INTRODUCTION TO JAVA

Unit Structure

1.1 INTRODUCTION:

Java is a high-level, third generation programming language, like C, FORTRAN, Smalltalk, Perl, and many others. You can use Java to write computer applications that play games, store data or do any of the thousands of other things computer software can do. Compared to other programming languages, Java is most similar to C. However although Java shares much of C's syntax, it is not C. Knowing how to program in C or, better yet, C++, will certainly help you to learn Java more quickly, but you don't need to know C to learn Java. A Java compiler won't compile C code, and most large C programs need to be changed substantially before they can become Java programs. What's most special about Java in relation to other programming languages is that it lets you write special programs called applets that can be downloaded from the Internet and played safely within a web browser. Java language is called as an Object-Oriented Programming language and before begining for Java, we have to learn the concept of OOPs(Object-Oriented Programming).

1.2 BASIC CONCEPT OF OOPS (OBJECT-ORIENTED PROGRAMMING):

There are some basic concepts of object oriented programming as follows:

1. Object
2. Class
3. Data abstraction

4. Data encapsulation
5. Inheritance
6. Polymorphism
7. Dynamic binding

1. Object

Objects are important runtime entities in object oriented method. They may characterize a location, a bank account, and a table of data or any entry that the program must handle. For example:

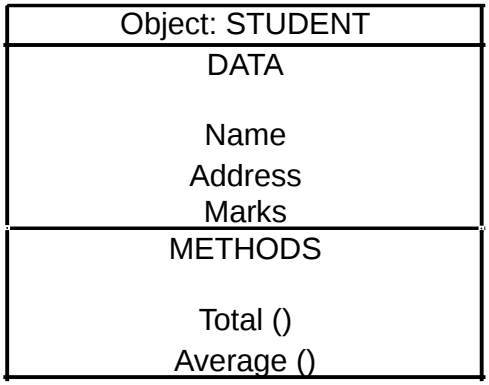


Fig.1.1 Representation of an object “STUDENT”

Each object holds data and code to operate the data. Object can interact without having to identify the details of each other’s data or code. It is sufficient to identify the type of message received and the type of reply returned by the objects. Another example of object is CAR

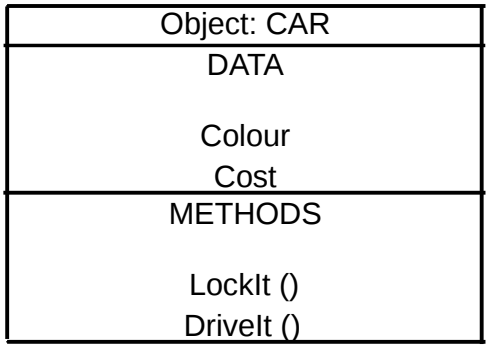


Fig.1.2 Representation of object “CAR”

Fig.1.1 and Fig.1.2 shows actual representation of object.

2. Classes

A class is a set of objects with similar properties (attributes), common behaviour (operations), and common link to other objects.

The complete set of data and code of an object can be made a user defined data type with the help of class.

The objects are variable of type class. A class is a collection of objects of similar type. Classes are user defined data types and work like the build in type of the programming language. Once the class has been defined, we can make any number of objects belonging to that class. Each object is related with the data of type class with which they are formed.

As we learned that, the classification of objects into various classes is based on its properties (States) and behaviour (methods). Classes are used to distinguish are type of object from another. The important thing about the class is to identify the properties and procedures and applicability to its instances.

For example: Vehicle

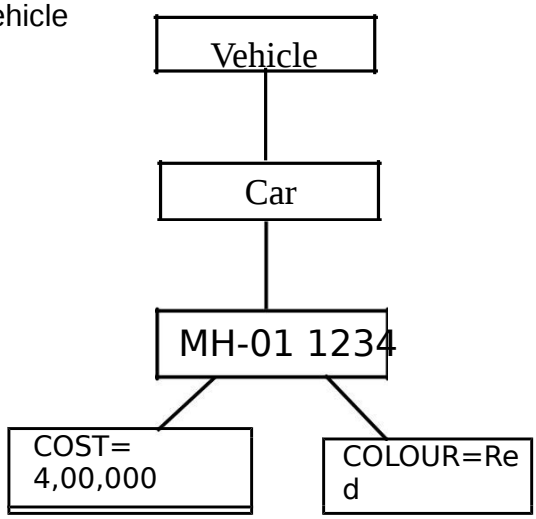


Fig.1.3 Representation of class

In above example, we will create an objects MH-01 1234 belonging to the class car. The objects develop their distinctiveness from the difference in their attribute value and relationships to other objects.

3. Data Abstraction

Data abstraction refers to the act of representing important description without including the background details or explanations.

Classes use the concept of abstraction and are defined as a list of abstract attributes such as size, cost and functions operate on these attributes. They summarize all the important properties of the objects that are to be created.

Classes use the concepts of data abstraction and it is called as Abstract Data Type (ADT).

4. Data Encapsulation

Data Encapsulation means wrapping of data and functions into a single unit (i.e. class). It is most useful feature of class. The data is not easy to get to the outside world and only those functions which are enclosed in the class can access it.

These functions provide the boundary between Object's data and program. This insulation of data from direct access by the program is called as **Data hiding**.

For example:

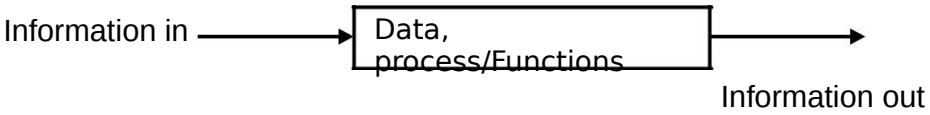


Fig 1.4: Encapsulation

5. Inheritance

Inheritance is the process by which objects of one class can get the properties of objects of another class. Inheritance means one class of objects inherits the data and behaviours from another class. Inheritance maintains the hierarchical classification in which a class inherits from its parents.

Inheritance provides the important feature of OOP that is reusability. That means we can include additional characteristics to an existing class without modification. This is possible deriving a new class from existing one.

In other words, it is property of object-oriented systems that allow objects to be built from other objects. Inheritance allows openly taking help of the commonality of objects when constructing new classes. Inheritance is a relationship between classes where one class is the parent class of another (derived) class. The derived class holds the properties and behaviour of base class in addition to the properties and behaviour of derived class.

For Example:

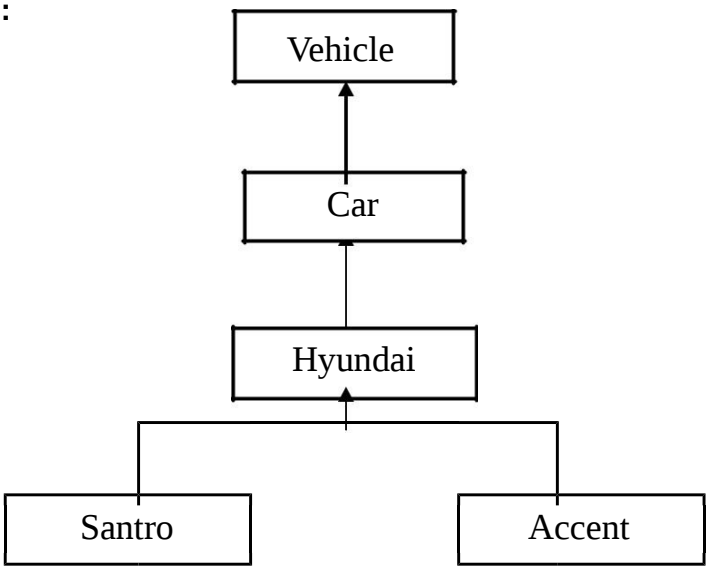


Fig.1.5 Inheritance

In Fig.1.5, the Santro is a part of the class Hyundai which is again part of the class car and car is the part of the class vehicle. That means vehicle class is the parent class.

