UNIT-2
Know Java basics , usage of classes and objects

Introduction to Java:

Java is an object oriented programming language. The most striking feature of the language is that it is a "platform neutral" language. Java is the first programming language that is not tied to any particular hardware or operating system. Programs developed in java can be executed anywhere on any system.

2.1 Explain the history and features of java

History of Java:

- Java is a general purpose and object oriented programming language developed by "SUN Micro Systems" of USA in 1991.
- Originally it is called as "Oak" by James Gosling, one of the inventors of the language.
- This language 'oak' is renamed as java in the year 1995.
- Java was designed for the development of software for consumer electronic devices like TV"s,VCRs , toasters …… originally Gosling thought C and C++ could be used to develop the project. But the problem he faced with them is they were system dependent languages and hence could not be used on various processors. So he started developing java language, which was completely system independent.
- The main feature of java language is that it is a plat form – independent language.
- Java is the first programming language that is not tied to any particular hardware or an operating system.
- Programs developed in java can be executed anywhere on any system.
- Java language is simple, portable, highly reliable and powerful language.

Features of Java :

1. Complied and Interpreted : Usually a computer language is either complied or interpreted. Java combines both these approaches thus making java a two stage system. First java complier translates source code into what is known as byte code instructions. Byte codes are not machine instructions and in second stage, java interpreter generates machine code that can be directly executed by machine.

2. Platform-Independent and portable: The most significant contribution of java over other languages is its portability. Java programs can be easily moved from one computer system to another, anywhere and anytime. Changes and upgrades in operating system, system resources will not force any changes in java program.

Java ensures portability in two ways. First, Java complier generates byte code instructions that can be implemented on any machine. Secondly, Size of data types are machine independent.

3. Object-Oriented : Java is a true object-oriented language. Almost everything in java is an object. All program code and data reside within an objects and classes. Java comes with extensive set of classes, arranged in packages.

4. Robust :

- Robust means "strong" java programs are strong and do not crash easily like c and C++ program there are 2 reasons for this
- Java has excellent in built exception handling feature
- Memory management, in java the user need not allocate (or) de-allocate the memory everything will be taken care by JVM, only.

5. Security: Become an important issue for a language that is used for programming on internet. Threat of viruses and abuse of resources are everywhere. Java systems not only verify all memory access but also ensure

that no viruses are communicated with an Applet. The absence of pointers in java ensures that programs cannot gain access to memory locations without proper authorization.

6. Distributed: Java is designed as a distributed language for creating applications on networks. It has the ability to share both programs and data. Java applications can open and access remote object as easily as they can do in a local system. This enables multiple programmers at multiple remote locations to work on a single project.

7. Simple, Small and Familiar: Java is a small and simple language. Many features of c and c++ are not part of java. **Eg:** Java does not use pointer, preprocessor's header file, go to statements multiple inheritance. Familiarity is another feature of java, because it is modeled a C and C++ languages. Java is simplified version of C++ language.

8. Multithreaded and interactive: Multithreaded means handling multiple tasks simultaneously. Java supports multithreaded programs. This means that we need not wait for the application to finish one task before beginning another. **Eg:** listen to audio clip while scrolling a page.

9. High Performance: Java performance is impressive for an interpreted language, mainly due to use of intermediate byte code. Java architecture is designed to reduce over heads during runtime. Multithreading enhance the overall execution speed of java program.

10. Dynamic and Extensible: Java is a dynamic language. Java is capable of dynamically linking new class libraries, methods and objects. Java programs support functions written in other languages such as C and C++. These functions are "native methods". These are linked at runtime.

2.2 Define an Applet and application in java

We can develop two types of java programs.

1. Stand – alone applications 2. Web applets

Applet:

1. Applet as a small java program developed for internet applications.

2. An applet located on a distant computer [server] can be downloaded via internet and executed on a local computer [client] using a java enabled web browser or applet viewer.

3. We can develop applets for doing anything from simple animated graphics to complex games and utilities.

4. Applets are used to create dynamic web pages on the browser with the help of HTML.

5. Dynamic web pages means scrolling text, changing images, messages displayed.

6. Applets are two types. They are 1. Local Applet 2. Remote Applet

7. Local Applet is designed with in the computer.

8. Remote Applet is developed within the web browser.

Application:

An application is a program that runs on your computer under the operating system of that computer .

Differences between applications and applets

Applications	Applets
1)It has only main() function	1)It does not have any main() function
2)Running of a program is Java <space> classname	2)Running of a program is applet viewer <space> classname.html

	(or) applet viewer <space> classname.java
3)It executes the program in the normal platform	3)It executes in web browser (or) applet viewer
4)While we are using we can't use html tags	4)Here we have to use html tags
5)It does not require internet browser	5)It may be requires internet browser

2.3 Understand Byte codes of Java, JVM

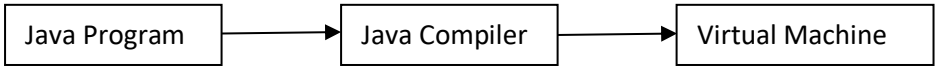
Byte codes of Java, JVM:

Java development team invented a group of instructions to express any operation. these instructions are called byte code instructions because the size of each instructions is one byte (=8 bits) exactly.

1. All language compilers translate source code into machine code for a specific computer. Java compiler also does the same thing. But the platform independent nature can be achieved by the following process.

2. Any java program can be re written using these few byte code instructions but the microprocessor cannot understand byte code instructions, so JVM will execute them.

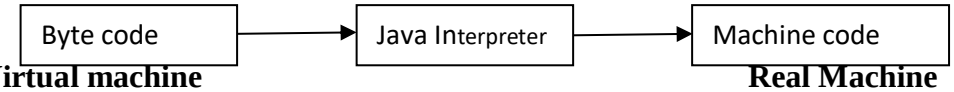
3. The following figure shows compiling a java program into byte code.



