NIIT BIRGUNJ

Java Programming for BCA

FAST TRACK

Bijay kumar

13

Introduction to object oriented: -

Object oriented programming is a solution to a programming problem that was first seen in large program being developed in 1970s. It offers powerful model for writing computer software. Object oriented programming allows the analysis and design of an application in terms of entities or objects. So, that the process replicates the human thought process as closely as possible. In object oriented programming, code and data are merged into a single thing an object.

Basic object oriented concept: -

There are several concepts underlying object oriented technology. These are: -

1. **Object: -**An object represents an entity in the real world. Every entity in the world is known as object. Object is combining of two things properties and method.

2. **Property: -**The characteristics of an object are represented as the variables in a class and referred to as the properties or attributes of the class. A requiredcharacteristic for an object or entity when represented in a class is called property.

3. **Method: -**An action required of an object or entity when represented in a class is called method. All objects in a class perform certain common action or operations.

Following are the features supported by OOPs: -

1. **Data abstraction: -**It is a process of identifying properties and methods related to a particular entity or objects as relevant to the application.

2. **Inheritance: -**It is the properties that allow the re-use of an existing class to build a new class. The new class inherits all the behavior of the original class. The original class is called super class or the super class, is the class from which another class inherits its behavior. A class that inherits the properties and methods of another class is called sub class.

3. **Encapsulation: -**It is a process that allows selective hiding of properties and methods in a class. The advantage of encapsulation is that a class can have many properties and methods but only some of these need to be exposed to the user.

4. **Polymorphism: -**It means that the same function may behave differently on different classes. The existing objects stay the same, and any changes made are only addition to it.Using this approach a programmer is able to maintain and revise code with less error since the original object is not changed.

5. **Reusability: -**The abstraction and encapsulation of data and operations comes the aspects of reusability. All object oriented language try to make parts of programs, which are easily reusable and extensive. Programs are broken down into reusable objects. Inheritance supports reusable concept.

Introduction to java programming: -

Java is programming language introduced by **Sun Micro System** in June 1995. Java is build upon C and C+ +. It derives its syntax from C and object oriented features from C+ +.

In the year 1991, a team of engineers from **Sun Micro System**wanted to design a language which could be used for electronics devices like T.V. ,washing machine, and so on. This team includes James Gosling, Patrick Naughton, Chris Warth, ED Frank and Mike Sheridan. Though C and C+ + were available to work with, these languages are designed in such a way that the compiler is targeted for a particular CPU. And it is not possible to have compilers for every CPU and require a lot of time to create ,hence easy and cost efficient solution was needed.

This software had to be small, fast and efficient and platform independent i.e. a code that could run on different CPUs and under different environments. This led to the creation of java, which was formally known 'oak'. It was named as 'java' in 1995.

What is Java?

Java is fully object oriented program. It is internet based programming language. Java can be used to create two types of program- Application and Applets.

Application: -

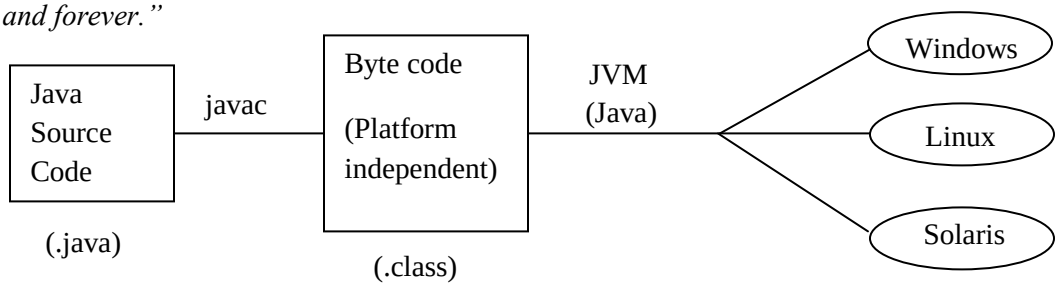
It is a program that runs on your computer under its operating system. Java application can be directly be executed by using the java interpreter.