6. Polymorphism

(Poly means “many” and morph means “form”). Polymorphism means the ability to take more than one form. Polymorphism plays a main role in allocate objects having different internal structures to share the same external interface. This means that a general class of operations may be accessed in the same manner even though specific activities associated with each operation may differ. Polymorphism is broadly used in implementing inheritance.

It means objects that can take on or assume many different forms. Polymorphism means that the same operations may behave differently on different classes. Booch defines polymorphism as the relationship of objects many different classes by some common super class. Polymorphism allows us to write generic, reusable code more easily, because we can specify general instructions and delegate the implementation detail to the objects involved.

For Example:

In a pay roll system, manager, office staff and production worker objects all will respond to the compute payroll message, but the real operations performed are object particular.

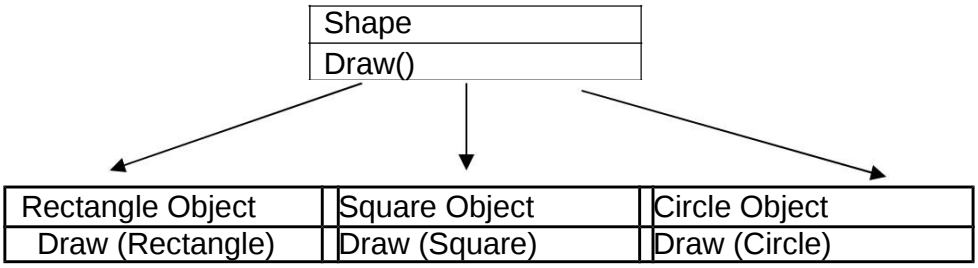


Fig.1.6 Polymorphism

7. Dynamic Binding

Binding refers to the linking of a procedure call to the code to be executed in response to the call. Dynamic binding means that the code related with a given procedure call is not known until the time of the call at run time.
Dynamic binding is associated polymorphism and inheritance.

1.3 JAVA HISTORY :

Java is a general-purpose, object-oriented programming language developed by Sun Microsystems of USA in 1991.Originally called Oak by James Gosling (one of the inventor of the language). Java was invented for the development of software for cunsumer electronic devices like TVs, tosters, etc. The main aim had to make java simple, portable and reliable. Java Authors: James , Arthur Van , and others

Following table shows the year and beginning of Java.

Year	**Progress**
1990	Sun decided to developed software that could be used for electronic devices. And the project called as Green Project head by James Gosling.
1991	Announcement of a new language named “Oak”
1992	The team verified the application of their new language to manage a list of home appliances using a hand held device.
1993	The World Wide Web appeared on the Internet and transformed the text-based interface to a graphical rich environment.
1994	The team developed a new Web browsed called “Hot

	Java" to locate and run Applets.
1995	Oak was renamed to Java, as it did not survive "legal" registration. Many companies such as Netscape and Microsoft announced their support for Java.
1996	Java language is now famous for Internet programming as well as a general purpose OO language.
1997	Sun releases Java Development Kit(JDK 1.1)
1998	Sun releases Software Development Kit (SDK 1.2)
1999	Sun releases Java 2 platform Standard Edition (J2SE) and Enterprise Edition(J2EE).
2000	J2SE with SDK 1.3 was released.
2002	J2SE with SDK 1.4 was released.
2004	J2SE with JDK 5.0 was released.

1.4 JAVA FEATURES:

As we know that the Java is an object oriented programming language developed by Sun Microsystems of USA in 1991. Java is first programming language which is not attached with any particular hardware or operating system. Program developed in Java can be executed anywhere and on any system.

Features of Java are as follows:

1. Compiled and Interpreted
2. Platform Independent and portable
3. Object- oriented
4. Robust and secure
5. Distributed
6. Familiar, simple and small
7. Multithreaded and Interactive
8. High performance
9. Dynamic and Extensible

1. Compiled and Interpreted

Basically a computer language is either compiled or interpreted. Java comes together both these approach thus making Java a two-stage system.

Java compiler translates Java code to Bytecode instructions and Java Interpreter generate machine code that can be directly executed by machine that is running the Java program.

2. Platform Independent and portable

Java supports the feature portability. Java programs can be easily moved from one computer system to another and anywhere. Changes and upgrades in operating systems, processors and system resources will not force any alteration in Java programs. This is reason why Java has become a trendy language for programming on Internet which interconnects different kind of systems worldwide. Java certifies portability in two ways.
First way is, Java compiler generates the bytecode and that can be executed on any machine. Second way is, size of primitive data types are machine independent.

3. Object- oriented

Java is truly object-oriented language. In Java, almost everything is an Object. All program code and data exist in objects and classes. Java comes with an extensive set of classes; organize in packages that can be used in program by Inheritance. The object model in Java is trouble-free and easy to enlarge.

4. Robust and secure

Java is a most strong language which provides many securities to make certain reliable code. It is design as garbage – collected language, which helps the programmers virtually from all memory management problems. Java also includes the concept of exception handling, which detain serious errors and reduces all kind of threat of crashing the system.

Security is an important feature of Java and this is the strong reason that programmer use this language for programming on Internet.

The absence of pointers in Java ensures that programs cannot get right of entry to memory location without proper approval.

5. Distributed

Java is called as Distributed language for construct applications on networks which can contribute both data and programs. Java applications can open and access remote objects on Internet easily. That means multiple programmers at multiple remote locations to work together on single task.

6. Simple and small

Java is very small and simple language. Java does not use pointer and header files, goto statements, etc. It eliminates operator overloading and multiple inheritance.

7. Multithreaded and Interactive

Multithreaded means managing multiple tasks simultaneously. Java maintains multithreaded programs. That means we need not wait for the application to complete one task before starting next task. This feature is helpful for graphic applications.

8. High performance

Java performance is very extraordinary for an interpreted language, majorly due to the use of intermediate bytecode. Java architecture is also designed to reduce overheads during runtime. The incorporation of multithreading improves the execution speed of program.

9. Dynamic and Extensible

Java is also dynamic language. Java is capable of dynamically linking in new class, libraries, methods and objects. Java can also establish the type of class through the query building it possible to either dynamically link or abort the program, depending on the reply.

Java program is support functions written in other language such as C and C++, known as native methods.

1.5 COMPARISON IN JAVA AND C++

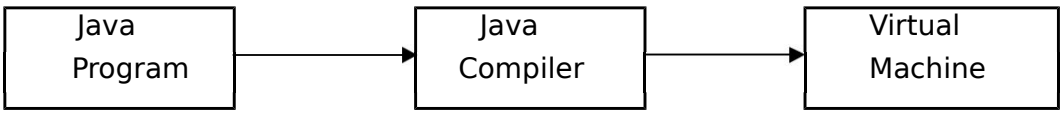
	Java	C++
1	Java is true Object-oriented language.	C++ is basically C with Object-oriented extension.
2	Java does not support operator overloading.	C++ supports operator overloading.
3	It supports labels with loops and statement blocks	It supports goto statement.
4	Java does not have C++ template classes as in C++.	has template classes.
5	Java compiled into Source code can be written byte code for the Java to be platform independent Virtual Machine. The andwrittentotake source code is advantage of platform.C++ independent on typically compiled into operating system. machine code.	

6	Java does not support multiple inheritance of classes but it supports interface.	C++ supports multiple inheritance of classes.
7	Runs in a protected virtual machine.	Exposes low-level system facilities.
8	Java does not support global variable. Every variable should declare in class.	C++ support global variable.
9	Java does not use pointer.	C++ uses pointer.
10	It Strictly enforces an object oriented programming paradigm.	It Allows both procedural programming and object-oriented programming.
11	There are no header files in Java.	We have to use header file in C++.

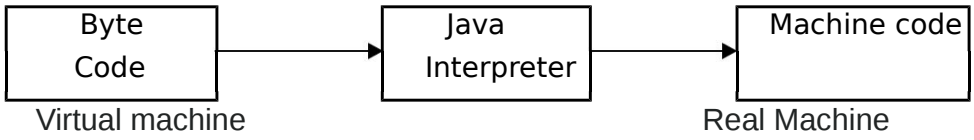
1.6 JAVA VIRTUAL MACHINE:

As we know that all programming language compilers convert the source code to machine code.Same job done by Java Compiler to run a Java program, but the difference is that Java compiler convert the source code into Intermediate code is called as bytecode. This machine is called the *Java Virtual machine* and it exits only inside the computer memory.

