages Overview, and Overview of Object Oriented Concepts, Skills check, Exercises

UNIT 2 INTRODUCTION TO JAVA

History, Overview of JVM & Bytecode, Java versions (J2me etc.), Java class library, JDK, Your First Java Application, Skills check, Exercises

UNIT 2 BEGINNING WITH JAVA PROGRAMMING

Java Keywords, Comments in Java, Variables and Assignments, Strings and Characters, Arithmetic Operators and Expressions, Type Conversion in Expressions, Type Conversion, Skills check, Exercises

BLOCK 2: JAVA CONTROL STATEMENTS, OPERATORS AND CLASSES

UNIT 1 JAVA CONTROL STATEMENTS

The if statement, The if-else statement, Blocks of Code, The for statement, Increment & Decrement operators, Backslash codes, Relational and Boolean logical operators, Ternary operators, Skills check, Exercises

UNIT 2 MORE ABOUT CONTROL STATEMENTS AND OPERATORS

Nested if statement, Variations of the for loop, The while loop, The do loop, Nested loops, The break statement, The continue statement, The switch statement, The bitwise operators, Skills check, Exercises

UNIT 3 **CREATING CLASSES**

The General form of a class, Creating Simple Classes, Adding Constructors, Constructor Overloading, The this keyword, Instance variables and methods, Static variables and methods, Local variables and its scope, Method overloading, Argument Passing, Wrapper Classes, System Class, Garbage Collection, Skills check, Exercises

BLOCK 3: INHERITANCE, INTERFACE, PACKAGES AND EXCEPTIONS IN JAVA

UNIT 1 **INHERITANCE**

Overview of Re-usability, Subclasses, Inheritance and Variables, Method Overriding, Inheritance and Methods, Inheritance and Constructors, Class Modifiers, Variable Modifiers, Constructor Modifiers, Method Modifiers, Object and Class classes, Skills Check, Exercises

UNIT 2 **INTERFACES AND PACKAGES**

Interfaces, Interface References, Interface Inheritance, The instanceof Operator, Packages, The import statement, Access control and Packages, Skills Check, Exercises

UNIT 3 **EXCEPTIONS**

Overview of Exceptions, Exception Handling, Catch Block and searches, The throw statement, Exception and Error classes, The throws clause, Custom exceptions, Skills check, Exercises

BLOCK 4: JAVA CLASS LIBRARY, FILE HANDLING AND GUI

UNIT 1 **INTRODUCTION TO JAVA CLASS LIBRARY**

Classes of java.util package, Classes of java.net package, Classes of java.lang package, Overview of Collection Framework, Skills check, Exercises

UNIT 2	**FILE HANDLING IN JAVA**
	Overview of File Handling, File Class in Java, Using Character Stream Classes, Using Byte Stream Classes, Using Scanner and Console Classes in Java, Skills check, Exercises
UNIT 3	**BUILDING GRAPHICAL USER INTERFACE (SWING)**
	Building applications using classes related to Graphical Interface, Creating Layouts in GUI, Using Events in GUI applications, Skills check, Exercises

Dr. Babasaheb
Ambedkar
Open University

PGDCA-102

OBJECT ORIENTED CONCEPTS

BLOCK 4: JAVA CLASS LIBRARY, FILE HANDLING AND GUI

UNIT 1
INTRODUCTION TO JAVA CLASS LIBRARY 03

UNIT 2
FILE HANDLING IN JAVA 20

UNIT 3
BUILDING GRAPHICAL USER INTERFACE (SWING) 41

BLOCK 4: JAVA CLASS LIBRARY, FILE HANDLING AND GUI

Block Introduction

GUI is a Graphical User Interface that works with icons or indicators along with electronic devices. Interface is a coordination involved among computer, program and humans. It is seen that graphical user interface uses visual elements that will help in showing information which is kept inside computer that is simple to understand. These elements makes easy for people to work by involving certain computer software. It is found that in order to design a good user interface it should allow easy and natural interaction among user and system. File is a mixture of bytes which are stored in secondary storage device such as disk. So we find that file handling is applied in order to read, write, append or update a file without directly opening it.

In this block students will be get knowledge on Java Library Classes with stress on files handling and its usages. The input and output operation of file along with process of data entered is well explained. The student will be given information on Byte oriented stream and Character oriented stream with their reading writing mechanism.

After this block, student will get knowledge on Scanner and Console classes with usage on java.util and basic understanding of wrapper class with input stream stdin. The knowledge about utilities in java.util along with pogramming structures allow students to work on professional written classes. The idea about Java.util package will be beneficial to them in longer use.

Block Objective

After learning this block, you will be able to understand:

- Graphical User Interface
- Concept of Interfacing
- File handling system
- Byte oriented and Character oriented streams
- Scanner and Console classes
- Java.util package

- Basic of java.lang package

Block Structure

Unit 1: Introduction to Java Class library

Unit 2: File Handling in Java

Unit 3: Building Graphical User Interface (Swing)

UNIT 1: INTRODUCTION TO JAVA CLASS LIBRARY

Unit Structure

1.0 Learning Objectives

1.1 Introduction

1.2 Classes of Java.util Package

1.3 Classes of Java.net Package

1.4 Classes of Java.lang Package

1.5 Overview of Collection Framework

1.6 Skills Check

1.7 Exercises

1.8 Let Us Sum Up

1.9 Answers for Check Your Progress

1.10 Glossary

1.11 Assignment

1.12 Activities

1.13 Case Study

1.14 Further Readings

1.0 Learning Objectives

After learning this unit, you will be able to understand:

- The Classes of java.util package
- The Classes of java.net package
- The Classes of java.lang package

1.1 Introduction

Java is an object oriented programming language, so in java programming, main two features are supported that is class and object. Basically we know that class is collection of objects and object is an instance of class. The uses of the java libraries of class spread the programmer efficiency by allowing computer programmer to focus on the functionality unique to their job. The library classes are generally planned with some typical usage pattern in observance, and the performance may be suboptimal if the actual usage differs. We deliver an approach for rewriting applications to use different customized versions of library classes that are generated using a combination of static analysis and profile information.

The java programming supports different type of classes as User defines class and also there are some classes available with java system that provide some important support to the java programmer for developing their programming logic as well as their programming architecture with very smooth and very fine way. These classes are called Library Classes. In Java support thousands of library classes and also each class contains various types of functions. The availability of a large numbers of libraries of standardized classes is a vital reason for popularity of Java as a standard programming language in modern software world. The use of class libraries grows for the programmer productivity by allowing programmers to focus on the characteristics for that are unique to their application without being burdened with the unexciting task of building and debugging of the standard infrastructure. This java library classes are often designed and executed with some typical usage pattern in real life job profile.

1.2 Classes of Java.util package

One of the fundamental packages which uses simplest of Java applications is Java utility package java.util. Java.util package exists since beginning of Java which was one of the original packages available in Java.

Utilities in java.util will show complex programming structures that are used to use in certain professional written classes. The majority of utility classes are related with collections of such package which will include classes for date and time, resource bundle handling, parsing a string and Timer API.

Java.util package contains the collections framework, legacy collection classes, event model, date and time facilities, internationalization, and miscellaneous utility classes. This reference will take you through simple and practical methods available in java.util package.

It is found that a number of classes in java.util are framed which will manage a collection of objects. It is seen that a Vector class supports variable-length arrays of objects, while Hashtable class creates hashtables or associative arrays which carriers key/value pairs of objects.

The java.util package contains a number of new classes in Java 1.1 to support internationalization. Many of these classes work in conjuction with the classes defined in the new java.text package. The most important new class is the Locale class, which represents a particular locale, or country and language, for internationalization purposes.

Interface Summary

Collection	The root interface in the collection hierarchy.
Comparator	Comparison fimction that imposes total ordering on collection of objects.
Enumeration	Object that implements Enumeration interface that generates series of elements at a time.
EventListener	A tagging interface that all event listener interfaces must extend.
Iterator	An iterator over a collection.
List	An ordered collection.
Listlterator	Iterator for lists which allows programmer to traverse list in either direction and modify list during iteration.
Map	An object that maps keys to values.
Map.Entry	A map entry.
Observer	Class implementation that observer interface when informed of changes in observable objects.
Set	A collection that contains no duplicate elements.
Sorter Map	Map that guarantees in ascending key order, sorted as natural ordering of its keys.
SortedSet	Set that guarantees its iterator to traverse set in ascending element order and sorted as per natural ordering of elements.

Check your progress 1

1. Which of following serves as collection classes?

a. Collection	c. HashSet
b. Iterator	d. None of above

1.3 Classes of Java.net Package

The java.net package provides easy way to access application layer, transport layer, and internet layer.