Applet: -

An applet is a small window based program that runs on an HTML (hyper text markup language). To run java applets, you need a java enabled browser such as Internet explorer, Netscape, Navigator, hot java or an applet viewer. Java enable browser means browser that can execute a java program on your system.
Anappletis an application designed to be transmitted over the Internet and executed by aJava-compatible Web browser. An applet is actually a tiny Java program, dynamically downloaded across the network, just like an image, sound file, or video clip.

Java is both an interpreted and compiled programming language. The sources code is compiled with the help of java compiler (javac) that converts the sources code into bytes codes. The interpreter (java) executed the byte code.

The goal of java designer was to develop a language. A language whereby the programmer called *"write the code once and could run this code anywhere, anytime and forever."*



Program to print single word or single sentence.

```
class Hello
{
        public static void main (String args[ ])
        {
        System.out.println ("welcome to java programming");
        }
}
```

Some points to remember: -

1. The name of the file plays a very important role in java. The java compiler use .java extension. Since, java is case sensitive. The code must be reside in a class and the class name and file name must ne same.
2. To compile the source code, executes the compiler by using javac command.
3. To interpret the byte code using JVM (java).

Comments: -

1. Single line comment. (//): -
   // this is java program.

2. Multiple lines: -
   / *
   ……
   ……
   ***/**

public static void main (String args [ ]): -

This is main method. This is the line from where program will begin. All the java applications need to have one main method.

The *public keywords* is an access modifiers, it means that the member can be accessed from anywhere. In this case, the main method is declared as public so, that the JVM can access this method.

The *static keyword* allows the functions main to be called without the need to create extends of the class. A class cannot be access without instantiating it. But in this case, there is a copy of this method available in memory after the class is located, even if no extends of that class has been created. Hence, this method most be static and should not be dependent on extends of any class being created.

The *void keyword* tells the complier that the method does not return any value when it is executed.

The *main* ( ) is a method which performs a particular class. All java application starts from the function main. A class which does not have a main method can be successfully compiled but it cannot be executed.

String args [ ] is one of the parameter i.e. passed to the main method. Any information that you need to pass to the main method is received by the variable args which is array type.

Program to display sum.

```
class Sum
{
public static void main (String args [ ])
        {
int  A,B,C;
        A =10;
        B =20;
        C = A + B;
        System.out.println ("sum is " + C);
        }
}
```

Program to display odd and even.

```
class student
{
        public static void main (String args[ ])
        {
        int A;
        A=11;
        if (A%2= =0)
        System.out.println ("The no. is even"+A);
        else
        System.out.println ("The no. is odd"+A);
        }
}
```

What is JVM?

Virtual machine is software concepts based on the idea of an imaginary computer, which has logical set of instruction and the instruction defined the operation of this computer. The java compiler converts the source code into byte code i.e. based on instruction. An interpreter is an application that understands these streams of instruction and converts these instructions for the underline hardware to which the interpreter is targeted. JVM creates a runtime system internally that helps the execution up code by:

1. Loading the .class file
2. Managing the memory and
3. Performing the garbage collection.

Introduction to java development kit: -

Java development kit which support to run the java programs, it contains package, classes, functions which help to create a program easy way. It also contains compiler and interpreter which help to compile and interpret the java program before using , user most installed jdk. There are different versions of the jdk available in the market like jdk 1.3, jdk 1.5.5, jdk 1.6, jdk1.7and so on.

Program to accept name from user and display it.

```
import java.io.*;
class Jpro
{
public static void main (String args [ ])
{
Stringna= new String ( );
BufferedReader br = new BufferedReader (new
InputStreamReader (System.in));
try
        {
System.out.println ("Enter your name");
na= br.readLine ( );
System.out.println ("Your name is" + na);
}
catch (Exception e)
{
System.out.println ("Error");
                }
        }
}
```

Program to accept two no. from user and display sum.

```
import java.io.*;
class Jpro
{
public static void main (String args[ ])
{
int a, b, c;
BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
try
{
System.out.println ("Enter your 1st no.");
a = Integer.parseInt (br.readLine ( ));
System.out.println ("Enter your 2nd no.");
b = Integer.parseInt (br.readLine ( ));
c= a + b;
System.out.println ("Sum is" +c);
}
catch (Exception e)
{
System.out.println ("Error");
}
}
}
```