Following figure shows the process of compilation.



The Virtual machine code is not machine specific. The machine specific code is generated. By Java interpreter by acting as an intermediary between the virtual machine and real machines shown below



Java Object Framework act as the intermediary between the user programs and the virtual machine which in turn act as the intermediary between the operating system and the Java Object Framework.

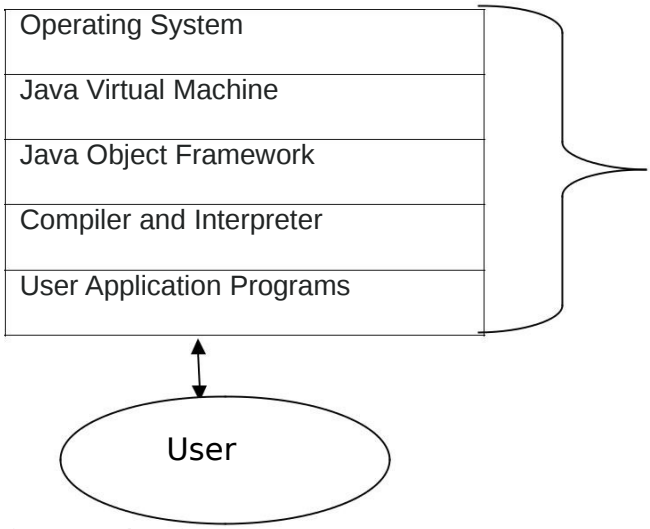


Fig: Layers of Interaction for Java programs

1.7 JAVA ENVIRONMENT:

Java environment includes a number of development tools, classes and methods. The development tools are part of the system known as Java Development Kit (JDK) and the classes and methods are part of the Java Standard Library (JSL), also known as the Application Programming Interface (API).

Java Development kit (JDK) – The JDK comes with a set of tools that are used for developing and running Java program. It includes:

1. Appletviewer(It is used for viewing the applet)
2. Javac(It is a Java Compiler)
3. Java(It is a java interpreter)
4. Javap(Java diassembler,which convert byte code into program description)
5. Javah(It is for java C header files)
6. Javadoc(It is for creating HTML document)
7. Jdb(It is Java debugger)

For compiling and running the program we have to use following commands:

a) javac (Java compiler)

In java, we can use any text editor for writing program and then save that program with ".java" extension. Java compiler convert the source code or program in bytecode and interpreter convert ".java" file in ".class" file. Syntax:

C:\javac filename.java

If my filename is "abc.java" then the syntax will be

C:\javac abc.java

b) java(Java Interpreter)

As we learn that, we can use any text editor for writing program and then save that program with ".java" extension. Java compiler convert the source code or program in bytecode and interpreter convert ".java" file in ".class" file.
Syntax:
C:\java filename

If my filename is abc.java then the syntax will be

C:\java abc

1.8 SIMPLE JAVA PROGRAM:

```
class FirstProgram
{
        public static void main(String args[])
        {
        System.out.println ("This is my first program");
        }
}
```

The file must be named "FirstProgram.java" to equivalent the class name containing the main method.
Java is case sensitive. This program defines a class called "FirstProgram".
A class is an object oriented term. It is designed to perform a specific task. A Java class is defined by its class name, an open curly brace, a list of methods and fields, and a close curly brace.
The name of the class is made of alphabetical characters and digits without spaces, the first character must be alphabetical.

The line “public static void main (String [] args)” shows where the program will start running. The word main means that this is the main method –
The JVM starts running any program by executing this method first.
The main method in “FirstProgram.java” consists of a single statement System.out. println ("This is my first program");
The statement outputs the character between quotes to the console.

Above explanation is about how to write program and now we have to learn where to write program and how to compile and run the program.

For this reason, the next explanation is showing the steps.

1. Edit the program by the use of Notepad.
2. Save the program to the hard disk.
3. Compile the program with the javac command.(Java compiler)
4. If there are syntax errors, go back to Notepad and edit the program.
5. Run the program with the java command.(Java Interpreter)
6. If it does not run correctly, go back to Notepad and edit the program.
7. When it shows result then stop.

1.9 SUMMARY :

In this unit, we learn the concept of Object Oriented Programming, Introduction of Java, History of Java, Features of Java, Comparison between C++ and Java, Java virtual Machine and Java Environment.

Questions and Answers:

Q.1) Explain the concept of OOPs.

Ans: refer 1.2

Q.2) Explain JVM?

Ans: refer 1.6

Q.3)Explain the features of JAVA?

Ans: refer 1.4

Q.4) Explain Difference between C++ and JAVA?

Ans: refer 1.5

2

DATA TYPES, VARIABLES AND CONSTANTS

Unit Structure

2.1 DATA TYPES:

A **data type** is a scheme for representing values. An example is int which is the Integer, a data type.
Values are not just numbers, but any manner of data that a computer can process.
The data type defines the kind of data that is represented by a variable.
As with the keyword class, Java data types are case sensitive.

There are two types of data types
- primitive data type
- non-pimitive data type

In primitive data types, there are two categories
- numeric means Integer, Floating points
- Non-numeric means Character and Boolean

In non-pimitive types, there are three categories
- classes
- arrays
- interface

Following table shows the datatypes with their size and ranges.

Data type	Size (byte)	Range
byte	1	-128 to 127
boolean	1	True or false
char	2	A-Z,a-z,0-9,etc.
short	2	-32768 to 32767
Int	4	(about) -2 million to 2 million
long	8	(about) -10E18 to 10E18
float	4	-3.4E38 to 3.4E18
double	8	-1.7E308 to 1.7E308

Fig: Datatypes with size and range

2.1.1 Integer data type:

Integer datatype can hold the numbers (the number can be positive number or negative number). In Java, there are four types of integer as follows:

- byte
- short
- int
- long

We can make ineger long by adding ‘l’ or ‘L’ at the end of the number.

2.1.2 Floating point data type:

It is also called as Real number and when we require accuracy then we can use it.

There are two types of floating point data type.
- float
- double

It is represent single and double precision numbers. The float type is used for single precision and it uses 4 bytes for storage

space. It is very useful when we require accuracy with small degree of precision. But in double type, it is used for double precision and uses 8 bytes of starage space. It is useful for large degree of precision.

2.1.3 Character data type:

It is used to store single character in memory. It uses 2 bytes storage space.

2.1.4 Boolean data type:

It is used when we want to test a particular condition during the excution of the program. There are only two values that a boolean type can hold: true and false.
Boolean type is denoted by the keyword boolean and uses only one bit of storage.

Following program shows the use of datatypes.

Program:

```
import java.io.DataInputStream;
class cc2
{
public static void main(String args[]) throws Exception
{
DataInputStream s1=new DataInputStream(System.in);
byte rollno;
int marks1,marks2,marks3;
float avg;

System.out.println("Enter roll number:");
rollno=Byte.parseByte(s1.readLine());

System.out.println("Enter marks m1, m2,m3:");
marks1=Integer.parseInt(s1.readLine());
marks2=Integer.parseInt(s1.readLine());
marks3=Integer.parseInt(s1.readLine());

avg = (marks1+marks2+marks3)/3;

System.out.println("Roll number is="+rollno);
System.out.println("Average is="+avg); }

}
```

Output:

```
C:\cc>java cc2
Enter roll number:
07
Enter marks m1, m2,m3:
66
```

77
88
Roll number is=7
Average is=77.0

2. 2 MIXING DATA TYPES:

Java allows mixing of constants and variables of different types in an expression, but during assessment it hold to very strict rules of type conversion.

When computer consider operand and operator and if operands are different types then type is automatically convert in higher type.