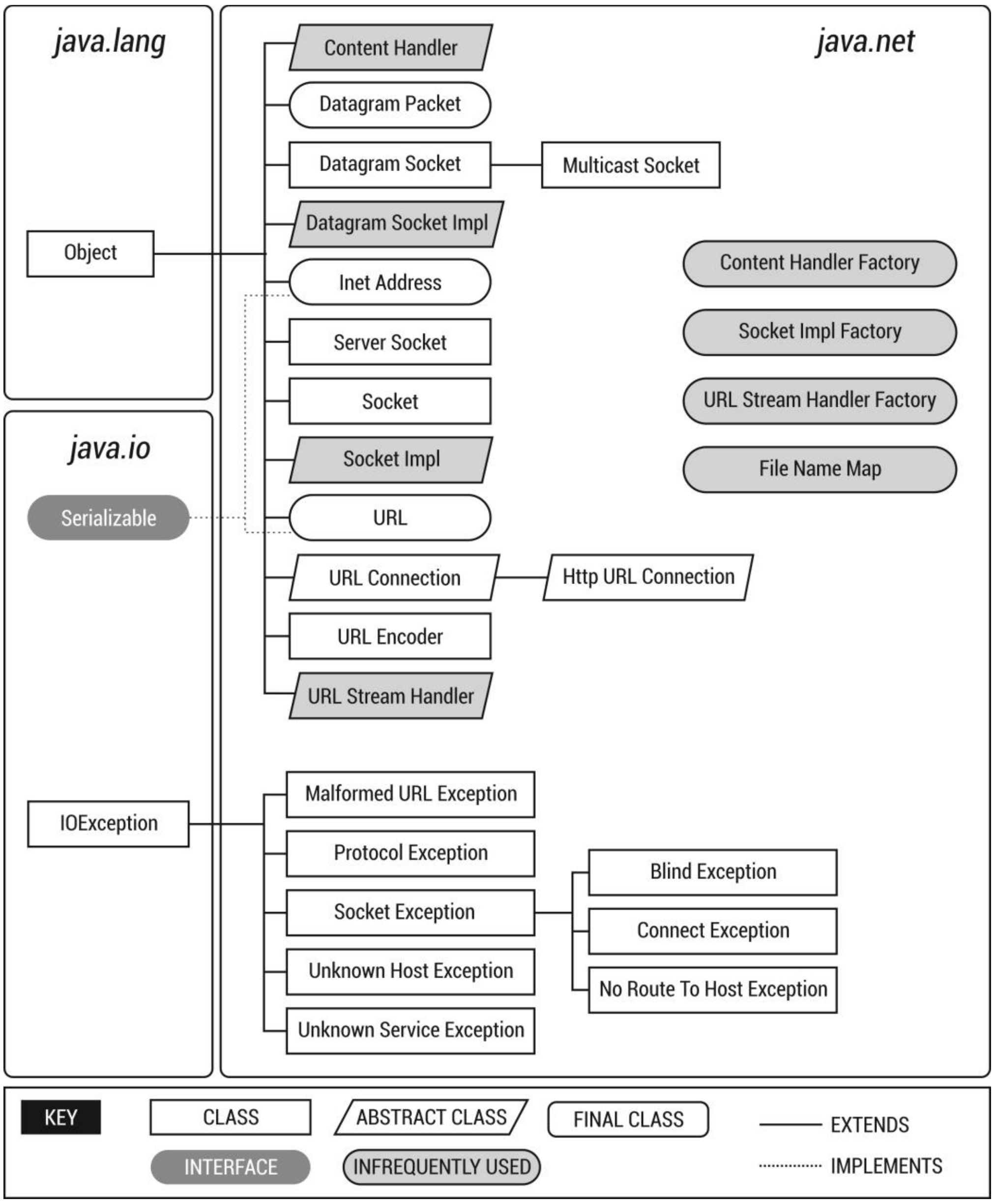


Fig 1.1 java.net package

It carries most relevant classes and methods:

URL

- URL(String spec) - constructor
- URLConnection openConnectiono - creates TCP connection of type given by the protocol

URL Connection (interface) / HttpURLConnection (subclass) - for making HTTP requests

- void setRequestMethod(String method) - set method (GET/POST/... -default is GET)
- void setDoOutput(boolean dooutput) - intend to use connection for output
- void setDoInput(boolean doinput) - intend to use connection for input
- Output Stream getOutput Stream() - output stream
- InputStream getInputStream0 - input stream
- void setRequestProperty(String key: String value) - set request header
- Hashtable getHeaderFieldsO - read response header
- Object getContento - read input and convert to an object

URLEncoder / URLDecoder - for encoding/decoding special chars in URLs (application/x-www-form-urlencoded)

- String encode(String s)
- String decode(String s)

Consider an example:

```
import java.net.*;

import java.io.*;

public class AltaVista {

        public static void main (String args[])

        {

try {
```

```
// make connection
URL url = new URL(http://www.altavista.com/cgi-bin/query?q=" +
                    URLEncoder.encode(args[0]));
URLConnection connection = url.openConnection();
connection.setDoInput(true);
InputStream in = connection.getInputStream();
// read reply
StringBuffer b = new StringBuffer();
BufferedReader r = new BufferedReader(new InputStreamReader(in));
String line;
while ((line = r.readLine()).!= null)
      b.append(line);
      String s = b.toString();
      // look for first search result, if any
      if (s.indexOf(">We found 0 results").!= -1)
      System.out.println("No results found.");
      else {
      int i = s.indexOf("\"status=”“)+9;
      int j = s.indexOf("”, i);
      System.out.println("First result: " + s.substring(i, j));
                  }
            }
      catch (Exception e) { e.printStackTrace(); }
            }
      }
```

Apart from this, there are several useful classes in java.net such as:

InetAddress - IP addresses (DNS lookup, etc.)

Socket,ServerSocket - TCP sockets (as in example client and server)

ProtocolHandler - defines mapping for URL.openConnection() to concrete URLConnection

ContentHandler - defines mapping for getObject() to Object (based on MIME type)

Check your progress 2

1. Which among the following package has classes and interfaces for networking?

a. java.io	c. java.net
b. java.util	d. java.lang

1.4 Classes of Java.lang package

The java.lang.Package class contain version information about the implementation and specification of a Java package. The java.lang package carries class for each Java primitive type such as Boolean, Byte, Short, Character, Integer, Float, Long, Double, Void.

Annotations

Deprecated	Marks program elements which are not used by programmers.
Override	Mark methods that override method declaration in superclass.
SafeVarargs	Claims to compiler about annotation target without argument.
SuppressWarnings	Shows compiler about warnings for program element.

Interfaces

Appendable	Declares methods to append characters or character sequences.
AutoCloseable	Defines interface for classes need to close once.
CharSequence	Shows ordered set of characters and methods to probe them.
Cloneable	Interface implements by classes wishes to support cloning.
Comparable	Implements classes wish to define natural order of their instances.
Iterable<T>	Implement interface used with enhanced for loop.
Readable	Shows sequence of characters incrementally read into CharBuffer.
Runnable	Shows command that can be executed.

Classes

Boolean	The wrapper for the primitive type boolean.
Byte	The wrapper for the primitive type byte.
Character	The wrapper for the primitive type char.
Charac.UnicodeBlock	Represents a block of Unicode characters.
Class<T>	The in-memory representation of a Java class.
ClassLoader	Loads classes and resources from a repository.
Compiler	Does nothing on Android.
Double	The wrapper for the primitive type double.
Float	The wrapper for the primitive type float.
Integer	The wrapper for the primitive type int.
Long	The wrapper for the primitive type long.
Math	Shows math constants and operations.
Number	Abstract superclass shows numeric base types.
Object	The root class of the Java class hierarchy.
Package	Contains information about a Java package.
Process	Represents an external process.
ProcessBuilder	Creates operating system processes.
Runtime	Allows Java applications to interface with environment.
RuntimePermission	Legacy security code; do not use.
SecurityManager	Legacy security code; do not use.
Short	The wrapper for the primitive type short.
StackTraceElement	A representation of a single stack frame.

Exceptions

ArithmeticException

ArrayIndexOutOfBoundsException

ArrayStoreException

ClassCastException

ClassNotFoundException

CloneNotSupportedException

IllegalArgumentException

IllegalMonitorStateException

IllegalStateException

IllegalThreadStateException

IndexOutOfBoundsException

InstantiationException

InterruptedException

NegativeArraySizeException

NoSuchFieldException

NoSuchMethodException

NullPointerException

NumberFormatException

RuntimeException

SecurityException

StringIndexOutOfBoundsException

Check your progress 3

1. The wrapper classes are found in ______ package

a. java.lang	c. java.io
b. java.util	d. java.net

1.5 Overview of Collection Framework

It is found that collections in java carries a framework which shows architecture that will store and carry out group of objects. The operations done on data like searching, sorting, insertion, manipulation, deletion can be done with the help of Java Collections.

It is applicable for single unit of objects. Such framework will provide:

Interfaces:

- Set
- List
- Queue
- Deque

Classes:

- ArrayList
- Vector
- LinkedList
- PriorityQueue
- HashSet
- LinkedHashSet
- TreeSet

The idea of collections framework is to:

- Create high performance framework to implement highly efficient collections of dynamic arrays, linked lists, trees and hashtables.
- Make framework that will allow several types of collections that will work in similar manner with high degree of interoperability.
- Ease the extend and/or adapting collection.