Program to accept any three no. from user and display greatest no.

```
import java.io.*;
class Jpro
{
public static void main (String args [ ])
{
int a, b, c;
BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
try
{
System.out.println ("Enter your 1st no.");
a = Integer.parseInt (br.readLine( ));
System.out.println ("Enter your 2nd no.");
b = Integer.parseInt (br.readLine( ));
System.out.println ("Enter your 3rd no.");
c = Integer.parseInt (br.readLine ( ));
if((a > b)&&(a > c))
System.out.println ("the greatest is" +a);
else if ((b >a)&&(b > c))
System.out.println ("the greatest is" +b);
else
System.out.println ("the greatest is" +c);
}
catch (Exception e)
{
System.out.println ("Error");
}
}
}
```

Program to accept empid, salary from user and display hra, ta, netsalary.

```
import java.io.*;
class Jpro
{
public static void main (String args [ ])
{
int id, sa, hra, ta, ns;
BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
Try
{
System.out.println ("Enter your id");
id = Integer.parseInt (br.readLine ( ));
System.out.println ("Enter your salary");
sa = Integer. parseInt (br.readLine( ));
hra=sal*15/100;
System.out.println ("hra is" + hra);
ta= sal*10/100;
System.out.println ("ta is"+ ta);
ns = sal + hra + ta;
System.out.println ("ns is"+ ns);
}
catch (Exception e)
{
System.out.println ("Error");
}
}  }
```

Program to accept no. from user and display their factorial.

```
import java.io.*;
class Factorial
{
public static void main (String args[ ])
{
int a, b, c;
c=1;
BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
try
{
System.out.println ("Enter Your number.");
a = Integer.parseInt (br.readLine());
for (b=1; b<=a; b++)
{
c=c*b;
}
System.out.println ("factorial is " +c);
}
catch (Exception e)
{
System.out.println ("Error");
}
}
}
```

Program to accept item code, quantity and rate from user and calculate amount.If amount is greater than 5000. Give 10% discount and display discount amount.

```
import java.io.*;
class Factorial
{
public static void main (String args[ ])
{
int  itcode, qty, rate, total, discount,ntot;
BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
try
{
System.out.println ("Enter your item code");
itcode = Integer. parseInt (br.readLine( ));
System.out.println ("Enter your quantity");
qty = Integer.parseInt (br.readLine( ));
System.out.println("Enter your rate");
rate = Integer.parseInt (br.readLine ( ));
total=qty*rate;
System.out.println ("total is" +total);
if (total>5000)
{
discount =total*10/100;
System.out.println ("Discount is"+discount);
}
else
{
discount=0;
System.out.println ("Discount is"+discount);
}
ntot=total-discount;
System.out.println ("net total is" +ntot);
}
catch (Exception e)
{
System.out.println ("Error");
}}}
```

Data types: -

A programming language is designed to process certain kinds of data consisting of numbers, characters, and string and to provide the required output. The task of processing data its executed by a sequence of instructions known as program. These instructions are formed using certain symbols and words according to rules known as syntax rules. Every program instruction must follow syntax rules of the language.

Every language has its own grammar. Like for example, java has its own data types, syntax to help create a program.

Data types indicate which type of value assigned into the variable. Like example, int, float, char etc. java has its rule in C and C+ +. Like C and C+ + , java also has primitive data types which is also known as built in data type. Java's notation of *"write ones, run anywhere"* is hidden in its data types implementation. Data types supported by this language are implemented in the same format across all the platform.The following data types support by java.