Following table shows the automatic type conversion.

	char	**byte**	**short**	**int**	**long**	**float**	**double**
Char	int	int	int	int	long	float	double
Byte	int	int	int	int	long	float	double
Short	int	int	int	int	long	float	double
Int	int	int	int	int	long	float	double
Long	long	long	long	long	long	float	double
Float	float	float	float	float	float	float	double
double	double	double	double	double	double	double	double

2.3 VARIABLES:

Variables are labels that express a particular position in memory and connect it with a data type.

The first way to declare a variable: This specifies its data type, and reserves memory for it. It assigns zero to primitive types and null to objects.

dataType variableName;

The second way to declare a variable: This specifies its data type, reserves memory for it, and puts an initial value into that memory. The initial
value must be of the correct data type.

dataType variableName = initialValue;

The first way to declare two variables: all of the same data type, reserves memory for each.

dataType variableNameOne, variableNameTwo;

The second way to declare two variables: both of the same data type, reserves memory, and puts an initial value in each variable.

dataType variableNameI = initialValueI, variableNameII=initialValueII;

2.3.1 Variable name:

Use only the characters ‘a’ through ‘z’, ‘A’ through ‘Z’, ‘0’ through ‘9’, character ‘_’, and character ‘$’.
A name cannot include the space character.
Do not begin with a digit.
A name can be of any realistic length.
Upper and lower case count as different characters.
A name cannot be a reserved word (keyword).
A name must not previously be in utilized in this block of the program.

2.4 CONSTANT :

Constant means fixed value which is not change at the time of execution of program. In Java, there are two types of constant as follows:

Numeric Constants
- Integer constant
- Real constant

Character Constants
- Character constant
- String constant

2.4.1 Integer Constant:

An Integer constant refers to a series of digits. There are three types of integer as follows:

a) Decimal integer
Embedded spaces, commas and characters are not alloed in between digits.
For example:
23 411
7,00,000
17.33

b) Octal integer
It allows us any sequence of numbers or digits from 0 to 7 with leading 0 and it is called as Octal integer.

For example:

011

00

0425

c) Hexadecimal integer

It allows the sequence which is preceded by 0X or 0x and it also allows alphabets from ‘A’ to ‘F’ or ‘a’ to ‘f’ (‘A’ to ‘F’ stands for the numbers ‘10’ to ‘15’) it is called as Hexadecimal integer. For example:
0x7
00X
0A2B

2.4.2 Real Constant

It allows us fractional data and it is also called as folating point constant.
It is used for percentage, height and so on.
For example:
0.0234
0.777
-1.23

2.4.3 Character Constant

It allows us single character within pair of single coute.
For example:
‘A’
‘7’
‘\’

2.4.4 String Constant

It allows us the series of characters within pair of double coute.
For example:
“WELCOME”
“END OF PROGRAM”
“BYE …BYE”
“A”

2.4.5 Symbolic constant:

In Java program, there are many things which is requires repeatedly and if we want to make changes then we have to make these changes in whole program where this variable is used. For this purpose, Java provides ‘final’ keyword to declare the value of variable as follows:
Syntax:
final type Symbolic_name=value;

For example:
If I want to declare the value of 'PI' then:

final float PI=3.1459

the condition is, Symbolic_name will be in capital letter(it shows the difference between normal variable and symblic name) and do not declare in method.

2.4.6 Backslash character constant:

Java support some special character constant which are given in following table.

Constant	Importance
'\b'	Back space
'\t'	Tab
'\n'	New line
'\\'	Backslash
'\"	Single coute
'\"'	Double coute

2.5 Comments:

A **comment** is a note written to a human reader of a program. The program compiles and runs exactly the same with or without comments. Comments start with the two characters "//" (slash slash). Those characters and everything that follows them on the same line are ignored by the java compiler. everything between the two characters "/*"and the two characters "*/" are unobserved by the compiler. There can be many lines of comments between the "/*" and the "*/".

2.6 COMMAND LINE ARGUMENTS:

Command line arguments are parameters that are supplied to the application program at the time of invoking its execution. They must be supplied at the time of its execution following the file name.

In the main () method, the args is confirmed as an array of string known as string objects. Any argument provided in the command line at the time of program execution, are accepted to the array args as its elements. Using index or subscripted entry can access the individual elements of an array. The number of element in the array args can be getting with the length parameter.

For example:

```
class Add
{
public static void main(String args[])
  {
       int a=Integer.parseInt(args[0]);
       int b=Integer.parseInt(args[1]);

       int c=a+b;
      System.out.println("Addition is="+c);
    }
}
```

output:

```
c:\javac Add.java
c:\java Add 5 2
7
```

2.7 SUMMARY:

In this unit, we learn the concept of dtata types, variable and constants with example. In constants, we gain knowledge of back slash character constant. Additionaly we study the concept of command line argument and comments which is also essential for us.

2.8 QUESTION:

1. Explain types of Datatypes with example?

   Ans: refer 2.1

2. Explain Constants with example?

   Ans: refer 2.4

3

TOKENS IN JAVA

Unit Structure

3.1 INTRODUCTION:

A Java program is basically a set of classes. A class is defined by a set of declaration statements and methods or functions. Most statements contain expressions, which express the actions carried out on information or data. Smallest indivisual thing in a program are known as tokens. The compiler recognizes them for building up expression and statements.

3.2 TOKENS IN JAVA:

There are five types of token as follows:

1. Literals
2. Identifiers
3. Operators
4. Separators

3.2.1 Literals:

Literals in Java are a sequence of characters (digits, letters and other characters) that characterize constant values to be stored in variables. Java language specifies five major types of literals are as follows:

1. Integer literals
2. Floating point literals
3. Character literals
4. String literals
5. Boolean literals

3.2.2 Identifiers:

Identifiers are programmer-created tokens. They are used for naming classes, methods, variables, objects, labels, packages and interfaces in a program. Java identifiers follow the following rules:

1. They can have alphabets, digits, and the underscore and dollar sign characters.
2. They must not start with a digit.
3. Uppercase and lowercase letters are individual.
4. They can be of any length.

Identifier must be meaningful, easily understandable and descriptive.

For example:

Private and local variables like “length”.
Name of public methods and instance variables begin with lowercase letter like “addition”

3.2.3 Keywords:

Keywords are important part of Java. Java language has reserved 50 words as keywords. Keywords have specific meaning in Java. We cannot use them as variable, classes and method. Following table shows keywords.

abstract	char	catch	boolean
default	finally	do	implements
if	long	throw	private
package	static	break	double
this	volatile	import	protected
class	throws	byte	else
float	final	public	transient
native	instanceof	case	extends
int	null	const	new
return	try	for	switch

interface	void	while	synchronized
short	continue	goto	super
assert	const		

3.2.4 Operator:

Java carries a broad range of operators. An operator is symbols that specify operation to be performed may be certain mathematical and logical operation. Operators are used in programs to operate data and variables. They frequently form a part of mathematical or logical expressions.

Categories of operators are as follows:

1. Arithmetic operators
2. Logical operators
3. Relational operators
4. Assignment operators
5. Conditional operators
6. Increment and decrement operators
7. Bit wise operators

3.2.4.1 Arithmetic operators:

Arithmetic operators are used to make mathematical expressions and the working out as same in algebra. Java provides the fundamental arithmetic operators. These can operate on built in data type of Java.

Following table shows the details of operators.

Operator	Importance/ significance
+	Addition
-	Subtraction
/	Division
*	Multiplication
%	Modulo division or remainder

Now the following programs show the use of arithmetic operators.

“+” operator in Java:

In this program, we have to add two integer numbers and display the result.