The Collections Framework hierarchy comprises of:

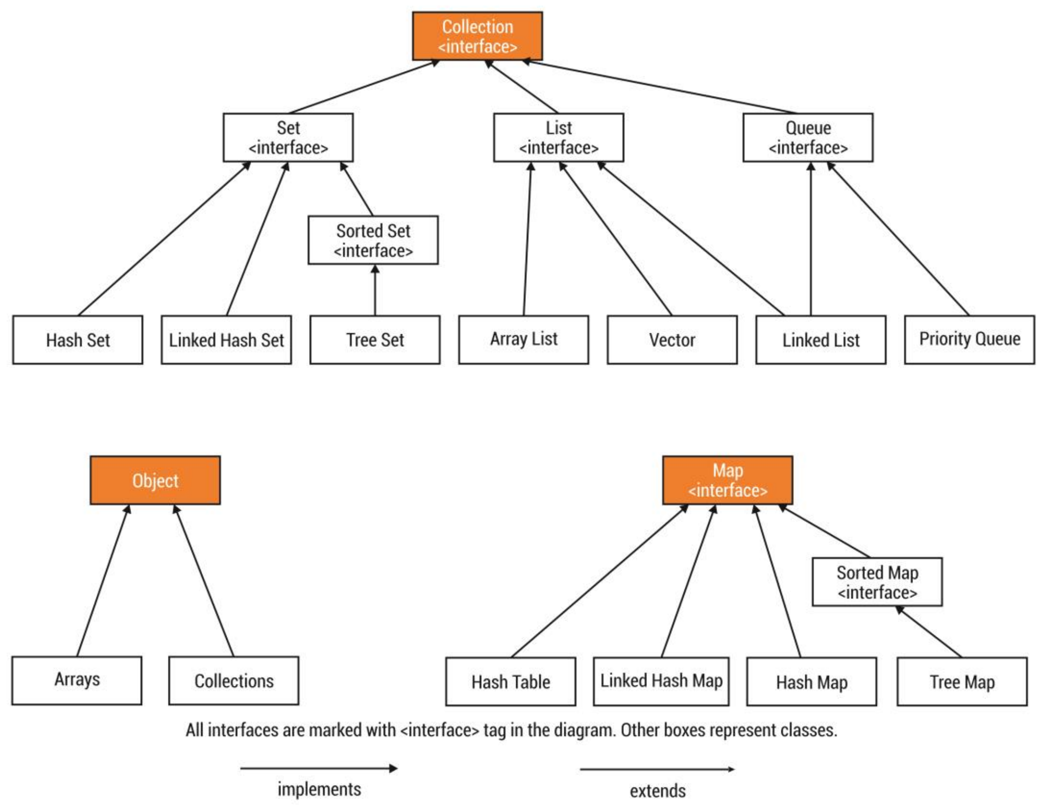


Fig 1.2 Collections Framework

List

It is an ordered Collection or sequence which carries duplicate elements that can put or accessed by their position in list having zero-based index. It contains:

- ArrayList
- LinkedList
- Vector

Set

It is a Collection or sequence of non-duplicate elements that has three main implementations of Set interface which are:

- HashSet
- TreeSet
- LinkedHashSet

HashSet: It stores elements in hash table which is best performing implementation having no guarantee in relation to iteration.

TreeSet: It stores elements in red-black tree which orders elements as per their values. It is slower than HashSet.

LinkedHashSet: It is implemented as hash table having a linked list which runs through it.

Map

Map is an object which maps keys to particular values. It has no duplicate keys. There are three important Map implementation interfaces as:

- HashMap
- TreeMap
- LinkedHashMap

HashMap: It has no guarantee related to order of iteration

TreeMap: It stores elements in red black tree which is based on values and is slower than HashMap.

LinkedHashMap: It will order elements as per order in which they were inserted into set.

Iterator/ListIterator

It is seen that both Iterator and ListIterator are applied to iterate with the help of elements of collection class. With the help of Iterator, we move in forward direction with the help of ListIterator.

- Iterator
- ListIterator

Check your progress 4

1. Which among the following stores elements in insertion order?

a. TreeMap	c. LinkedMap
b. HashMap	d. LinkedHashMap

1.6 Skills Check

In Java, skills are required to study about Hash, Tree and LinkedHash maps that stores elements. The detail study of list and set allow us to bifurcate with related skills that can be used to study them.

Check your progress 5

1. In Java, skills are required to study about Hash, Tree and LinkedHash maps that stores elements

   a. True

   b. False

1.7 Exercises

Example 1:

Let us develop a LinkedHashMap with the help of following code:

```
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.Map;
import java.util.Set;
public class LinkedHashMapExample {
  public static void main(String[] args) {
  LinkedHashMap<String, String> hMap = new
                                   LinkedHashMap<String, String>();
hMap.put("A1", "Nishit");
hMap.put("A2", "Rohit");
hMap.put("A3", "Peeyush");
hMap.put("A4", "Deepak");
hMap.put("A5", "Ashok");
Set set = hMap.entrySet();
Iterator itr = set.iterator();
while (itr.hasNext()) {
  Map.Entry m = (Map.Entry) itr.next();
  System.out.println(m.getKey() + " " + m.getValue());
}
}
}
```

On running the above program, we get an output as:

```
A1 Nishit
A2 Rohit
A3 Peeyush
A4 Deepak
A5 Ashok
```

1.8 Let Us Sum Up

In this unit we have learnt that Java is an object oriented programming language which supports two features class and object. In this, class is collection of objects and object is an instance of class.

In Java, utilities such as java.util will show complex programming structures which can be used further in many professional written classes. This package contains collection of framework, legacy collection classes, event model, date and time facilities.

It is studied that java.net package provides easy way to access application layer, transport layer, and internet layer. The java.lang.Package class contain version information about the implementation and specification of a Java package.

1.9 Answers for Check Your Progress

Check your progress 1

Answers: (1-c)

Check your progress 2

Answers: (1-c)

Check your progress 3

Answers: (1-a)

Check your progress 4

Answers: (1-d)

Check your progress 5

Answers: (1-a)

1.10 Glossary

1. **Package -** It is a collection of Classes and Interfaces.
2. **List -** It is a collection or sequence having duplicate elements that access by position in list with zero index.
3. **Map -** In programming, map shows an object that describes about particular values.

1.11 Assignment

Discuss Java Library classes in Detail.

1.12 Activities

Is the output of this program true? Discuss.

```
import java.net.*;

class networking {

   public static void main(String[] args) throws UnknownHostException {

      InetAddress obj1 = InetAddress.getByName("cisco.com");

      InetAddress obj2 = InetAddress.getByName("sanfoundary.com");

      boolean x = obj1.equals(obj2);
      System.out.print(x);

   }

}
```

1.13 Case Study

Discuss the output of the program?

```
class Dog {

         String colour;

 Dog(String c) {

          colour = c;

 }

 public String toString(){

          return colour + " dog";

          }

}

public class TestHashMap {

         public static void main(String[] args) {

                  HashMap<Dog, Integer> hashMap = new HashMap<Dog,
Integer>();
```

```
                    Dog d1 = new Dog("red");
                    Dog d2 = new Dog("black");
                    Dog d3 = new Dog("white");
                    Dog d4 = new Dog("white");
                    hashMap.put(d1, 10);
                    hashMap.put(d2, 15);
                    hashMap.put(d3, 5);
                    hashMap.put(d4, 20);
                    //print size
                     System.out.println(hashMap.size());
                     //loop HashMap
                     for (Entry<Dog, Integer> entry : hashMap.entrySet()) {
                              System.out.println(entry.getKey().toString() + " - " +
  entry.getValue());
                     }
           }
}
```

1.14 Further Readings

1. The online Java tutorial @ http://docs.oracle.com/javase/tutorial/.
2. Paul Deitel and Harvey Deitel, "Java How to Program", 9th ed, 2011.
3. Y. Daniel Liang, "Introduction to Java Programming", 9th ed, 2012.
4. Bruce Eckel, "Thinking in Java", 4th ed, 2007.

UNIT 2: FILE HANDLING IN JAVA

Unit Structure

2.0 Learning Objectives

2.1 Introduction

2.2 Overview of File Handling

2.3 File Class in Java

2.4 Using Character Stream Classes

2.5 Using Byte Stream Classes

2.6 Using Scanner and Console Classes in Java

2.7 Skills Check

2.8 Exercises

2.9 Let Us Sum Up

2.10 Answers for Check Your Progress

2.11 Glossary

2.12 Assignment

2.13 Activities

2.14 Case Study

2.15 Further Readings

2.0 Learning Objectives

After learning this unit, you will be able to understand:

- The File Class in Java
- The Character Stream Classes
- The Byte Stream Classes

2.1 Introduction

File is a mixture of bytes which are stored in secondary storage device such as disk. So we find that file handling is applied in order to read, write, append or update a file without directly opening it. It is seen that files in Java can be:

- Text File
- Binary File

As studied, Text File carries information related to only textual data. While working, you can save the Text files in either a plain text (.TXT) format or rich text (.RTF) format as like files in Notepad. Binary Files carries both textual data and custom binary data such as font size, text color and text style.

2.2 Overview of File Handling

Working with files uses input and output operation that we have done through the use of screen and keyboard. Once the program finished executing, then all entered data gets lost as primary memory is volatile. If you want to use the data for later use then it is required to keep it as permanent storage device. It is found that Java language will provide concept of file with which data can be stored on disk or secondary storage device. It is found that the data stored can be read or of any use whenever it is required.