Data types	**Size in bits**
1. Byte	8
2. Char	16
3. Boolean	1
4. Short	16
5. Int	32
6. Long	64
7. Float	32
8. Double	64

Variable: -

A variable is a value that can change as necessary during the execution of a program; they are represented by symbolic names. The value of variable changes whenever a new value is assigned to it. Every variable has three characteristics; *name, initial value, scope.*

Name of variable is called as identifier. Whenever a variable is declared, it is either assigned a value or it holds a default value. A variable has scope, which determines its availability in the different sections of the program and its life time syntax for declaring variable is;

Data type identifier = [value];

Example, int a = 0; or, int a;

Some important rules to remember while declaring the variable;

1. The variable has to begin with a letter or dollar sign (~). The remaining characters can be letters, digits, dollar sign, or underscore etc.
2. Variable can contain only two special characters i.e. underscore and dollar sign. All other special characters are not allowed.
3. Variable cannot contain space.
4. Java is case sensitive, so the variable 'a' is different from 'A'.

Operators: -

Java has rich set of operators. This set can be divided into some subsets;

1. Arithmetic operator
2. Bitwise operator
3. Relational operator
4. Logical operator
5. Assignment operator
6. Increment / Decrement operator

Bitwise operator: -

Several bitwise operators are provided in the java language. These operators are used with int; short, long, byte, char, and data types. Following are the various bitwise operators.

Operator	**Meaning**	**Example**

~	Bitwise unary NOT	B = ~ a
&	Bitwise AND	C = a & b
||	Bitwise OR	C = a| b
>>	Shift right	B = a>>2
<<	Shift left	B = a<<1

Program using AND, Greater, and Equal operator.

```
class Mm
{
public static void main (String args [ ])
{
int a, b, c;
a =5;
b =10;
c= a+ b;
System.out.println ("Sum is" +c);
if (a>b&& a>c)
{
System.out.println ("The greatest is" +a);
}
else if (b>a && b>c){
System.out.println ("The greatest is" +b); }
else{
System.out.println ("The greatest is" +c);}}}
```

Ternary operator (? :): -

Ternary operator can replace an if-else construct. The syntax of this operator is

Condition?expression 1: expression 2

Is true

If false

Program of ternary operator.

```
class student
{
        public static void main (String args[])
        {
        int sal, day = 30;
        sal=day == 20? 2000: 3000;
        System.out.println ("your salary is" + sal);
        }
}
```

Assignment operator (=): -

Assignment operator (=) assign values to a variable. We can assign value more than one variable at a time. For example,

```
int a=5;
int b, c, d;
b= c = d =0;
```

Operator precedence: -

All these operators have some order of precedence. This order of precedence besides which operator is executed first when an expression contains multiple operators. The following table shows the order of precedence of operator that is commonly used.

Operator	Associatively
1. ( ) [ ]	Left to right
2. + + - --	Right to left
3. * / %	Left to right
4. + -	Left to right

Type casting: -

Type casting is a process in which one data type is converted into another data types. If this is not done properly, we may loss the data value.

Program of type casting.

```
classTypeca
{
        public static void main (String args[])
        {
        float a = (float) 23;
        System.out.println ("the value of a is" +a);
        float c= 34.568f;
        int b = (int) c +10;
        System.out.println ("the value of b is" +b);
        }
}
```

Relational operator: -

Those operator used to comparison of different variable and constant is referred as relational operator. There are six types of relational operators.

Symbol	Meaning
1. <	Less than
2. < =	Less than or equal to
3. >	Greater than
4. > =	Greater than or equal to
5. = =	Equal to
6. = !	Not equal to

Logical operator: -

Those operator which is use to perform different types of logical operation or expression is known as logical operator. There are three types of logical operators;

1. AND: -If all the conditions are true then result will be true.
2. OR: - If any one condition is true then result will be true.
3. NOT: - If condition will be true then result will be false and vice-versa.

Arithmetic operator: -

That operator which is used to perform arithmetic calculations is referred as arithmetic operator. There are mainly five types of arithmetic operators.

Operator	Meaning
1. +	Addition
2. –	Subtraction
3. *	Multiplication
4. /	Division
5. %	Remainder

Increment / Decrement operator: -

That operator which is used to increment / decrement is known as increment / decrement operator. There are mainly two types of increment / decrement operator. They are;

i

1. **Pre increment / Decrement operator: -**

   Those operators which is use to increase / decrease the variable first later on data is displayed referred as pre increment /decrement operator. For example

   int a;
   a= 1;
   + + a;

   int a;
   a=1;
   --a;

2. **Post increment / decrement operator: -**

   Those types of operator which displays the variable value first later on it increase / decrease the variable value. For example,

   int a;
   a =1;
   a+ +

   int a;
   a= 1;
   a - -

Features of java: -

The following list support java features or make java strong program;

1. **Simple: -**

   Java was designed to be easy for the professional programmers to learn and use effectively. Java is based on C and C+ +. It inherits C from syntax and C+ + from object oriented features.