Source code **Byte code**

Fig: Process of compilation

4. The byte code is not machine specific. The machine code is generated by the java interpreter by acting an intermediary between virtual machine and the real machine.



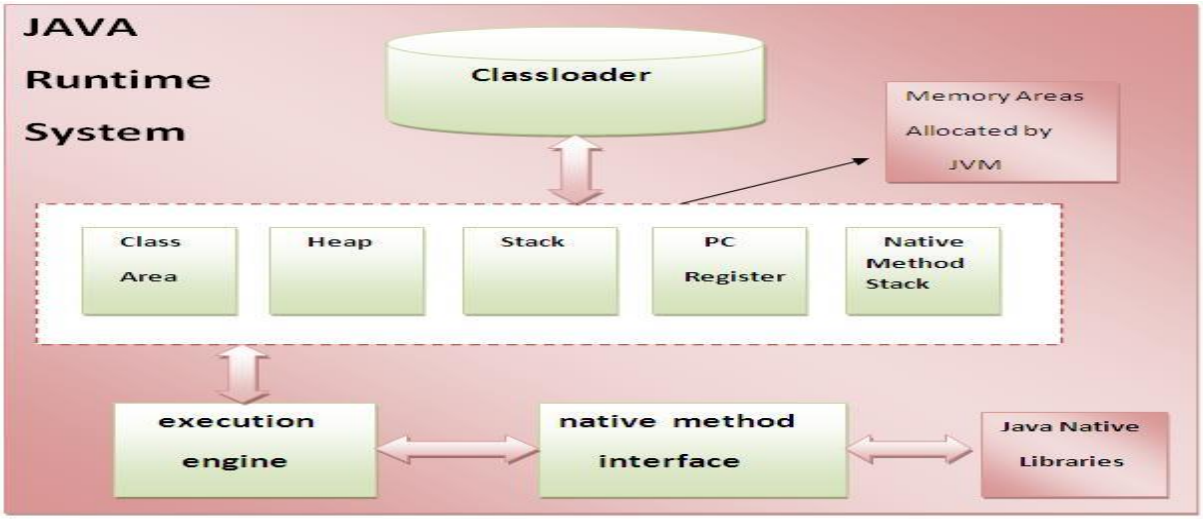
Virtual machine **Real Machine**

Fig: Process of converting byte code to machine code

Java virtual machine(Jvm):

Java virtual machine is the heart of entire java program execution process . It is responsible for taking the .class file and converting each byte code instruction into the machine language .

First of all the .java program is converted into a .class file consisting of byte code instructions by the java compiler . Now this .class file is given to the JVM



Class loader subsystem: Class loader subsystem peforms following functions

- First of all , it loads the .class file into memory .
- Then it verifies whether all byte code instructions are proper or not . If it finds any instruction suspicious the execution is rejected immediately .
- If the byte instructions are proper then it allocates necessary memory to execute .

This memory is divided into 5 parts called Run time data areas

Method area: Method area is the memory block which stores the class code(variables and methods).

Heap: This is the area where objects are created .

Java stacks: Method code is stores on method area . but while running a method it needs some more memory to store data results .This memory is allotted on java stacks .

Pc(program counter): these are the registers which contain memory address of the instructions of the methods .

Native method stacks : Java methods are executed on java stacks. Native methods are executed on Native method stacks .To execute native methods , native method libraries are required . These header files are located and connected to JVM by a program called native method interface .

Execution engine: This contains interpreter and JIT compiler , which is responsible for converting byte code instructions to machine code instructions .

2.4 Know the process of entering and executing a Java Program

Entering and executing a Java Program:

Implementation of a java application program involves a series of steps. They include

1. Creating the program(Entering the program)
2. Compiling the program
3. Executing program

Creating the program:

We can create a program using any text editor. Assume we have enter the following program

```
class Simple
{
```

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

{

System.out.println("this is sample program");