We see in case of files that are handled in java requires to import package having the name as java.io that carries all required classes which is required to do input and output (I/O) operations. So it seen that file handling in java can be done with the help of two streams:

- byte oriented
- character oriented

Byte oriented stream: In case of byte oriented stream, the data is written and read from file using 8-bit bytes. As seen, some common class to do oriented streaming of data from file uses:

- ByteArrayInputStream
- FileInputStream
- FileOutput Stream
- PrintStream

Such type of stream is good for binary data like java .class file where there are images and machine data.

Character oriented stream: In this case, data is accessed each character per time. It is used to do character oriented streaming of data that uses files like:

- FileReader
- FileWriter
- InputStreamReader
- PrintWriter

Such stream is good for text source like java source file, text document etc.

Consider an example:

```
import java.io.PrintWriter;
import  java.io.StringWriter;
public static String log(Throwable t)
   {
      StringWriter sw = new StringWriter();
      PrintWriter pw = new PrintWriter(sw,  true);
      exp.printStackTrace(pw);
      pw.flush();
      sw.flush();
      writeLog("exception"+ sw.toString());
   }
public void writeLog(String msg)
   {
      try
         {
            BufferedWriter out = new
            BufferedWriter(new  FileWriter("D:\\log\\stack.log",true));
            out.write(msg+"n");
            out.close();
         }
      catch (IOException e)
         {
            e.printStackTrace();
         }
   }
```

Check your progress 1

1. Which among the following is a channel by which data flow from source to destination?

a. String	c. Stream
b. Character	d. Buffer

2. In Java, Stream classes are categorised into _______ groups:

a. 1	c. 3
b. 2	d. 4

2.3 File Class in Java

In Java, files and directories are carried out and calculated by File class. The File class will not really show for input and output to files. It shows an identifier of files and directories. As file object are created, it does not show that really they present on disk file with identifier which carries file object. To defining file in File Class, you need several types of constructors.

File Class constructors:

Constructor	Description
File(File parent, String child)	It creates new File from parent abstract as pathname and child pathname string.
File(String pathname)	It creates new File by altering required pathname string into abstract pathname
File(String parent: String child)	It creates new File from parent and child pathname strings.
File(String parent, String child)	It creates new File by converting given file URI into abstract pathname.

File class methods:

Method	Description
Boolean canExecute()	It test for execution of file shown by abstract pathname.
Boolean canRead()	It tests for reading of file shown by abstract pathname.

Boolean canWrite()	It tests for alteration of file shown by abstract pathname.
int compareTo (File pathname)	It compares two abstract pathnames.
Boolean createNewFile()	It creates new from abstract pathname when file with such name does not exist yet.
Boolean delete()	It deletes file or directory shown by abstract pathname.
Long getFreeSpacet()	It returns number of unallocated bytes in partition named by abstract path name.
String getName()	It returns name of file or directory shown by abstract pathname.
String getParent()	It returns pathname string of abstract pathname's parent, if pathname have no parent directory.
File getParentFileQ	It returns abstract pathname of abstract pathname's parent. if pathname have no parent directory.
String getPath()	It converts abstract pathname into pathname string.
String toStringQ	It returns pathname string of abstract pathname.
Boolean mkdir()	It creates directory name by abstract pathname.
Long getTotalSpace()	It returns size of partition name by abstract pathname.
Long getUsableSpace()	It returns number of bytes present to virtual machine on partition name by abstract pathname.
int hashCodeQ	It finds a hash code for abstract pathname.
Boolean isAbsoluteO	It tests for this abstract pathname absoluteness.
Boolean isDirectory()	It tests for directory containing a file from abstract pathname.

Boolean isFile()	It tests for normality of file as file shown by this abstract pathname.
Boolean isHidden()	It tests if file named by this abstract pathname is hidden file.
long lastModifiedO	It returns time that file shows by this abstract pathname that was last changed.
Long length()	It returns length of file shown by this abstract pathname
String[] list()	It returns array of strings naming files and directories in directory shown by this abstract pathname

Check your progress 2

1. Which among the following method will be helpful to read from file?

a. get()	c. scan()
b. read()	d. readFileInput()

2.4 Using Character Stream Classes

In Java, character streams classes are like byte streams as they have 16-bit Unicode characters instead of 8 bit bytes. Such type of stream classes is implemented by Reader and Writer classes along with their subclasses. It is seen that Readers and Writers classes support similar operations like Input Streams and Output Streams rather than byte-stream methods that works on bytes or byte arrays. The character stream methods works with characters and character arrays or strings.

The main benefit of character streams is that they make it simple to write a program which does not depend on particular character encoding. It is found that Java keeps strings in Unicode, which serves as an international standard character encoding which is capable of showing written languages. Normally it is seen that user-readable text files basically uses encodings which does not relates to Unicode or ASCII. It is sometimes seen that character streams hides dealing with such complex encodings that shows two classes which acts as bridges between byte streams and character streams. It is seen that an InputStreamReader class will work as character-input stream which reads bytes from byte-input stream and converts it into characters as per specific encodings. Also, Output StreamWriter class will carry out character output stream which alters characters into bytes as per particular encoding and then writes to byte-output stream.

Another benefit of character streams is that it potentially becomes efficient as compared to byte streams. The working of many Java original byte streams is oriented around byte-at-a-time read and writes operations. Character-stream classes are oriented around buffer-at-a-time read and write operations. With such type of difference in combination with efficient locking scheme makes the character stream classes to add overhead of encoding conversion that exists in several cases.

Character-stream class	**Description**	**Corresponding byte class**
Reader	Abstract class/character-input streams	Input Stream
BufferedReader	Buffers input, parses lines	BufferedlnputStream
LineNumberReader	Keeps track of line numbers	LineNumberInputStream
CharArrayReader	Reads from a character array	ByteArrayInputStream
InputStreamReader	Translates byte stream into character stream	(none)
FileReader	Translates bytes from file into character stream	FileinputStream
FilterReader	Abstract class for filtered character input	Filter Input Stream
PushbackReader	Allows characters to be pushed back	PushbacklnputStream
PipedReader	Reads from a PipedWriter	PipedlnputStream
StringReader	Reads from a String	StringBufferInputStream
Writer	Abstract class for character-output streams	Output Stream
BufferedWriter	Buffers output, uses platform's line separator	BufferedOutput Stream
CharArrayWriter	Writes to a character array	ByteArrayOutput Stream
FilterWriter	Abstract class for filtered	Filter Output Stream

	character output	
Output StreamWriter	Translates character stream into byte stream	(none)
FileWriter	Translates character stream into byte file	FileOutput Stream
PrintWriter	Prints values and objects to a Writer	PrintStream
PipedWriter	Writes to a PipedReader	PipedOutput Stream
StringWriter	Writes to a String	(none)

Check your progress 3

1. _________ is used to convert byte oriented data into character oriented data in Java.

   a. InputStreamReader c. StringStreamReader

   b. Output StreamReader d. PrintStreamReader

2.5 Using Byte Stream Classes

As studied earlier, the file copying is achieved in JDK 1.0 along with byte streams. It is found that the Byte streams will able to read or write files that carries ASCII characters which ranges from 0 to 255. So, byte streams will only be able to copy files having only English alphabets and no other language alphabets.

It is found that classes which supports byte streams is categorised as input stream classes and output stream classes. The only difference among them depends on type of job they do. Further, the input stream classes open the file in read mode whereas output stream classes opens file in write mode. Also, the character stream classes divide these into reader classes and writer classes. So the job of input stream classes and reader classes along with job of output stream classes and writer classes results in similar in spite of different divisions.

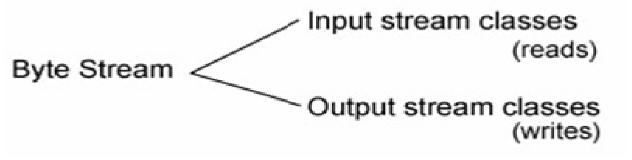


Fig 2.1 I/O Stream

Input Stream Hierarchy

The input stream classes are collection of several classes which performs similar job of opening file in read mode. Apart from this, they require different needs of programmer where classes exist in java.io package that results in hierarchical structure as shown:

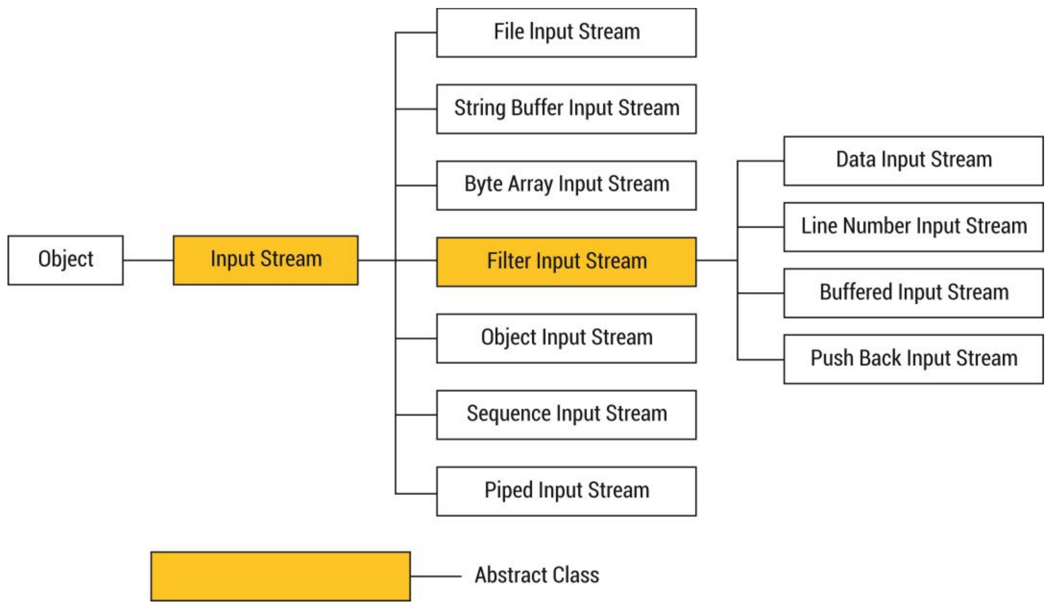


Fig 2.2 Input Stream hierarchy

Further it was observed that the hierarchy has 12 input stream classes where classes are divided in two categories as high-level streams and low-level streams. Out of 12 input stream classes, classes of Filter Input Stream are known as high-level streams having 4 and remaining are known as low-level streams that has 8.

Output Stream Hierarchy

Just like input stream hierarchy, the output stream hierarchy also deals with output stream classes.

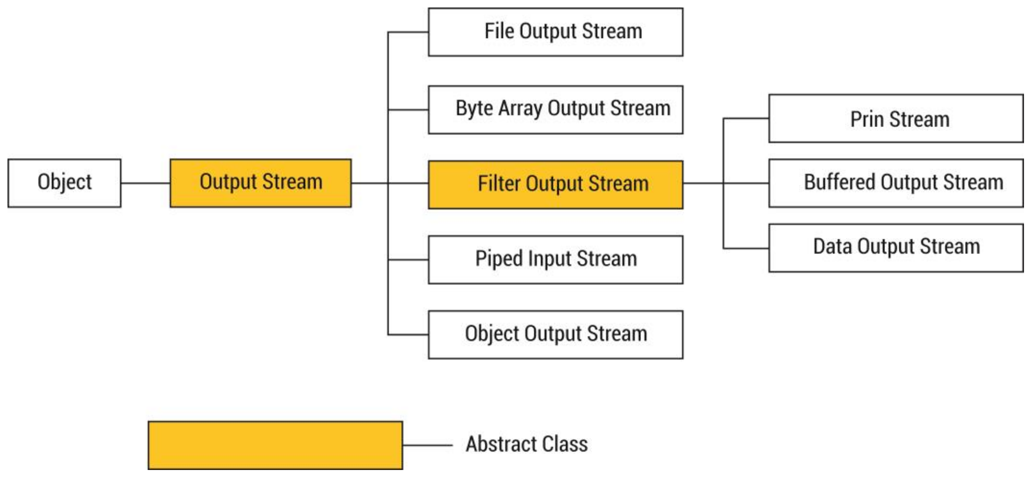


Fig 2.3 Output Stream hierarchy

In this, every subclasses of Filter Output Stream known as high-level streams has 3 streams while balance are known as low-level streams as 6. In this, 1 stream is linked to another to get higher functionality. Consider an example of java Byte Array Output Stream class where data is written in two files as shown:

```
import java.io.*;
class S{
 public static void main(String args[])throws Exception{
  FileOutputStream fout1=new FileOutputStream("f1.txt");
  FileOutputStream fout2=new FileOutputStream("f2.txt");
  ByteArrayOutputStream bout=new ByteArrayOutputStream();
  bout.write(139);
  bout.writeTo(fout1);
  bout.writeTo(fout2);
  bout.flush();
  bout.close();//has no effect
  System.out.println("success...");
 }
}
```

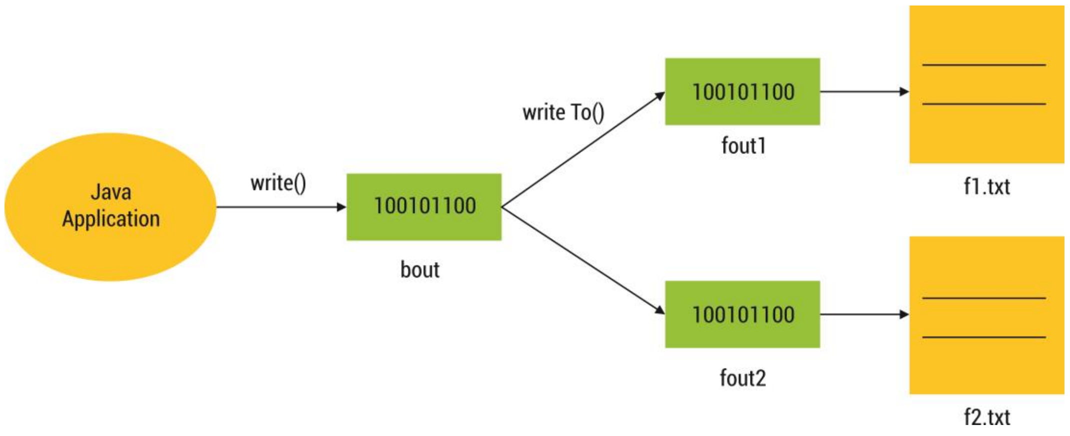


Fig 2.4 Output of program

Check your progress 4

1. Which method of InputStream is used to read integer of next available byte input?

   a. read()　　　c. get()

   b. scanf()　　　d. getInteger()

2. Which data type is returned by every method of Output Stream?

   a. int　　　c. byte

   b. float　　　d. None of above

2.6 Using Scanner and Console Classes in Java

Scanner Class

The Scanner class is obtained from java.util which shows a wrapper class that adds up input stream as stdin giving number of convenience methods that will read lines and breaks line into tokens. Such set of notes covers set of Scanner a method which is frequently used.

It is observed that a constructor for Scanner class takes Java InputStream, File or String as parameter which forms a Scanner object. Normally, Scanner class works with all which supports iterator that done through collection of tokens.

In Java, the variable System.in is declared as an InputStream and it points to stdin. System.in is a byte stream so you can't read from it directly if you want to read character strings, which is what you normally want to do. So it is better to wrap a Scanner object around System.in so as to take care about string oriented I/O. The following statement accomplishes this task:

```
Scanner console = new Scanner(System.in);
```

To illustrate the use of a Scanner, consider the following problem:

Finally, read and type check lines of user input where every line contains three fields organized as:

```
name(String) age(int) single(boolean)
```

Fields in parentheses shows errors that reports using proper error message. Here is a sample input file:

```
brad 10 true
nels 10
nels 10 false
nels 10 10
brad
10 20 30
10 true 40
brad nels
```

The output of the program shows:

```
line 2: must have a field for singleness
line 4 - 10: singleness should be a boolean
line 5: must have fields for age and singleness
line 6 - 30: singleness should be a boolean
line 7 - true: age should be an integer
line 7 - 40: singleness should be a boolean
Line 8: line must have the format 'name age singleness'
line 9 - nels: age should be an integer
line 9: must have a field for singleness
```

For this purpose, study the program shown which performs following task:

```
import java.util.Scanner;
class datacheck {
  static public void main(String args[]) {
    Scanner console = new Scanner(System.in);
    Scanner lineTokenizer;
int lineNum = 0;
while (consolehasNextLine()) {
lineTokenizer = new Scanner(console.nextLine());
lineNum++;
// determine if the line has a name field
```

```
if (lineTokenizer.hasNext()) {
        lineTokenizer.next(); {
        // consume the valid token
}
else {
System.out.printf("Line %d: line must have the format 'name age singleness'\n", lineNum);
        continue; // proceed to the next line of input
        }
  // determine if the line has a second field, and if so, whether that
  // field is an integer
  if (lineTokenizer.hasNext()) {
       if (lineTokenizer.hasNextInt()) {
        lineTokenizer.nextInt(); // consume the valid token
       }
       else
        System.out.printf("line %d - %s: age should be an integer\n",
                             lineNum, lineTokenizer.next());
  }
  else {
       System.out.printf("line %d: must have fields for age and singleness\n",
                           lineNum);
       continue; // proceed to the next line of input
  }
   // determine if the line has a third field, and if so, whether that
  // field is a boolean
  if (lineTokenizer.hasNext()) {
       if (lineTokenizer.hasNextBoolean())
        lineTokenizer.nextBoolean(); // consume the valid token
        else {
          System.out.printf("line %d - %s: singleness should be a boolean\n",
                              lineNum, lineTokenizer.next());
          continue; // proceed to the next line of input
        }
    }
      else {
          System.out.printf("line %d: must have a field for singleness\n", lineNum);
          continue; // proceed to the next line of input
      }
      lineTokenizer.close();  // discard this line
    }
  }
}
```

Console Class

In Java, Console class is an easy way to deal with stdio that supports reading from stdin and writing to stdout. It reads character strings so you do not have to worry about wrapping System.in inside a Scanner class.