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

     int a = 6;
```

```
    int b = 3;

    System.out.println("a = " + a);
    System.out.println("b =" + b);

    int c = a + b;
    System.out.println("Addition = " + c);
  }
}
```

Output:
a= 6
b= 3
Addition=9

“-” operator in Java:

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

    int a = 6;
    int b = 3;

    System.out.println("a = " + a);
    System.out.println("b =" + b);

    int c = a - b;

    System.out.println("Subtraction= " + c);
}
}
```

Output:
a=6
b=3
Subtraction=3

“*” operator in Java:

```
Class MultiplicationInt
{
 public static void main (String args[])
 {
    int a = 6;
    int b = 3;

    System.out.println("a = " + a);
```

```
    System.out.println("b =" + b);

    int c = a * b;
    System.out.println("Multiplication= " + c);
}
}
```

Output:
a=6
b=3
Multiplication=18

“/” operator in Java:

```
Class DivisionInt
{
  public static void main (String args[])
  {
    int a = 6;
    int b = 3;

    System.out.println("a = " + a);
    System.out.println("b =" + b);

    c = a / b;
    System.out.println("division=" + c);
}
}
```

Output:
a=6
b=3
Division=3

Remainder or modulus operator (%) in Java:

```
Class Remainderoptr
{
  public static void main (String args[])
  {
    int a = 6;
    int b = 3;

    System.out.println("a = " + a);
    System.out.println("b =" + b);

    c = a % b;
    System.out.println("remainder=" + c);

  }
  }
```

Output:

a=6
b=3
Remainder= 0

When both operands in the expression are integers then the expression is called Integer expression and the opration is called Integer arithmetic.
When both operands in the expression are real then the expression is called Real expression and the opration is called Real arithmetic.

When one operand in the expression is integer and other is float then the expression is called Mixed Mode Arithmetic expression and the opration is called Mixed Mode Arithmetic operation.

As we learn the Arithmetic operation on integer data and store data in integer variable. But the following program shows the use of operators with integer data and store data in float variable.

Program: write a program to calculate average of three numbers.

```
class Avg1
{
public static void main(String args[])
{
        int a=3;
        int b=3;
        int c=4;
        int avg;
avg=a+b+c;
avg=avg/3;
System.out.println("Avg of three numbers="+avg);
}
}
```

Output:

Avg of three numbers=3

3.2.4.2 Logical operators:

When we want to form compound conditions by combining two or more relations, then we can use logical operators. Following table shows the details of operators.

Operators	Importance/ significance
||	Logical – OR
&	Logical –AND
!Logical –NOT	

The logical expression defer a value of true or false. Following table shows the truth table of Logical – OR and Logical – AND.

Truth table for Logical – OR operator:

Operand1	Operand3	Operand1 || Operand3
T	T	T
T	F	T
F	T	T
F	F	F

T - True
F - False

Truth table for Logical – AND operator:

Operand1	**Operand3**	**Operand1 && Operand3**
T	T	T
T	F	F
F	T	F
F	F	F

T – True
F – False
Now the following program shows the use of Logical operators.

```
class LogicalOptr
{

 public static void main (String args[])
 {

    boolean a = true;
    boolean b = false;

    System.out.println("a||b = " +(a||b));
    System.out.println("a&&b = "+(a&&b));
    System.out.println("a! = "+(!a));
 }
}
```

Output:
a||b = true
a&&b = false
a! = false

3.2.4.3 Relational Operators:

When evaluation of two numbers is performed depending upon their relation, assured decisions are made.
The value of relational expression is either true or false.

If A=7 and A < 10 is true while 10 < A is false.

Following table shows the details of operators.

Operator	Importance/ significance
>	Greater than
<	Less than
!=	Not equal to
>=	Greater than or equal to
<=	Less than or equal to

Now, following examples show the actual use of operators.

1) If 10 > 30 then result is false
2) If 40 > 17 then result is true
3) If 10 >= 300 then result is false
4) If 10 <= 10 then result is true

Now the following program shows the use of operators.

(1) Program 1:

```
class Reloptr1
{

 public static void main (String args[])
 {
    int a = 10;
    int b = 30;

    System.out.println("a>b = " +(a>b));
    System.out.println("a<b = "+(a<b));
    System.out.println("a<=b = "+(a<=b));
  }
 }
```

Output:

```
a>b = false
a<b = true
a<=b = true
```

(2) Program 3

```
class Reloptr3
{
  public static void main (String args[])
 {
```

```
    int a = 10;
    int b = 30;
    int c = 30;

    System.out.println("a>b  =  " +(a>b));
    System.out.println("a<b  =  "+(a<b));
    System.out.println("a<=c  =  "+(a<=c));
    System.out.println("c>b  =  " +(c>b));
    System.out.println("a<c  =  "+(a<c));
    System.out.println("b<=c  =  "+(b<=c));
  }
}
```

Output:

```
a>b  =  false
a<b  =  true
a<=c  =  true
c>b  =  true
a<c  =  true
b<=c  =  true
```

3.2.4.4 Assignment Operators:

Assignment Operators is used to assign the value of an expression to a variable and is also called as Shorthand operators.

Variable_name binary_operator = expression

Following table show the use of assignment operators.

Simple Assignment Operator	Statement with shorthand Operators
A=A+1	A+=1
A=A-1	A-=1
A=A/(B+1)	A/=(B+1)
A=A*(B+1)	A*=(B+1)
A=A/C	A/=C
A=A%C	A%=C

These operators avoid repetition, easier to read and write.

Now the following program shows the use of operators.

```
class Assoptr
{

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

```
    int a = 10;
    int b = 30;
    int c = 30;
          a+=1;
          b-=3;
          c*=7;

    System.out.println("a = " +a);
    System.out.println("b = "+b);
    System.out.println("c = "+c);

 }
}
```

Output:

a = 11
b = 18
c = 310

3.2.4.5 Conditional Operators:

The character pair ?: is a ternary operator of Java, which is used to construct conditional expressions of the following form:

Expression1 ? Expression3 : Expression3

The operator ? : works as follows:

Expression1 is evaluated if it is true then Expression3 is evaluated and becomes the value of the conditional expression. If Expression1 is false then Expression3 is evaluated and its value becomes the conditional expression.

For example:

```
A=3;
B=4;
C=(A<B)?A:B;
C=(3<4)?3:4;
C=4
```

Now the following program shows the use of operators.

```
class Coptr
{

  public static void main (String args[])
  {

     int a = 10;
     int b = 30;
     int c;
          c=(a>b)?a:b;
          System.out.println("c = " +c);
```

```
    c=(a<b)?a:b;
        System.out.println("c = " +c);

 }
}
```

Output:
c = 30
c = 10

program3: Write a program to check whether number is positive or negative.

```
class PosNeg
{
        public static void main(String args[])
        {
        int a=10;
        int flag=(a<0)?0:1;
        if(flag==1)
                System.out.println(“Number is positive”);
        else
                System.out.println(“Number is negative”);
        }
}
```

Output:
Number is positive

3.2.4.6 Increment and Decrement Operators:

The increment operator ++ adds 1 to a variable. Usually the variable is an integer type, but it can be a floating point type. The two plus signs must not be split by any character. Usually they are written immediately next to the variable.

Following table shows the use of operators.

Expression	Process	Example	end result
A++	Add 1 to a variable after use.	int A=10,B; B=A++;	A=11 B=10
++A	Add 1 to a variable before use.	int A=10,B; B=++A;	A=11 B=11
A--	Subtract 1 from a variable after use.	int A=10,B; B=A--;	A=9 B=10
--A	Subtract 1 from a variable before use.	int A=10,B; B=--A;	A=9 B=9

Now the following program shows the use of operators.