}

}
```

We must save this program in file called Sample.java. Filename is same as the class name. This file is called source file.

Compiling the program: To compile the program in Java language we must run the compiler 'javac', with name of source file on the command line at shown below.

javac Sample.java

If everything is ok, the javac compiler creates a file called 'Sample.class' contains bytecode of the program.

Executing the program: In order to execute a java program we use java interpreter to run the standalone program at the command prompt type '**java Sample'**

Now the interpreter looks for the main method in the program and begins executions from there. When executed our programs [above program] displays the following output

Output: This is sample program

2.5 Know about Comment and Keywords in Java

COMMENTS: Comments are statements in a java program are not read or interpreted by the java compiler. They are used to add more meaning to the code and to provide understanding to the developers.

Java supports the following types of comments.

1. **Single line comment:** Single line comments starts with double slash (\\) and ends at the end of line.
2. **Multiple line comments:** For longer comments we can write in multiple lines by starting with a /* and ending with */.
3. **java documentation comments**: These comments start with /** and end with */ .this comments are used to provide description for every feature in a java program and it helps in creation of html file a called API (application programming interface) document.

Ex: /** description about a class */ class code

/** description about a method */ Method

Keywords: Keyword is one which have a predefine meaning in the language. Java has 50 reserved key words. These keywords combined with operators and separators according to syntax. Keywords have specific meaning in java, we cannot use them as variable names, class name and method names.

Java keywords:

abstract	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this

break	double	implements	protected	throw
byte	else	import	public	throws
case	extends	instanceof	return	transient
catch	enum	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

2.6 Know about Unicode and naming system in java

Unicode System:

- Unicode is a universal international standard character encoding that is capable of representing most of the world's written languages.
- The Unicode standard uses hexadecimal to express a character. For example, the value 0x0041 represents the Latin character A.
- The Unicode standard was initially designed using 16 bits to encode characters because the primary machines were 16-bit PCs.

Before Unicode, there were many language standards:

- **ASCII** (American Standard Code for Information Interchange) for the United States.
- **ISO 8859-1** for Western European Language.
- **KOI-8** for Russian.
- **GB18030 and BIG-5** for chinese, and so on.

Java Soft people wanted to provide a facility to include characters not only from English but also from all other human languages to be used in java programs. This will enable the programmers to write and execute a java program in any language , which becomes an advantage on internet . This system also called Unicode system. Unicode uses a 2 bytes so that any character from any language can be encoded successfully.

Naming system in java: Naming conventions specify the rules to be followed by a java programmer while writing the names of packages , classes ,methods, etc .

- A package contains classes and interfaces . Names of packages in java are written in small letters
  Ex :- java.awt , java.io,java.applet,java.swing
- A class is a model for creating objects. An interface is also similar to a class . Each word of class names and interface names start with capital letter .
  Ex: String , BufferedReader, DataInputStream, ActionListener , RandomAcess .
- A class contains variables and methods . the first word of a method name is in small letters . then second word onwards , each new word starts with a capital letter

  Ex: println() ,readLine()

- The naming convention for variables names is same as that for methods
  Ex: age , empName

- Constants represent fixed values that cannot be altered . constants should be written in capital letters .
  Ex: PI, MAX _VAL

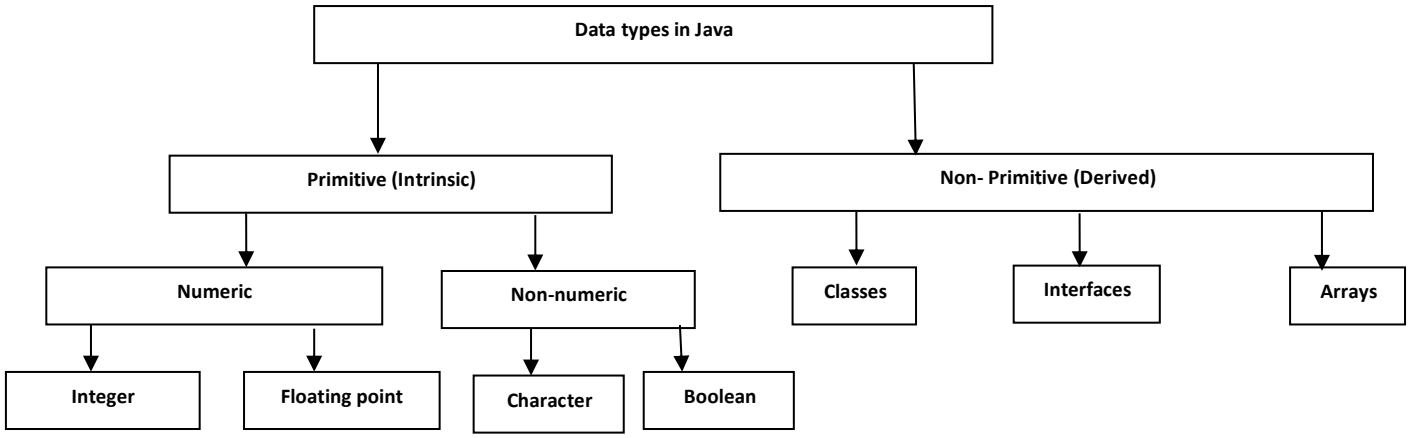
- All keywords should be written by using small letters

  Ex: public ,void , static

2.7 List and explain basic dataypes of java :

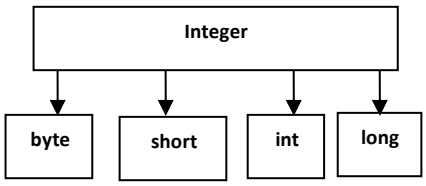
- Every variable in java has a data type.
- Data types **specify the size and type of values** that can be stored.
- A variable represents a memory location which holds data.

Data types in java under various categories



Integer types

- Integer types can hold whole numbers.
- The size of the values that can be stored depends on the integer data type we choose.
- Java supports **four types of integers** as shown in fig. they are **byte, short, int, and long.**
- Java does not support the concept of unsigned types and therefore all java values are signed meaning they can be positive or negative.
- Table shows the memory size and range(-2^{n-1} to $2^{n-1}-1$) of all the four integer data types.



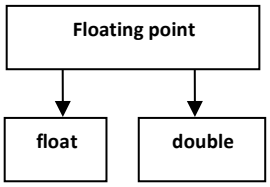
Data type	Size in memory	Minimum and maximum values
byte	1-byte	- 128 to +127
short	2-bytes	-32,768 to +32,767
int	4-bytes	-2,147,483,648 to +2,147,483,647
long	8-bytes	-9,223,372,036,854,775,808 to +9,223,372,036,854,775,808

Floating point types:

- Integer types can hold only whole numbers and therefore we use another type known as floating point type to hold numbers containing fractional parts such as 27.59 and -1.375.
- There are **two kinds of floating point** storage in java as shown in fig.
- The float type values are single-precision numbers while the double types represent double-precision numbers.
- Floating point numbers are treated as double-precision quantities. To force them to be in single-precision mode, we must append f or F to the numbers. **Example:** 1.23f, 7.56745F

- Table gives size and range of these two types.

Data type	memory size	minimum & maximum values
float	4-bytes	1.4e-045 to 3.4e+038
double	8-bytes	4.9e-0324 to 1.8e+308



Character type

- In order to store character in memory, java provides a character data type called char.
- Java uses **Unicode** to represent characters.
- The chara type assumes a size of **2 bytes** but, basically, it can hold only a single character.
- Unicode defines a fully international character set that can represent all of the characters found in human languages.

Boolean type

- Boolean type is used when we want to test a particular condition during the execution of the program.
- There are only two values that a boolean type can take: **true or false.**
- Boolean type is denoted by the keyword boolean and uses only **one bit** of storage.
- All comparision operators return boolean values.

Example program