You can obtain the system provide console object from System.console. It is a good idea to check whether this object exists, since it does not exist for non-interactive Java programs, which includes program in which stdin is redirected. For example, the following invocation will not have a console object:

```
java foo < inputfile    // no console object for programs with redirected input
```

A console object also may not exist for Java programs run with older versions of the Java virtual machine and some systems will not provide it for security purposes. In all of these cases you will have to fall back on System.in and System.out.

Consider a sample code:

```
import java.util.Scanner;
import java.io.Console;   // Console is in the java.io library
class datacheck {
        static public void main(String args[]) {
        Scanner lineTokenizer;
        String input_line;
        Console input_reader = System.console();
        // I'm just going to exit if the console is not provided
        if (input_reader == null) {
        System.err.println("No console.");
        System.exit(1);
        }
        // you cannot use the C/C++ idiom of
        //   while (input_line = input_reader.readLine())
        // because the test expects a boolean and the above assignment returns
        // a String reference. In C/C++ the return of null would be interpreted
        // as false, but no so in Java. Hence the infinite loop that I have written
        // with a break statement for when readLine returns null
        while(true) {
        input_line = input_reader.readLine();
        if (input_line == null) break;
        line_scanner = new Scanner(input_line);
        int lineNum = 0;
        lineNum++;
        // determine if the line has a name field
        if (lineTokenizer.hasNext()) {
        lineTokenizer.next();   // consume the valid token
        }
        else {
```

```
console.printf("Line %d: line must have the format 'name age singleness'\n", lineNum);
continue;  // proceed to the next line of input
}
// determine if the line has a second field, and if so, whether that
// field is an integer
if (lineTokenizer.hasNext()) {
if (lineTokenizer.hasNextInt()) {
lineTokenizer.nextInt(); // consume the valid token
}
else
console.printf("line %d - %s: age should be an integer\n",
                lineNum, lineTokenizer.next());
}
else {
console.printf("line %d: must have fields for age and singleness\n",
               lineNum);
continue; // proceed to the next line of input
}
// determine if the line has a third field, and if so, whether that
// field is a boolean
if (lineTokenizer.hasNext()) {
if (lineTokenizer.hasNextBoolean())
lineTokenizer.nextBoolean(); // consume the valid token
else {
console.printf("line %d - %s: singleness should be a boolean\n",
                lineNum, lineTokenizer.next());
continue; // proceed to the next line of input
}
```

```
}
else {
console.printf("line %d: must have a field for singleness\n", lineNum);
continue; // proceed to the next line of input
}
lineTokenizer.close(); // discard this line
            }
      }
}
```

Check your progress 5

1. Which among the following Scanner class is used to scans the next token as a byte?

    a. public String nextLine() c. public short nextShort()

    b. public byte nextByte() d. public int nextInt()

2.7 Skills Check

There are lots of skills required to read Java's input from System.in console. It can be done by consider following two ways to read input from console such as:

a. Input Stream Reader wrapped in a Buffered Reader

b. Scanner classes in JDK1.5

If we form a class called Console Input using main method, then you have to consider a code shown where comment is to be replaced;

```
InputStream stream = System.in;
Scanner scanner = new Scanner(stream);
System.out.println("Name:");
String input = scanner.next();
System.out.println("Hello " + input);
scanner.close();
```

The output we get with errors as:

```
import java.io.InputStream;

import java.util.Scanner;
```

Check your progress 6

1. If we form a class called Console Input using main method, then we have to consider a code shown where comment is to be replaced

   a. True

   b. False

2.8 Exercises

Exercise 1:

Consider a Java Scanner class which reads int, string and double value as an input?

Solution:

```
import java.util.Scanner;
class ScannerTest{
 public static void main(String args[]){
   Scanner sc=new Scanner(System.in);
   System.out.println("Enter your rollno");
   int rollno=sc.nextInt();
   System.out.println("Enter your name");
   String name=sc.next();
   System.out.println("Enter your fee");
   double fee=sc.nextDouble();
   System.out.println("Rollno:"+rollno+"  name:"+name+"  fee:"+fee);
   sc.close();
 }
}
```

On running the above program, we get:

```
Enter your rollno
10
Enter your name
Anika
Enter
60000
Rollno:10 name:Anika fee:60000
```

2.9 Let Us Sum Up

It is learnt in this unit that file is a mixture of bytes that are stored in secondary storage device which can be applied to read, write, append or update data.

It is seen that working with files uses input and output operation which can be implement using screen and keyboard. Once the program finished executing, then all entered data gets lost as primary memory is volatile

It is studied that in Byte oriented stream, data is written and read from file using 8-bit bytes while in Character oriented stream, data is accessed by character per time.

In Java, files and directories are carried out and calculated by File class. The File class will not really show for input and output to files. It shows an identifier of files and directories.

Further it is studied that Scanner class is obtained from java.util that shows wrapper class which adds input stream as stdin by giving number of convenience methods which read lines and breaks line into tokens. While Console class is an easy way to deal with stdio that supports reading from stdin and writing to stdout.

2.10 Answers for Check Your Progress

Check your progress 1

Answers: (1-c), (1-b)

Check your progress 2

Answers: (1-b)

Check your progress 3

Answers: (1-a)

Check your progress 4

Answers: (1-a), (1-d)

Check your progress 5

Answers: (1-b)

Check your progress 6

Answers: (1-a)

2.11 Glossary

1. **File -** It is a combination of bytes that are placed inside secondary storage device.
2. **Stream -** It is a channel through which data flow from source to destination
3. **Byte oriented stream -** In Java, the data in byte oriented stream is written and read from file using 8-bit bytes.
4. **Character oriented stream -** In Java, data in character oriented stream is accessed each character per time.

2.12 Assignment

Discuss Byte oriented steam and Character oriented stream.

2.13 Activities

Give some examples related class constructor in Java.

2.14 Case Study

Discuss the output of this program?

```
import java.io.*;

class filesinputoutput {

   public static void main(String args[]) {

      InputStream obj = new FileInputStream("inputoutput.java");

      System.out.print(obj.available());

   }

}
```

2.15 Further Readings

1. The online Java tutorial @ http://docs.oracle.com/javase/tutorial/.

2. Paul Deitel and Harvey Deitel, "Java How to Program", 9th ed, 2011.

3. Y. Daniel Liang, "Introduction to Java Programming", 9th ed, 2012.

4. Bruce Eckel, "Thinking in Java", 4th ed, 2007.

UNIT 3: BUILDING GRAPHICAL USER INTERFACE (SWING)

Unit Structure

3.0 Learning Objectives

3.1 Introduction

3.2 Building Applications using Classes Related to Graphical Interface

3.3 Creating Layouts in GUI

3.4 Using Events in GUI Applications

3.5 Skills Check

3.6 Exercises

3.7 Let Us Sum Up

3.8 Answers for Check Your Progress

3.9 Glossary

3.10 Assignment

3.11 Activities

3.12 Case Study

3.13 Further Readings

3.0 Learning Objectives

After learning this unit, you will be able to understand:

- The applications using classes.
- The Layouts in GUI.
- The Events in GUI applications.

3.1 Introduction

GUI also known as Graphical User Interface uses icons or other visual indicators to work with electronic devices instead of working only with text with the help of command line. It is found that all versions of Windows use GUI apart from MS-DOS. The interface was initially developed at Xerox PARC by Alan Kay, Douglas Engelbart. In this, Java Graphics like APIs, AWT and Swing, delivers high reusable GUI components like button, text field, label, choice, panel and frame for certain GUI applications. In Java, the classes can be reused again instead of re-invent the wheels.

3.2 Building Applications using Classes Related to Graphical Interface

The computer systems require some form of user interface which will help to communicate among human beings. Out the whole, the commonly used interface as of today is graphical user interface which are used in most computers.

The graphical user interface will use visual elements which will help in showing information that is kept inside computer which is easy to understand. Such elements make easy for people to work by involving certain computer software. It is found that a GUI uses windows, icons, and menus to carry out commands that can be in shape of opening files, deleting files and moving files. The GUI software is easier to apply as compared to command line software as there is no need to type and to keep in memory the individual commands.

It is found that the most common graphical elements are shown by WIMP which is a mixture of window, icon, menu and pointer. In this:

- **Window:** It is a space on the screen which shows information. The contents here are displayed irrespective from rest of the screen.
- **Icon:** An icon is small picture which shows objects like program or file which can be easily carried out in different ways.
- **Menu:** It shows a list of choices to users. In this, the options are selected with the use of mouse or other pointing device. It shows the possibility inside a software which is in terms of commands and functions.
- **Pointer:** It is an onscreen symbol which shows movement of device which is controlled by the user in order to select windows, icons and menus.

These four elements have dominated user interface design since they were first introduced in the mid-1980s. Most recently, mobile devices, such as smartphones and tablets, have starting using these elements in new ways due to constraints in space and available input devices. Touchscreen technology has introduced new interactions such as pinching and rotating, which are not supported by traditional input devices.