```
class IncDecOp
{
public static void main(String args[])
{
int x=1;
int y=3;
int u;
int z;
u=++y;
z=x++;
System.out.println(x);
System.out.println(y);
System.out.println(u);
System.out.println(z);
}
}
```

Output:

```
3
4
4
1
```

3.2.4.7 Bit Wise Operators:

Bit wise operator execute single bit of their operands. Following table shows bit wise operator:

Operator	Importance/ significance
|	Bitwise OR
&	Bitwise AND
&=	Bitwise AND assignment
|=	Bitwise OR assignment
^	Bitwise Exclusive OR
<<	Left shift
>>	Right shift
~	One's complement

Now the following program shows the use of operators.

(1) Program 1

```
class Boptr1
{

 public static void main (String args[])
 {
    int a = 4;
    int b = a<<3;
```

```
        System.out.println("a = " +a);

        System.out.println("b = " +b);

 }
}
```

Output:

a =4
b =16

(2) Program 3

```
Class Boptr3
{
public static void main (String args[])
 {
    int a = 16;
    int b = a>>3;
        System.out.println("a = " +a);
        System.out.println("b = " +b);
 }
}
```

Output:

a = 16
b = 3

(Please refer following table)

356	138	64	33	16	8	4	3	1
3^8	3^7	3^6	3^5	3^4	3^3	3^3	3^1	3^0

3.2.5 Separator:

Separators are symbols. It shows the separated code.they describe function of our code.

Name	use
()	Parameter in method definition, containing statements for conditions,etc.
{}	It is used for define a code for method and classes
[]	It is used for declaration of array
;	It is used to show the separate statement
,	It is used to show the separation in identifier in variable declarartion
.	It is used to show the separate package name from sub-packages and classes, separate variable and method from reference variable.

3.3 OPERATOR PRECEDENCE IN JAVA:

An arithmetic expression without any parentheses will be calculated from left to right using the rules of precedence of operators.

There are two priority levels of arithmetic operators are as follows:

(a) High priority (* / %)
(b) Low priority (+ -)

The evaluation process includes two left to right passes through the expression. During the first pass, the high priority operators are applied as they are encountered.
During the second pass, the low priority operators are applied as they are encountered.

For example:

Z=A-B/3+C*3-1
When A=10, B=13, C=3
First pass:
Z=10-(13/3) + (3*3)-1
Z=10-4+3-1

Second pass:
Z=6+3-1
Z=7
Answer is=7
Following table shows associativity of operators.

Operator	Associativity	Rank
[ ]	Left to right	1
( )	Left to right	3
.	Left to right	
-	Right to left	
++	Right to left	
--	Right to left	
!	Right to left	
~	Right to left	
(type)	Right to left	
*	Left to right	3
/	Left to right	
%	Left to right	
+	Left to right	4
-	Left to right	
<<	Left to right	5
>>	Left to right	
>>>	Left to right	

<	Left to right	6
<=	Left to right	
>	Left to right	
>=	Left to right	
Instanceof	Left to right	
==	Left to right	7
!=	Left to right	
&	Left to right	8
^	Left to right	9
|	Left to right	10
&&	Left to right	11
||	Left to right	13
?:	Right to left	13
=	Right to left	14

3.4 SUMMARY:

In this unit, we learn the cocept of tokens in java.There are 4 types of tokens as we learn:

1. Literals
2. Identifiers
3. Operators

Types of operators are:

1. Arithmetic operators
2. Logical operators
3. Relational operators
4. Assignment operators
5. Conditional operators
6. Increment and decrement operators
7. Bit wise operator

We learn these operators with example.

4. separator

4

CONTROL STRUCTURE

Unit Structure

4.1 INTRODUCTION:

In Java, program is a set of statements and which are executed sequentially in order in which they appear. In that statements, some calculation have need of executing with some conditions and for that we have to provide control to that statements. In other words, Control statements are used to provide the flow of execution with condition.

In this unit, we will learn the control structure in detail.

4.2 CONTROL STRUCTURE:

In java program, control structure is can divide in three parts:

Selection statement
Iteration statement
Jumps in statement

4.2.1 Selection Statement:

Selection statement is also called as Decision making statements because it provides the decision making capabilities to the statements.

In selection statement, there are two types:

if statement

switch statement

These two statements are allows you to control the flow of a program with their conditions.

4.2.1.1 if Statement:

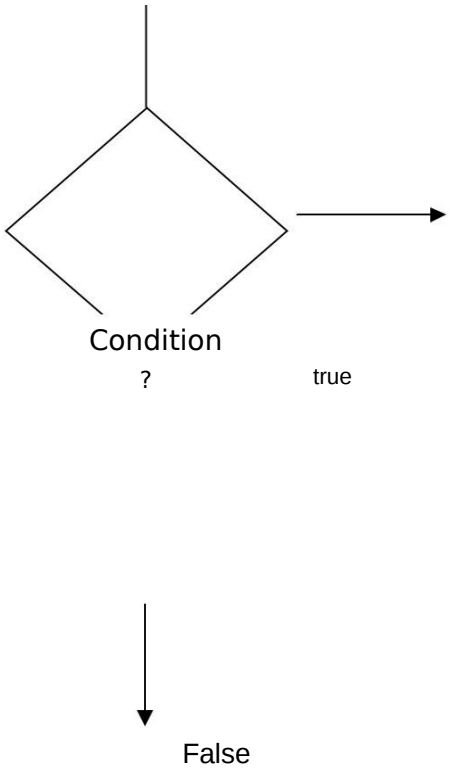
The “if statement” is also called as conditional branch statement. It is used to program execution through two paths. The syntax of “if statement” is as follows:

Syntax:

```
if (condition)
{
Statement 1;
Statement 2;
...
}
else
{
Statement 3;
Statement 4;
...
}
```

The “if statement” is a commanding decision making statement and is used to manage the flow of execution of statements. The “if statement” is the simplest one in decision statements. Above syntax is shows two ways decision statement and is used in combination with statements.

Following figure shows the “if statement”



4.2.1.1.1 Simple if statement:

Syntax:

If (condition)

{

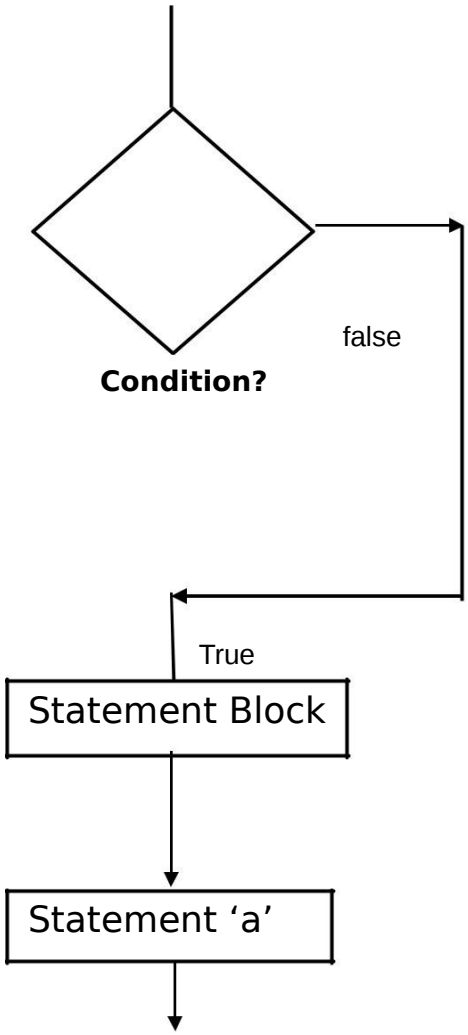
Statement block;

}

Statement-a;

In statement block, there may be single statement or multiple statements. If the condition is true then statement block will be executed. If the condition is false then statement block will omit and statement-a will be executed.

Following figure shows the flow of statement.



4.2.1.1.2 The if…else statement:

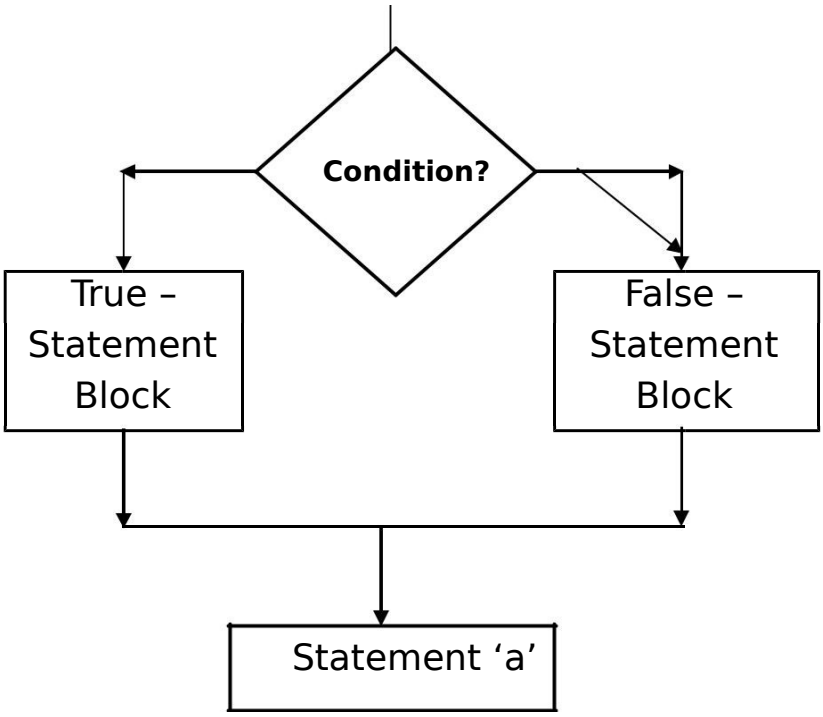
Syntax:

```
If (condition)
        {
                True - Statement block;
        }
else
        {
                False - Statement block;
        }
```

Statement-a;

If the condition is true then True - statement block will be executed. If the condition is false then False - statement block will be executed. In both cases the statement-a will always executed.

Following figure shows the flow of statement.



Following program shows the use of if statement.

Program: write a program to check whether the number is positive or negative.

```
import java.io.*;

class NumTest

{

        public static void main (String[] args) throws IOException

        {

                int Result=11;

                System.out.println("Number is"+Result);

                                if ( Result < 0 )

                {
```

```
            System.out.println("The number "+ Result +" is negative");
                    }
                    else
                    {
            System.out.println("The number "+ Result +" is positive");
                    }
                System.out.println("------- * ---------");
                }
}
```

Output:

```
C:\MCA>java NumTest
Number is 11
The number 11 is positive
------- * ---------
```

(All conditional statements in Java require boolean values, and that's what the ==, <, >, <=, and >= operators all return. A boolean is a value that is either true or false. If you need to set a boolean variable in a Java program, you have to use the constants true and false. Boolean values are no more integers than are strings).

For example: write a program to check whether the number is divisible by 2 or not.

```
import java.io.*;
class divisorDemo
{
public static void main(String[ ] args)
{
        int a =11;
        if(a%2==0)
```

```
        {
            System.out.println(a +" is divisible by 2");
        }
        else
        {
             System.out.println(a+" is not divisible by 2");
        }
}
}
```

Output:

```
C:\MCA>java divisorDemo
11 is not divisible by 2
```

4.2.1.1.3 Nesting of if-else statement:

Syntax:

```
if (condition1)
        {
        If(condition2)
                {
                        Statement block1;
                }
        else
                {
                        Statement block2;
                }
        }
        else
```

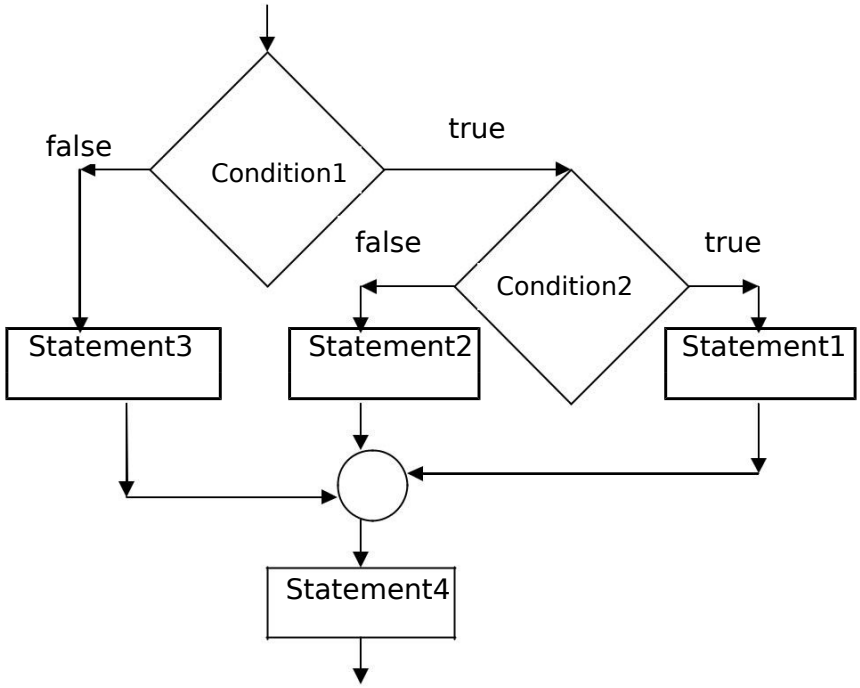
{

Statement block3;

}

Statement 4:

If the condition1 is true then it will be goes for condition2. If the condition2 is true then statement block1 will be executed otherwise statement2 will be executed. If the condition1 is false then statement block3 will be executed. In both cases the statement4 will always executed.



For example: Write a program to find out greatest number from three numbers.

```
class greatest
    {
public static void main (String args[ ])
        {
          int a=10;
          int b=20;
```

```
                int c=3;
                if(a>b)
                {
                   if(a>c)
                   {
                        System.out.println("a is greater number");
                    }
                   else
                    {
                         System.out.println("c is greater number");
                    }
                }
               else
               {
                  if(c>b)
                   {
                         System.out.println("c is greater number");
                   }
                   else
                    {
                           System.out.println("b is greater number");
                    }
               }
          }
     }
```

Output:

C:\MCA>java greatest

b is greater number

4.2.1.2 switch statement:

In Java, switch statement check the value of given variable or statement against a list of case values and when the match is found a statement-block of that case is executed. Switch statement is also called as multiway decision statement.

Syntax:

```
switch(condition)// condition means case value
{
        case value-1:statement block1;break;
        case value-2:statement block2;break;
        case value-3:statement block3;break;
        …
        default:statement block-default;break;
}
statement a;
```

The condition is byte, short, character or an integer. value-1,value-2,value-3,…are constant and is called as labels. Each of these values be matchless or unique with the statement. Statement block1, Statement block2, Statement block3,..are list of statements which contain one statement or more than one statements. Case label is always end with ":" (colon).

Program: write a program for bank account to perform following operations.

-Check balance
-withdraw amount
-deposit amount

For example:

```
import java.io.*;
class bankac
      {
       public static void main(String args[]) throws Exception
```

```
{
 int bal=20000;
 int ch=Integer.parseInt(args[0]);
 System.out.println("Menu");
 System.out.println("1:check balance");
System.out.println("2:withdraw amount... plz enter choice
and amount");
 System.out.println("3:deposit amount... plz enter choice and
amount");
 System.out.println("4:exit");
 switch(ch)
    {
     case 1:System.out.println("Balance is:"+bal);
     break;

     case 2:int w=Integer.parseInt(args[1]);
     if(w>bal)
         {
          System.out.println("Not sufficient balance");
          }
          bal=bal-w;
          System.out.println("Balance is"+bal);
          break;
          case 3:int d=Integer.parseInt(args[1]);
          bal=bal+d;
          System.out.println("Balance is"+bal);
          break;
```

```
                default:break;
            }
        }
    }
```

Output:

```
C:\MCA>javac bankac.java
C:\MCA>java bankac 1
Menu
1:check balance
2:withdraw amount... plz enter choice and amount
3:deposit amount... plz enter choice and amount
4:exit
Balance is:20000

C:\MCA>java bankac 2 2000
Menu
1:check balance
2:withdraw amount... plz enter choice and amount
3:deposit amount... plz enter choice and amount
4:exit
Balance is18000