```
import java.io.*;
class datatypes
{
public static void main(String args[])
{
boolean  t=true;
byte a=10;
short s=5;
int p=15;
long L=1550L;
float f=10.5f;
double d=10.555;
char ch='x';
String str="java";
System.out.println(t);
System.out.println(a);
System.out.println(s);
System.out.println(p);
System.out.println(L);
System.out.println(f);
System.out.println(d);
System.out.println(ch);
System.out.println(str);
}
}
```

2.8 Explain Java literals

Literals in java represent values to be stored in variables. Java language specifies five major types of literals They are

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

5. Boolean literals

1. Integer literals: These are whole numbers without any fractional part.

Java allows three types of integer literals: decimal, octal and hexadecimal.

- **Decimal integer:** An integer literal consisting of sequence of digits is taken to be decimal integer constant unless it begins with 0.
  Ex: 49,-19 etc.
- **Octal integer:** An Octal integer consists of any combination of digits from 0 to 7 with a leading 0.
  Ex: decimal 8 is written as 010_8.
- **Hexadecimal integer:** A sequence of digits proceeded by 0X or 0X is taken to be a hexadecimal integer. They may also include alphabets from A to F (or) a to f. Letters from a to f represents the numbers from 10 to 15.
  Ex: decimal 11 is written as 0XB as hexadecimal integer.

2. Floating-point literals: Float literals represent fractional numbers.They can be expressed in either standard notation or scientific notation.

Standard notation----3.14,2,46

Scinetific notation----$0.6132*10^4$

While writing these literals we can use E or e for scientific notation,

F or f for float literal and D or d for double literal.

Ex: float f1=123.4f;

3. Character literals: It consists of single character enclosed in single quotes.

Character literals include the following

- General Characters-A,b,9
- Special Characters-? @
- Escape sequences-\n,\t,\b

4. String literals: It is sequence of characters enclosed in double quotes.

Ex: "COMPUTER","999"

5. Boolean literals: Boolean literals represent only two values true and false..

2.9 Declare And Initialize Variables

- A variable is an identifier that denotes a storage location used to store a data value.
- Unlike constants that remain unchanged during the execution of a program, a variable may take different values at different times during the execution of the program.

Declaration of Variables:

- A variable must be declared before it is used in the program.
- Declaration does there things:
  1. It tells the compiler what the variable name is.
  2. It specifies what type of data the variable will hold.
  3. The place of declaration besides the scope of the variable.

The general form of declaration of variable is

Data type variable 1, variable 2,----------variable n;

- Variables are separated by commas (,) and the declaration statement ends with semicolon (;)

Ex:
```
int a, b;
Float t;
Byte b;
Boolean z;
```

Initialization: The process of giving initial values to variables is known as **initialization.**

Initialization of variables: A variable must be given a value after it has been declared but before it is used in the expression.

- This can be done in two ways
  1. By using an assignment statement
  2. By using read statement

By using an assignment statement: It is simple method to give a value to a variable.

Syntax: **Variable name=value;**

Ex: a=50;
ch='x';
d=3.20503f;

- It is also possible to declare and initialize a value in a single step.

Syntax: **Type Variable name=value;**

Ex: int a=50;
char ch='x';
float d=3.20503f;

Using read() or readLine() statement: -we may also give values to variable interactively through the keyboard using the „readLine()" method.

The readLine() method (which is invoked using an object of class DataInputStream), read the input form the keyboard as a string which can be converted to the corresponding data type using the data type wrapper class.

Accepting input from the keyboard:-A stream is required to accept input from the keyboard. A stream represents flow of data from one place to another place. Basically, there are 2- types of streams: input & output streams.

Keyboard is represented by System.in, when we write System.in, we are representing a standard input device i.e, keyboard , by default.

And, to accept data from the keyboard we need to connect it to an input stream.

Connect the keyboard to an input stream object. Here, we can use InputStreamReader that can read data from the keyboard.

InputStreamReader obj= new InputStreamReader(System.in);

Connect InputStreamReader to BufferedReader, which is another input type of stream to read data properly, coming from the stream.

BufferedReader br=new BufferedReader(obj);

Two 2-steps can be combined &rewrite in a single statement as.

BufferedReader br=new BufferedReader (new InputStreamReader(System.in));

--> we can read the data coming from the keyboard using read() & readLine() method available in BufferedReader class.

1)Accepting a single character from the keyboard: In order to accept a single character from the keyboard.

char ch= br.read();

Here the read() method reads a single character from Keyboard. But it returns it's ASCII number which is an integer, therefore this integer number can't be stored into character type variable ch , we should convert it into char type by writing (char) before the method as.

char ch=(char)br.read();

2. By using read statement: We may also give values to variable interactively through the keyboard using the 'readLine()' method.

Accepting a string from keyboard:-

String str= in.readLine();

Here, readLine() accepts a string from keyboard& returns the string into str.

Accepting an Integer value from the keyboard :

int n=Integer.parseInt(in.readLine());

Here, the readLine() method accepts the integer number from the keyboard as a string,& this should be converted into an int by using parseInt () method of integer class.

Accepting a float value from keyboard;-

float n=Float.parseFloat(in.readLine());

Accepting a Double value from keyboard;-

double n=Double.parseDouble (in.readLine ());

Accepting a byte value from keyboard;-

byte n=Byte.parseByte (in.readLine ());

Accepting a short value from keyboard;-

short n=Short .parseShort (in.readLine ());

Accepting a long value from keyboard;-

long n=Long.parseLong (in.readLine ());

Accepting a Boolean value from keyboard;-

boolean x=Boolean.parse**Boolean (in.readLine ());**

Example Program:

```
import java.io.*;;
class Addition
{
public static void main(String  args[])throws IOException
{
BufferedReader br = new  BufferedReader(new InputStreamReader(System.in))
System.out.println("enter a ,b values");
int a=Integer.parseInt(in.readLine());
int b=Integer.parseInt(in.readLine());
  int c=a+b;
System.out.println("addition of two numbers="+c);
}
}
```

Using Scanner class methods: Scanner is a pre-defined class in java.util package. By using this class also we can read the data from the keyboard.

The main advantage of this class is, we can use separate method to read each data type value directly in expected format without applying wrapper classes.

Note: there is no method to read single character in Scanner class.

Syntax:

Scanner sc=new Scanner(System.in);

To read integer then

int n= sc.nextInt();

To read float then

float f=sc.nextFloat();

like that we can read different types of data by using Scanner class methods

2.10 Perform type conversion and casting features:

It is common to assign a value of one type to a variable of another type

Automatic type conversion:-

- If the two types are compatible, then java will perform the convention automatically. This is known as **"Automatic Type conversion" (or) "Implicit casting".**
- Automatic Type conversion is possible only if the following conditions are satisfied.
  - If the 2- types are compatible
  - The destination type is large than the source type .

    Eg:- char ch='A'

    int number=ch;

Type casting (or) casting: - Explicit conversion between incompatible types is called **casting**. It is possible to convert one primitive data type into another primitive data type .this is in 2-ways.

1) Widening
2) Narrowing

Widening: - Converting a lower data type into higher type is called **Widening.**

Eg:- 1)char ch =’A’;

int num =(int) ch;// num contains 65, the ASCII values of A

2) int x=9500;
float sal=(float)x;

Features of widening:

- Widening is safe because there will **not be any loss of data** (or) precision.
- Widening of data type is also called as '**promotion of data'**
- In widening the programmer **need not use the cast operator**

Narrowing: - converting a higher data into a lower data type is called **'narrowing'.**

Ex: int n=66;
char ch=(char)n; //ch contains 'B'.

Features of Narrowing:-

- Here we will **loss some data** so narrowing is **not safe**.
  eg:- double d= 12.6789;
  int n=(int)d; //n stores only 12.
- The **programmer should use cast operator** when going for narrowing. So narrowing is also called as **demotion of data**.
- Narrowing the data type is also called as **Explicit casting**.

Example for type conversion :

```
class  AutomaticConversion
{
  int a='x';
  double d=10;
System.out.println(the value of a=+a);
System.out.println(the value of d=+d);
```

Example for typecasting:

```
class Conversion
{
        public static void main(String args[])
        {
                byte b;
                int i = 257;
                double d = 323.142;
                System.out.println("\\nConversion of int to byte.");
                b = (byte) i;
                System.out.println("i and b " + i + " " + b);
                System.out.println("\\nConversion of double to int.");
                i = (int) d;
                System.out.println("d and i " + d + " " + i);
                System.out.println("\\nConversion of double to byte.");
                b = (byte) d;
                System.out.println("d and b " + d + " " + b);
        }
}
```

2.11 Use one-dimensional and two-dimensional array

Array: Array is a group of data items of same datatype that share a common name. The advantage of using arrays is that they simplify programming by replacing a lot of statements by just one (or) 2-statements.

Type of arrays:- Arrays are generally categorized into 2-types

1) Single dimensional arrays (or) 1D arrays
2) Multi dimensional arrays (or) (2D, 3D, ….arrays).

One-dimensional arrays: A one dimensional array is **an array with only one subscript**.

For example: If we want to represent a set of five numbers by an array variable number then we may create the variable number as follows:

int number[ ]=new int[5];

Creating an array: Creation of an array involves 3 steps.
1. Declaring the array
2. Creating memory locations
3. Putting values into the memory locations

1.Declaration of arrays: arrays in java may be declared in 2 methods.

Method1----------**type arrayname[ ];** **Ex:** int number[ ];
Method2----------**type[ ] arrayname;** **Ex:** int[ ] count;
We do not enter the size of the arrays in the declaration.

2. Creation of arrays: After declaring an array we need to create it in the memory. Java allows us to create arrays **using new operator** as shown below.

Syntax: arrayname = new type[size];
Ex: number = new int[5];

It is also possible to combine the two steps declaration & creation into one as shown below.
Ex: int number[ ]=new int[5];

3.Intialization of arrays: The final step is to put values into the array created. This process is known as initialization. This is done using the array subscripts as shown below.

arrayname[subscript]=value; **Ex:** number[0]=35;…..number[4]=45;
We can also initialize arrays automatically in the same way as the ordinary variables when they are declared as shown below.
Syntax: type arrayname[ ]={list of values};
Ex: int number[ ]={1,2,3};

Array length: Arrays store the allocated size in a variable name "**length"**. We can obtain the length of the array 'a 'using a. length. **Ex**: int len=a. length;

write a java program for declaration and initialization of one dimensional array at compile time

```
class Testarray1{
public static void main(String args[])
{
 int a[]={1,2,3,4,5};//declaration, instantiation and initialization
 //printing array
for(int i=0;i<5;i++)
System.out.println(a[i]);
}
}
```

write a java program for declaration and initialization of one dimensional array at runtime

```
 import java.io.*;
import java.util.Scanner;
class Testarray2{
public static void main(String args[])throws IOException
{
 Scanner sc =new Scanner(System.in);
System.out.println("enter array length");
 int n= sc.nextInt();
int a[]= new int[n];
 System.out.println("enter array elements");
```