Importance of Interface Design

To design a good user interface is a difficult task. A good user interface will allow an easy and natural interaction among user and system. While a slick look interface is cool which concerns with overriding when it comes to user interface design that is usable. Usability is basically ISO standard to which a product can be used by specific users having specified goals that works with effectiveness, efficiency and satisfaction in particular manner. There are certain practical terms that are well-designed with interface such as:

- **Leamability:** How easy is it for users to accomplish basic tasks?
- **Efficiency:** How quickly can they perform tasks?
- **Memorability:** How easily can users establish proficiency?
- **Errors:** How many errors do users make?
- **Satisfaction:** How pleasant is it to use the design?

User interface design is critical to the success of a system, and it should be part of the original design concepts for the system.

Check your progress 1

1. Identify the three software parts of GUI program?

   a. Windows, Buttons, Mice

   b. GUI Components, Graphics, Code

   c. GUI Components, Event Listeners, Application Code

   d. Frames, Code, Events

3.3 Creating Layouts in GUI

There are two different modes you can use to arrange and organize your GUIs: Fixed and Automatic. Up until now, every Unity GUI example provided in this guide has used Fixed Layout. To use Automatic Layout, write GUI Layout instead of GUI when calling control functions. You do not have to use one Layout mode over the other, and you can use both modes at once in the same On GUI() function.

Fixed Layout makes sense to use when you have a pre-designed interface to work from. Automatic Layout makes sense to use when you don't know how many elements you need up front, or don't want to worry about hand-positioning each Control. For example, if you are creating a number of different buttons based on Save Game files, you don't know exactly how many buttons will be drawn. In this case Automatic Layout might make more sense. It is really dependent on the design of your game and how you want to present your interface.

There are two key differences when using Automatic Layout:

GUI Layout is used instead of GUI

No Rect() function is required for Automatic Layout Controls

```
/* Two key differences when using Automatic Layout */

// JavaScript

function OnGUI () {

    // Fixed Layout

    GUI.Button (Rect (25,25,100,30), "Fixed Layout Button");

    // Automatic Layout

    GUILayout.Button ("Automatic Layout Button");

}

// C#

using UnityEngine;

using System.Collections;

public class GUITest : MonoBehaviour {

    void OnGUI () {

        // Fixed Layout

        GUI.Button (new Rect (25,25,100,30), "Fixed Layout Button");

        // Automatic Layout

        GUILayout.Button ("Automatic Layout Button");

    }

}
```

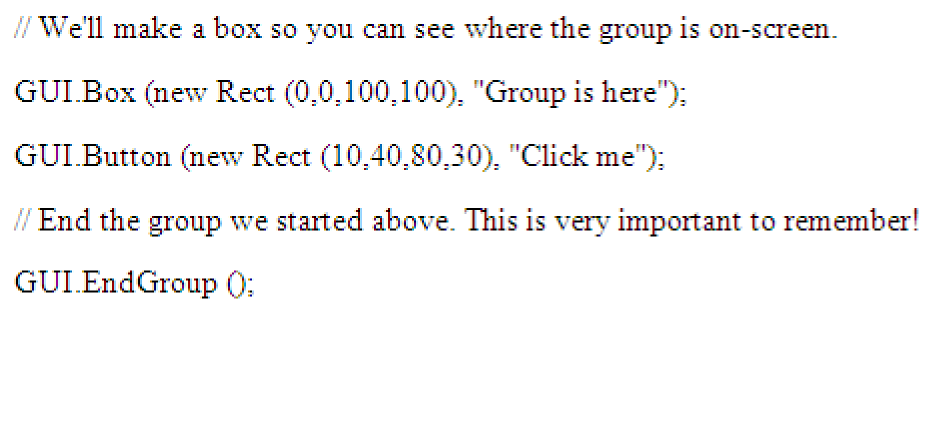
Arranging Controls

Depends on the type of layout mode, there exists different controls which can handle and shows grouping. It is found that in Fixed Layout, you will put different Controls into Groups, whereas in case of Automatic Layout, you will be able to control and handle Areas, Horizontal Groups and Vertical Groups.

Fixed Layout - Groups

Groups are a convention available in Fixed Layout Mode. They allow you to define areas of the screen that contain multiple Controls. You define which Controls are inside a Group by using the GUI.BeginGroup() and GUI.EndGroup() functions. All Controls inside a Group will be positioned based on the Group's top-left corner instead of the screen's top-left corner. It is found that if you reposit group at runtime, then relative positions of controls in group will be adjusted. As an example, it's very easy to center multiple Controls on-screen.

```
/* Center multiple Controls on the screen using Groups */
//JavaScript
function OnGUI () {
        // Make a group on the center of the screen
        GULBeginGroup (Rect (Screen.width 2 - 50, Screen.height / 2 -
50, 100, 100));
        // All rectangles are now adjusted to the group. (0,0) is the top
left corner of the group.
        // we'll make a box so you can see where the group is on-screen
GUI.Box (Rect (0,0,100,100), "Group is here");
GUI.Button (Rect (10,40,80,30), "Click me");
        End the group we started above. This is very important to
remember! GUI.EndGroup 0;
}
// C#
using UnityEngine;
using System.Collections;
public class GUITest : MonoBehaviour {
void OnGUI () {
        // Make a group on the center of the screen
        GULBeginGroup (new Rect (Screen.width / 2 - 50, Screen.height ' 2 -
    50, 100, 100));
        // All rectangles are now adjusted to the group. (0,0) is the topleft
    corner of the group.
```



```
    // We'll make a box so you can see where the group is on-screen.
    GUI.Box (new Rect (0,0,100,100), "Group is here");
    GUI.Button (new Rect (10,40,80,30), "Click me");
    // End the group we started above. This is very important to remember!
    GUI.EndGroup ();
  }
}
```

You can also nest multiple Groups inside each other. When you do this, each group has its contents clipped to its parent’s space.

```
/* Using multiple Groups to clip the displayed Contents */
//JavaScript
var bgImage : Texture2D; // background image that is 256 x 32
var fglmage : Texture2D; // foreground image that is 256 x 32
var playerEnergy = 1.0; //a float between 0.0 and 1.0
function OnGUI 0 {
        // Create one Group to contain both images
        // Adjust the first 2 coordinates to place it somewhere else on-screen
GUI.BeginGroup (Rect (0,0,256,32));
        // Draw the background image
        GUI.Box (Rect (0,0,256,32), bglmage);
        // Create a second Group which will be clipped
        // We want to clip the image and not scale it, which is why we need
the second Group
GUI.BeginGroup (Rect (0,0,p1ayerEnergy * 256, 32));
// Draw the foreground image
```

```
    GUI.Box (Rect (0,0,256,32), fgImage);
    // End both Groups
    GUI.EndGroup ();
    GUI.EndGroup ();
    }
// C#
    using UnityEngine;
    using System.Collections;
    public class GUITest : MonoBehaviour {
    // background image that is 256 x 32
    public Texture2D bglmage;
    // foreground image that is 256 x 32
    public Texture2D fglmage;
    // a float between 0.0 and 1.0
    public float playerEnergy = 1.0f;
    void OnGUI 0 {
    // Create one Group to contain both images
    // Adjust the first 2 coordinates to place it somewhere else on-screen
    GUL.BeginGroup (new Rect (0,0,256,32));
    // Draw the background image
    GULBox (new Rect (0,0, 256,32), bgImage);
    // Create a second Group which will be clipped
    // We want to clip the image and not scale it, which is why we need the
    second Group
    GUI.BeginGroup (new Rect (0,0,playerEnergy * 256, 32));
```

```
            // Draw the foreground image
            GUI.Box (new Rect (0,0,256,32), fgImage);
            // End both Groups
            GUI.EndGroup ();
        GUI.EndGroup ();
    }
}
```

Check your progress 2

1. Which among the following component exists in all GUI programs?

a. Frame	c. Monitor
b. Mouse	d. Button

3.4 Using Events in GUI Applications

To design an interactive Graphical User Interfaces is a difficult task, especially for those who have no experience. Creating user interfaces with the help of toolkit is time consuming process as it does not integrate in scientific-computing work-flow during explanation of algorithms and data-flow where objects shown by GUI are likely to change.

Visual computing, where the programmer creates first a graphical interface and then writes the callbacks of the graphical objects, gives rise to a slow development cycle, as the work-flow is centered on the GUI, and not on the code.

It is seen that if you want your program to calculate geometric objects, then you have to guide computer about set of 3 numbers and to tell him how to rotate that point along given axis. Also, if you want to use sphere, then a bit more work requires your program to function which will create points, spheres, etc. It knows how to rotate them, to mirror them, to scale them. So in pure procedural programming you will have procedures to rotate, scale, mirror, each one of your objects. If you want to rotate an object you will first have to find its type and then apply the right procedure to rotate it.