C:\MCA>java bankac 3 2000
Menu
1:check balance
2:withdraw amount... plz enter choice and amount
```

3:deposit amount... plz enter choice and amount

4:exit

Balance is22000

C:\MCA>java bankac 4

Menu

1:check balance

2:withdraw amount... plz enter choice and amount

3:deposit amount... plz enter choice and amount

4:exit

C:\MCA>java bankac

4.2.2 Iteration Statement:

The process of repeatedly executing a statements and is called as looping. The statements may be executed multiple times (from zero to infinite number). If a loop executing continuous then it is called as Infinite loop. Looping is also called as iterations.

In Iteration statement, there are three types of operation:

for loop
while loop
do-while loop

4.2.2.1 for loop:

The for loop is entry controlled loop. It means that it provide a more concious loop control structure.

Syntax:

for(initialization;condition;iteration)//iteration means increment/ decrement

{

Statement block;

}

When the loop is starts, first part(i.e. initialization) is execute. It is just like a counter and provides the initial value of loop. But the thing is, I nitialization is executed only once. The next part(i.e. condition) is executed after the initialization. The important thing is, this part provide the condition for looping. If the condition will satisfying then loop will execute otherwise it will terminate.

Third part(i.e. iteration) is executed after the condition. The statements that incremented or decremented the loop control variables.

For example:

```
import java.io.*;

class number

{

public static void main(String args[]) throws Exception

{

int i;

System.out.println("list of 1 to 10 numbers");

for(i=1;i<=10;i++)

{

System.out.println(i);

}

}

}
```

Output:

```
C:\MCA>javac number.java

C:\MCA>java number

list of 1 to 10 numbers

1

2
```

3

4

5

6

7

8

9

10

Here we declare i=1 and then it check the condition that if i<10 then only loop will be executed. After first iteration the value of i will print and it will incremented by 1. Now the value of i=2 and again we have to check the condition and value of i will print and then increment I by 1 and so on.

4.2.2.2 while loop:

The while loop is entry controlled loop statement. The condition is evaluated, if the condition is true then the block of statements or statement block is executed otherwise the block of statement is not executed.

Syntax:

```
While(condition)
{
Statement block;
}
```

For example: Write a program to display 1 to 10 numbers using while loop.

```
import java.io.*;
class number
{
public static void main(String args[]) throws Exception
{
```

```
int i=1;
System.out.println("list of 1 to 10 numbers");
while(i<=10)
{
System.out.println(i);
i++;
}
}
}
```

Output:

```
C:\MCA>javac number.java
C:\MCA>java number
list of 1 to 10 numbers
1
2
3
4
5
6
7
8
9
10
```

4.2.2.3 do-while loop:

In do-while loop, first attempt of loop should be execute then it check the condition.

The benefit of do-while loop/statement is that we get entry in loop and then condition will check for very first time. In while loop, condition will check first and if condition will not satisfied then the loop will not execute.

Syntax:

```
do
{
Statement block;
}
While(condition);
```

In program,when we use the do-while loop, then in very first attempt, it allows us to get enter in loop and execute that loop and then check the condition.

Following program show the use of do-while loop.

For example: Write a program to display 1 to 10 numbers using do-while loop.

```
import java.io.*;
class number
{
   public static void main(String args[]) throws Exception
   {
     int i=1;
     System.out.println("list of 1 to 10 numbers");
   do
    {
            System.out.println(i);
            i++;
     }while(i<=10);
 }
```

}

Output:

list of 1 to 10 numbers

1

2

3

4

5

6

7

8

9

10

4.2.3 Jumps in statement:

Statements or loops perform a set of operartions continually until the control variable will not satisfy the condition. but if we want to break the loop when condition will satisy then Java give a permission to jump from one statement to end of loop or beginning of loop as well as jump out of a loop.

"break" keyword use for exiting from loop and "continue" keyword use for continuing the loop.

Following statements shows the exiting from loop by using "break" statement.

do-while loop:

do

{

…….

…….

if(condition)

```
{
break;//exit from if loop and do-while loop
}
……..
……..
}
While(condition);
……..
……..
```

For loop:

```
for(…….)
{
…….
……..
if(……..)
break; ;//exit from if loop and for loop
…….
…….
}
…….
……..
```

While loop:

```
while(…….)
{
```

…….

……..

if(……..)

break; ;//exit from if loop and while loop

…….

…….

}

Following statements shows the continuing the loop by using “continue” statement.

do-while loop:

do

{

……..

……..

if(condition)

{

continue;//continue the do-while loop

}

……..

……..

}

While(condition);

……..

……..

For loop:

for(…….)

{

…….

……..

if(……..)

continue ;// continue the for loop

…….

…….

}

…….

……..

While loop:

while(…….)

{

…….

……..

if(……..)

continue ;// continue the while loop

…….

…….

}

…….

…….

Labelled loop:

We can give label to a block of statements with any valid name.following example shows the use of label, break and continue.

For example:

```
Import java.io.*;
class Demo
  {
        public static void main(String args[]) throws Exception
        {
         int j,i;
         LOOP1: for(i=1;i<100;i++)
                {
                   System.out.println(“”);
                    if(i>=10)
                      {
                       break;
                       }
                       for(j=1;j<100;j++)
                          {
                             System.out.println(“$ ”);
                            if(i==j)
                              {
                                 continue LOOP1;
                               }
                          }
                }
            System.out.println(“ End of program “);
        }
 }
```

Output:

```
$
$ $
$ $ $
$ $ $ $
$ $ $ $ $
$ $ $ $ $ $
$ $ $ $ $ $ $
$ $ $ $ $ $ $ $
$ $ $ $ $ $ $ $ $
```

End of program

4.3 SUMMARY:

In this unit, we covered Selection Statement, Iteration Statement and Jump in Statement.

In Selection statement, we covered if statement and switch statement with example.

In Iteration Statement, we covered for loop, while loop and do-while loop with example.

In Jump in Statement, we covered break, continue and label with example.

5

CLASSES

Unit Structure

5.1 OBJECTIVE :

In this lesson of Java Tutorial, you will learn...

How to create class
How to create method
How to create constructor

5.2 CLASS

Definition: A class is a collection of objects of similar type. Once a class is defined, any number of objects can be produced which belong to that class.

Class Declaration

```
class classname
{
…
ClassBody
…
}
```

Objects are instances of the Class. Classes and Objects are very much related to each other. Without objects you can't use a class.

A general class declaration:

```
class name1
{
//public variable declaration
void methodname()
{
//body of method…
//Anything
}
}
```

Now following example shows the use of method.

```
class Demo
{
private int x,y,z;
public void input()
{
x=10;
y=15;
}
public void sum()
{
z=x+y;
}
public void print_data()
{
System.out.println(“Answer is =” +z);
}
public static void main(String args[])
{
Demo object=new Demo();
object.input();
object.sum();
object.print_data();
}
}
```

In program,

Demo object=new Demo();

object.input();

object.sum();

object.print_data();

In the first line we created an object.

The three methods are called by using the dot operator. When we call a method the code inside its block is executed.

The dot operator is used to call methods or access them.

5.2.1 Creating “main” in a separate class

We can create the main method in a separate class, but during compilation you need to make sure that you compile the class with the “main” method.

```
class Demo
{
private int x,y,z;
public void input() {
x=10;
y=15;
}
public void sum()
{
z=x+y;
}
public void print_data()
{
System.out.println(“Answer is =” +z);
}
}
class SumDemo
{
public static void main(String args[])
{
Demo object=new Demo();
object.input();
object.sum();
object.print_data();
}
}
```

Use of dot operator

We can access the variables by using dot operator. Following program shows the use of dot operator.

```
class DotDemo
{
int x,y,z;
public void sum(){
z=x+y;
}
public void show(){
System.out.println("The Answer is "+z);
}
}
class Demo1
{
public static void main(String args[]){
DotDemo object=new DotDemo();
DotDemo object2=new DotDemo();
object.x=10;
object.y=15;
object2.x=5;
object2.y=10;
object.sum();
object.show();
object2.sum();
object2.show();
}}
```

output :

```
C:\cc>javac Demo1.java
C:\cc>java Demo1
The Answer is 25
The Answer is 15
```

Instance Variable

All variables are also known as instance variable. This is because of the fact that each instance or object has its own copy of values for the variables. Hence other use of the "*dot"* operator is to initialize the value of variable for that instance.