```
 for(int i=0;i<n;i++)
a[i]=sc.nextInt();
 System.out.println("the elements are");
for(int i=0;i<n;i++)
System.out.println(a[i]);
}
}
```

Two-dimensional arrays: Two dimensional array are the arrays with 2-indexes (or) subscripts called row & columns.(or) A two dimensional array represents several rows & columns of data.

Creating two-dimensional array: Creation of array involves three steps.

1. Declare the array
2. Creating memory locations
3. Putting values into the memory location

1. Declaration of arrays: Array in java can be declared in two methods.

Method1---------- **type arrayname[ ][ ];**

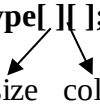
Method2---------- **type[ ][ ] arrayname;**

Form1 Example:	**Form2 Example:**
int marks[ ][ ];	int[ ][ ] marks;
float avg[ ][ ];	float[ ][ ] avg;

In declaration should not enter the size of an array.

2. Creation of arrays (allocating memory): Java allows to create arrays **using "new" operator**

Syntax: array name=new type[ ][ ];



Ex: marks = new int[3][5];

- We can write declaration and initialization in a single step.
  Syntax: type array name=new type[ ][ ]; **Ex:** int marks[][]=new int[3][5];

3. Initialization of array: it can be done by using

Syntax: type arrayname[ ][ ] = { {list of values}, {list of values} };

Ex: intmarks[2][3]={40,70,60,28,47,50};

(or)

int marks[ ][ ]={{40,70,60},{28,47,50}};

Example: write a java program for declaration and initialization of one dimensional array at compile time

```
class Testarray3
{
public static void main(String args[])
{
  int arr[][]={{1,2,3},{2,4,5},{4,4,5}};
  for(int i=0;i<3;i++)
 for(int j=0;j<3;j++)
   System.out.print(arr[i][j]+"\t ");
 }
 }
```

write a java program for declaration and initialization of one dimensional array at runtime

```
import java.util.Scanner;
class JavaProgram
```

```
{
  public static void main(String args[])
  {
     int m, n, i, j;
     int arr[][] = new int[10][10];
     Scanner scan = new Scanner(System.in);

  System.out.print("Enter Number of Rows ");
     m = scan.nextInt();
     System.out.print("Enter Number of Columns ");
     n = scan.nextInt();
   System.out.print("Enter  Array Elements : ");
     for(i=0; i<m; i++)
     {
        for(j=0; j<n; j++)
        {
           arr[i][j] = scan.nextInt();
        }
     }

     System.out.print("The Array is :\n");
     for(i=0; i<m; i++)
     {
        for(j=0; j<n; j++)
        {
           System.out.print(arr[i][j]+ "   ");
        }
        System.out.println();
     }
  }
}
```

2.12 Know various types of operators.

Operators: An operator is **a symbol that tells the computer to perform certain operation**.

An operator acts on some variables, called **operands.** If an operator acts on a single acts on a single operand, it is called 'unary operator', if it acts on two variables it is called 'binary operator'.

Java operators:

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Assignment Operators
- Increment & decrement Operators
- Conditional Operators
- Bitwise Operators
- Special Operators

1. The Arithmetic Operators: These operators are **used to perform fundamental arithmetic operations** like addition, subtraction,... There are 5-arithmetic operators in java.

Operator	Description	Example

+	**Addition**	A + B
-	**Subtraction**	A - B
*	**Multiplication**	A * B
/	**Division**	B / A
%	**Modulus**	B % A

2. The Relational Operators: They are used for the purpose of comparing two operands. They are of 6-types

1) > greater than operator
2) > = greater than(or) equal to operator
3) < lessthan operator
4) <= lessthan (or) equal to operator
5) = = equal to
6) != Not equal to

3. TheLogical operator: These operators **used to construct compound conditions by combining 2 or more relations**. Logical operator one of 3-types

--> && logical and operator
--> || logical or operator
--> ! logical not operator

eg:- if (a=1||b==1||c==1) System .out.println ("yes");

4. The Assignment Operators: This operator is **to store some value into a variable. It** is used in 3-ways.

a) To store a value into a variable eg :- int x=5;
b) To store a value of one variable into another eg:-int x=y;
c) To store a value of expression to a variable, eg - int x=y+z-4;

Note -1) we can't use more than one variable at left hand side of.
eg:- x+y=10; //invalid.
2) We can't use a literal (or) constant at left head side of eg:- 15=x;//invaded.

Shorthand Notation:-

Syntax: v op=exp;
x=x+10; ==> x+=10;

p=p/100;==>p/=100;

There are following assignment operators supported by Java language:

Operator	**Example**
=	C = A + B will assign value of A + B into C
+=	C += A is equivalent to C = C + A
-=	C -= A is equivalent to C = C — A
*=	C *= A is equivalent to C = C * A
/=	C /= A is equivalent to C = C / A
%=	C %= A is equivalent to C = C % A

5.Increment & decrement Operators: These are used to perform addition or subtraction by 1. The increment & decrement operators are ++,--Assume integer B holds 20 then:

Operator	**Description**	**Example**
++	**Increment** — Increase the value of operand by 1	B++ gives 21
--	**Decrement** — Decrease the value of operand by 1	B—gives 19

Pre increment	**post increment:**
++x;	**x++;**
x=3;	x=3
m==++x;	m=x++;
m=4	m=3
x=4	x=4

6. Conditional Operator(or) ternary operator(? :) Conditional operator is also known as the ternary operator. Since this operator acts on 3-variables, it is called ternary operator.

Syntax:- **variable =expression1 ? expression2:expression3;**

This means, expression1 is evaluated first , if it is true then expression2 value is stored otherwise expression3 value is stored into variable.

7. Bitwise Operators: There are 7- bitwise operator in java. These operator acts on individual bit(0&1) of the operands. They act on integer data types.

1) **Complement operator (~):**it given the complement from of given number by changing O's as 1's and vice - verse.

2) **Bitwise and operator(&):-**perform 'and 'operation form on the individual bits number.

E.g. - x=10=0000 1010
y=11=0000 1011
x&y = 0000 1010

3) **Bitwise 'or' operator (|):-** this operator perform 'or' operator on the bits of the number.

E.g. - x=10=0000 1010
Y=11=0000 1011
X|y = 0000 1011

4) **Bitwise 'xor' operator(^):-**this operator perform exclusive or 'xor' operation on the bits of the number. when we have odd no. of is in the i/p bits ,we can get o/p bit as1.

E.g. - x=10=0000 1010
Y=11=0000 1011
X ^ y = 0000 1001

5) **Bitwise left shift operator(<<)**:-this operator perform 'left shift' operator on the bits of the number towards left a specified no. of positions

E.g. - x=10=0000 101 X<<2=0010 1000=40

6) **Bitwise right shift operator**(>>):-this operator perform 'right shift' operator on the bits of the number towards right a specified no. of positions .

E.g. :- if x=10 then x=10=0000 1010
X>>2 is x>>2=0000 0010 =2

8. Special Operators: Java supports some special operators.

- **instanceof Operator:** This operator allows us to determine whether the object belongs to a particular class or not.

  Syntax :- Boolean variable=object instanceof class;
  Ex:stu instanceOf student

  is true if the object person belongs to the class student; otherwise it is false.

Example program :