In object oriented programming, new abstraction just like object carries both data and procedures which is applied and altered data. In this, the data entries are known as attributes of object and procedures methods. So with the help of object oriented programming, an object is described to rotate. A point object could be implemented in python with:

```
code snippet #0
from numpy import cos, sin
class Point(object):
    """ 3D Point objects """
    x = 0.
    y = 0.
    z = 0.
    def rotate_z(self, theta):
        """ rotate the point around the Z axis """
        xtemp =  cos(theta) * self.x + sin(theta) * self.y
        ytemp = -sin(theta) * self.x + cos(theta) * self.y
        self.x = xtemp
        self.y = ytemp
```

The above program code will form a Point class. Point objects can be created as instances of the Point class as:

```
>>> from numpy import pi
>>> p = Point()
>>> p.x = 1
>>> p.rotate_z(pi)
>>> p.x
-1.0
>>> p.y
1.2246467991473532e-16
```

To carry out objects, developer requires no knowledge about internal details of their procedures. It is seen that as long as object has rotate method, the developer knows how to rotate it.

Check your progress 3

1. A Graphics object is:

   a. object showing part of Frame which can be drawn.

   b. object that shows whole Frame.

   c. object that shows full monitor.

   d. object that shows graphics board.

3.5 Skills Check

In Java GUI programming, lot of skills are used to interface icons and elements with electronic devices.. The knowledge about Automatic Layout and Fixed Layout will allow user to design a program which supports several elements.

In order to design a graphical user interface, you require following steps:

- Creating a new form or dialog
- Placing and arranging components in the form
- Defming properties of the components
- Binding components to the source code
- Making forms functional
- Localizing forms
- Previewing Forms

Check your progress 4

1. In order to design a graphical user interface, it require

   a. Creating a new form of dialogue

   b. Placing and arranging components in the form

   c. Defining properties of the components

   d. All of the above

3.6 Exercises

Exercise 1:

Consider a Swing application in Java and write a program with the help of application in Jython?

Solution:

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
public class HelloWorld {
  public static void main(String[] args) {
    JFrame frame = new JFrame("Hello Java!");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(300, 300);
    JButton button = new JButton("Click Me!");
    button.addActionListener(
      new ActionListener() {
        public void actionPerformed(ActionEvent event) {
          System.out.println("Clicked!");
        }
      }
     );
     frame.add(button);
     frame.setVisible(true);
   }
 }
```

This simple application draws a JFrame that is completely filled with a JButton. When the button is pressed, “Click Me!” prints to the command line.

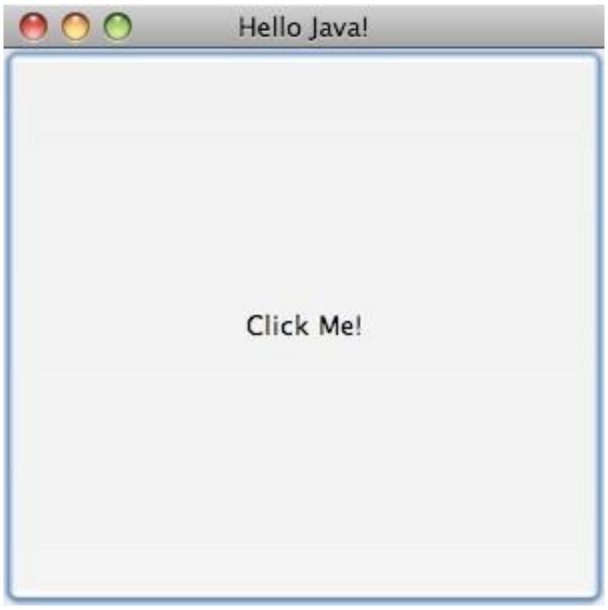


Fig 3.1 Java

3.7 Let Us Sum Up

In this unit we have learnt that GUI is a Graphical User Interface that uses icons or visual indicators to work with electronic devices rather than working with text along with command line.

It is seen that graphical user interface uses visual elements that will help in showing information which is kept inside computer that is simple to understand. These elements makes easy for people to work by involving certain computer software It is found that in order to design a good user interface it should allow easy and natural interaction among user and system.

3.8 Answers for Check Your Progress

Check your progress 1

Answers: (1-c)

Check your progress 2

Answers: (1-a)

Check your progress 3

Answers: (1-a)

Check your progress 4

Answers: (1-d)

3.9 Glossary

1. **GUI** - It is a Graphical User Interface that works with icons or indicators along with electronic devices
2. **Interface** - It is a coordination involved among computer, program and humans.

3.10 Assignment

Discuss the advantages of GUI?

3.11 Activities

Fill in the blanks so that this program displays a Frame:

```
import java.awt.*;

public class microGUI

{

  public static void main ( String[] args )

   {

    Frame frm = new ___________();

    frm.___________( 150, 100 );

    frm.___________( true );

   }

}
```

3.12 Case Study

Prepare a report describing about various GUI elements.

3.13 Further Readings

1. The online Java tutorial @ http://docs.oracle.com/javase/tutorial/.
2. Paul Deitel and Harvey Deitel, "Java How to Program", 9th ed, 2011.
3. Y. Daniel Liang, "Introduction to Java Programming", 9th ed, 2012.
4. Bruce Eckel, "Thinking in Java", 4th ed, 2007.

Block Summary

The block gives detailed explanation to students on Java Library Classes along with understanding of Graphical User Interface. The concept of files handling in Java will give more knowledge in terms of how data is kept and work through these files. The information related to input and output operation of file with concept of Byte oriented stream and Character oriented stream will really beneficial to user who has little knowledge about Java programming.

After this block, student will get knowledge about Scanner and Console classes with usage on java.util and basic understanding of wrapper class with input stream stdin. The knowledge about utilities in java.util along with pogramming structures allow students to work on professional written classes. The idea about Java.util package will be beneficial to them in longer use.

Block Assignment

Short Answer Questions

1. What are wrapper classes?
2. It is possible to use the File class to list the contents of the current working directory.
3. What is the purpose of the File class?
4. Explain Java.util package with examples?

Long Answer Questions

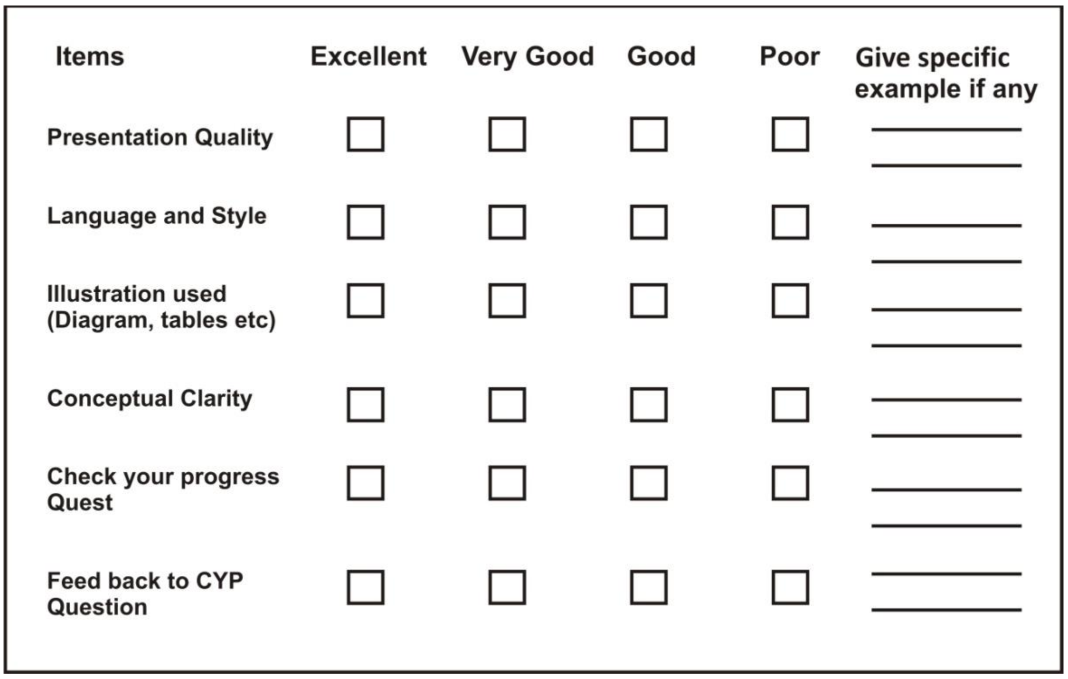
1. Explain in detail about GUI in Java?
2. Compare among Character oriented stream and Byte oriented stream?
3. What is meant by Stream and what are the types of Streams and classes of the Streams?

Enrolment No. ☐

1. How many hours did you need for studying the units?

Unit No	**1**	**2**	**3**	**4**
Nos of Hrs				

2. Please give your reactions to the following items based on your reading of the block:



Items	Excellent	Very Good	Good	Poor	Give specific example if any
Presentation Quality	☐	☐	☐	☐	______ ______
Language and Style	☐	☐	☐	☐	______ ______
Illustration used (Diagram, tables etc)	☐	☐	☐	☐	______ ______
Conceptual Clarity	☐	☐	☐	☐	______ ______
Check your progress Quest	☐	☐	☐	☐	______ ______
Feed back to CYP Question	☐	☐	☐	☐	______ ______

3. Any Other Comments

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

Education is something which ought to be brought within the reach of every one.

- Dr. B. R. Ambedkar

Dr. Babasaheb Ambedkar Open University
Jyotirmay' Parisar, Opp. Shri Balaji Temple, Sarkhej-Gandhinagar Highway, Chharodi, Ahmedabad-382 481.