```
class Vehicle
{
}
class car
{
  public static void main(String args[]){
    Vehicle v1 = new Vehicle();
    boolean result =  v1 instanceof  Vehicle;
    System.out.println( result);
  }
}
```

This would produce following result:

True

- **Dot Operator(.):** The dot(.) operator is used to access the instance variables and methods of class object.
  Ex: Person1.age; - refers to the variable age
  Person1.salary(); - refers to the function salary()
  It is also used to access class and sub – packages in a package.

Priority of operator:-
1) ( ) &[ ]
2) ++,--
3) *,/,%
4) +,-
5) Relational operator
6) Boolean bitwise operators
7) Logical operator
8) Ternary operator
9) Assignment operator (=).

2.13 Know The Syntax Of Selection Statements Of Java

❖ **Control statements** The control statements are used to control the flow of execution of the program. Java contains the following types of control statements:

1- Selection Statements
2-RepetitionStatements
3- Branching Statements

Selection statements (or) decision making statements: Java language possesses such decision making capabilities and supports the following statements known as control or decision making statements.

1. if statement
2. switch statement
3. conditional operator statement

1. If Statement: The if statement may be implemented in different forms depending on the complexity of conditions to be tested.

i. simple if statement
ii. If..else statement
iii. Nested if..else statement
iv. else if ladder

i. Simple if statement: This is a control statement to execute a single statement or a block of code, when the given condition is true and if it is false then it skips **if** block and rest code of program is executed.

Syntax:
```
if(test expression)
{
        statement-block;
}
statement-x;
```

Example:
```
import java.io.*;
class Even
{
        public static void main(String args[])
        {
            int n=5;

                if(n%2= =0)
                {
                System.out.println("given number");
```

```
                    }
                }
        }
```

ii. <u>If-else Statement:</u>The **"if-else"** statement is an extension of if statement .If test expression is true,true block statements are executed otherwise false block statements are executed.In both cases control is transferred to statement-x

<u>Syntax:</u>

```
if(test expression)
{
        True-blockstatement(s);
 }
else
{
        False-block statement(s)
}
 statement-x;
```

<u>Example:</u>

```
class Biggest
{
        Public static void main(String args[])
         {
                int a=25,b=35;
                if(a>b)
                {
                        System.out.println("a is big"+a);
                }
                else
                {
                        System.out.println("b is big"+b);
                }
         }
    }
```

iii. <u>Nested If-else Statement:</u>when a series of decisions are involved, we may have to use more than one if…else statement in nested form as follows:

<u>Syntax:</u>

```
if(test condition1)
{
    if(test condition2)
     {
            <statement1>;
     }
    else
     {
            <statement2>;
    }
 }
else
 {
    <statement3>;
 }
statement-x;
```

Here if the condition1 is true then condition2 is checked. If condition2 is true the condition statements in internal if block is executed otherwise statements in else block is executed.

If condition1 is false then statement in outer else block(statement3) is executed. After this control is transfer to

statement -x.

Example:

```
Class NestingIfStmt
{       Public static void main(String args[])
        {
                int a=25,b=35,c=20;
                if(a>b)
                {
                        if(a>c)
                          {
                                System.out.println("a is big"+a);
                           }
                          else
                        {
                                System.out.println("c is big"+c);
                         }
                  }
                else
                  {
                        if(b>c)
                        {
                                System.out.println("b is big"+b);
                        }
                        else
                        {

                                System.out.println("c is big"+c);
                        }
                 }
          }
}
```

iv. Else if ladder: There is another way of putting ifs together when multipath decisions are involved. A multipath decision is a chain of ifs in which the statement associated with each else is an if.

Syntax:

```
if(condition1)
{
     <statement1>;
}
else if(condition 2)
{
     <statement2>;
}
else if(condition 3)
{
     <statement3>;
}
  :
  :
 else
 {
     <statement default>;
  }
statement-x;
```

Example:

```
class Ladder
```

```
{
        public static void main(String args[])
        {
                int per=40;
                if(per>=75)
                        {
                        System.out.println("First class with distinction");
                 }
                else if(per>=60)
                {
                        System.out.println("First class");
                 }
                else if(per>=55)
                {
                        System.out.println("Second class");
                }
                else if(per>=40)
                 {
                        System.out.println("Third class");
                 }
                else
                {
                        System.out.println("Fail");
                 }
         }
  }
```

2. Switch Statement: The keyword **"switch"** is followed by an expression that should evaluates to byte, short, char or int primitive data types ,only. In a switch block there can be one or more labeled cases. The expression that creates labels for the case must be unique. The switch expression is matched with each case label. Only the matched case is executed , if no case matches then the default statement (if present) is executed.

Syntax:

```
switch(expression)
 {
      case value-1:
           block-1
           break;
      case value-2:
           block-2
           break;
           ...
         ...
       default:
          default-block
           break;
}
statement-x;
```

Example:

```
import java.io.*;
import java.util.Scanner;
class SwitchStmt
{
public static void main(String args[])
{
int ch;
```

```
Scanner sc=new Scanner(System.in);
System.out.println("Enter your choice ");
ch=Integer.parseInt();
switch(ch)
{
case1:System.out.println("Sunday");
break;
case2:System.out.println("Monday");
break;
case3:System.out.println("Tuesday");
break;
case4:System.out.println("Wednesday");
break;
case5: System.out.println("Thursday");
 break;
case6:System.out.println("Friday");
break;
case 7:System.out.println("Saturday");
break;
default:
System.out.println("Invalid");
}
}
}
```

3.The conditional operator ? :

The general form of use of the conditional operator is as follows:

Syntax: Conditional expression ? expression1: expression2;

Example:

```
import  java.io.*;
class Selection1
{
        public static void main(String args[])
        {
            int a,b,c;
            a=25;
            b=35;
            c=(a>b)?a:b;
            System.out.println("Largest number is" +c);
        }
}
```

2.14 Know The Syntax Of Iteration Statements Of Java

Repetition Statements (or) iteration statements (or) loop statements:

The process of repeatedly executing a block of statements is known as **Looping**

Java languages provides three loop or iterative statements

1. While statement
2. Do statement
3. For statement
4. Foreach loop

1. While Statement: This is a entry-controlled looping or repeating statement. The test condition is evaluated and if the condition is true, then the body of the loop is executed. After execution of the body, the test condition is once again evaluated and if it is true, the body is evaluated once again. This process is repeated until the test conditionfails.

Syntax: while(test condition)

```
        {
            //body of the loop
        }
```

Example:

```
class Reverse
{
    public static void main(String args[])
    {
        int n=132,rev=0,rem;
        while(n>=1)
        {
            rem=n%10;
            rev=(rev*10)+rem;
            n=n/10;
        }
        System.out.println("Reverse of a given number" + rev);
    }
}
```

2. do-while Statement: Do-while looping statement is an Exit-Controlled loop statement. First do block statements are executed then the condition given in **while** statement is checked. So in this case, even the condition is false in the first attempt, do block of code is executed at least once.

Syntax:

```
do
{
    //Bodyoftheloop
}while (test condition);
```

Example:

```
class DoWhileStmt
{
    public static void main(String args[])
    {
        int i=1;
        System.out.println("odd numbers from 1 to 50");
        do
        {
            if(i%2!=0)
            System.out.println(i + "");
            i++;
        }while(i<=50);
    }
}
```

3. For Statement: This is another entry-controlled loop that provides a compact way to iterate over a range of values. From a user point of view, this is reliable because it executes the statements within this block repeatedly till the specified condition is true.

Syntax:

```
for (initialization; condition; increment or decrement)
{
    //body of the loop
}
```

Initialization: The loop is started with the value specified.

Condition: It evaluates to either 'true' or 'false'. If it is false then the loop is terminated.

Increment or decrement: After each iteration, value increments or decrements.

Example: Here num is initialized to value "1", condition is checked whether num<=10. If it is so then control goes into the loop and current value of num is printed. Now num is incremented and checked again whether num<=10.If it is so then again it enters into the loop. This process continues till num>10. It prints values 1 to10 on the screen.

```
class Factorial
{
```

```
public static void main(String args[])
{
int n=5,fact=1

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

4. For each loop: It is mainly used to traverse array or collection elements. The advantage of for-each loop is that it eliminates the possibility of bugs and makes the code more readable.

Advantage of for-each loop:

- It makes the code more readable.
- It elimnates the possibility of programming errors.

Syntax of for-each loop:

for(data_type variable : array | collection){}

Simple Example of for-each loop for traversing the array elements:

```
class ForEachExample1{
  public static void main(String args[]){
   int arr[]={12,13,14,44};

   for(int i:arr){
     System.out.println(i);
   }

 }
}
```

Output:12

```
13
14
44
```

2.15 Know the syntax of jump statements, break and continue statements

Branching Statements: Jump statements in java are used to transfer the control to another part of java program. It supports 3 jump statements. They are

1. Break statement
2. Continue statement
3. Return statement

1. Break statement: The break statement can be used in 3-way:

- inside a loop to come out of it

- inside the switch block to come out of the switch block.
- used in nested blocks to go to the end of a block.

Syntax:

```
break;          //. breaks the innermost loop or switch statement.
break label;    // breaks the outermost loop in a series of nested loops
```

Example on unlabeled break:

```
import java.io.*;
class BreakStmt
{
        public static void main(String args[])
        {
                for(int i=0;i<10;i++)
                {
                        if(i= =5)
                                break;
                        System.out.println(i);
                }
        }
}
```

Example on labeled break:

```
boolean x=true;
b1:
{
        b2:
        {
                b3:
                {
                        System.out.println("block3");
                        If(x) break b2;
                }
                System.out.println("block2");
        }
        System.out.println("block1");
}
```

o/p: block3
block 2

2.Continuestatement:

This is a branching statement that are used in the looping statements (while, do-while and for) to skip the current iteration of the loop and resume the next iteration .

Syntax: continue;

Example:

```
import java.io.*;
class ContinueStmt
{
        public static void main(String args[])
        {
                for(int i=1;i<=10;i++)
                {
                        if(i%2= =0)
                                continue;    //skip next statement if i%2 is 0
                        System.out.println(i);
                }
        }
}
```

3. Return statement: This statement is used to return a value to the caller method and terminates execution of method.

Syntax:

return;
return value;

return; //This returns nothing. So this can be used when method is declared with void return type.
return expression; //It returns the value evaluated from the expression.

Example:

```
class ret1
{
public static void main(String[] args)
{
  boolean t=true;
  System.out.println("true");
     if(true)
    {
        return;
     }
     System.out.println("false");
 }
 }
```

Output:

Before the return statement.