2. **Secure: -**

   Java programmer more secure them normal program when downloading. The JVM helps to download java file by using sandbox concept which prevent virus effect.

3. **Portability: -**

   Portability indicates that run the program any platform (operation system). Java support platform independent concept by using byte code.

4. **Object oriented: -**
   Java supports fully object oriented concept because it is based on class and object and supports the four features; data abstraction, inheritance, encapsulation, and polymorphism.
5. **Robust: -**
   The ability to create a robust program was given a high priority in the design of java. These are two main reasons for program failure; memory management mistake and mishandled exceptional conditions (run time error). Java solves this both problems so it is robust program.
6. **Multithreaded: -**
   Java was designed to meet the real world requirement of creating interactive networked programs. Java supports multithreaded programming, which allows users to write programs that do many things simultaneously.
7. **Architecture neutral: -**
   Java programs need not modify when operating system upgrades or processor upgrades and change in core system resources. The goal of java is *"write ones, run anywhere, anytime forever."*
8. **Interpreted and high performance: -**
   Java enables the creation of cross platform (operating system) programs by compiling into as intermediate representation called java byte code. This code can be interpreted on any system that provides a java virtual machine. Java was designed to perform well on very low power CPUs. Java runtimes system that provides this features none of the benefits of the platform independent code.
9. **Distributed and: -**
   Java is designed for the distributed environment of the internet because it handles TCP/IP protocols. Java handledthe remote objects by using RMI (remote method invocation) method.
10. **Dynamic: -**
    Java programs carry with them amounts of runtime type information i.e. used to verify and resolve access to objects at runtime. This makes it possible to dynamically link code in the safe manner.

Program of class / object.

```
class student
{
public void disp()
{
int a, b, c=0;
a=10;
b=20;
c=a+b;
System.out.println("sum is" + c);
}}
class smain
{
public static void main(String args [ ])
{
student s = new student ();
s.disp();
System.out.println("program is completed");}}
```

Program of class / object.

```
class student
{
public void disp(int a, int b )
{
intc=0;
c=a+b;
System.out.println("sum is" + c);
}}
class smain
{
```

```
public static void main(String args[])
{
student s = new student ();
s.disp(10, 20);
System.out.println("program is completed");
}}
```

Write a program to find out odd and even using parameterize method and accepted by user.

```
import java.io.*;
class student
{
public void disp (int x)
{
if (x%2= =0)
{
System.out.println (“Even”+x);}
else {
System.out.println (“Odd”+x);
}}}
class ss
{
publicstatic void main (String args[])
{
student s = new student ( );
int A;
BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
try
{
System.out.println (“Enter your number”);
A=Integer.parseInt (br.readLine());
s.disp (A);
}
catch (Exception e){
System.out.println (“Error”);}
}}
```

Accept any two no. from user and compare them which one is the greater no. by using parameterize method.

```
import java.io.*;
class Student
{
public void disp (intx, int y)
{
if (x > y)
{
System.out.println (“The greater is” +x);}
else
{
System.out.println (“The greater is” +y);
}}
class smain
{
publicstatic void main (String args[])
{
Student s = new Student ( );
int a, b;
BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
try
{
System.out.println (“Enter your number”);
a=Integer.parseInt (br.readLine());
System.out.println(“Enter your another number”);
b=Integer.parseInt (br.readLine ( ));
s.disp(a,b); }
catch (Exception e){
System.out.println (“Error”) ;}}}
```

Constructor: -

Constructor is special type of method which is called automatically when we create the object of class. It established an object to initialize itself when it is created. A constructor method has the same name as the class name and cannot use void keyword.

Example of constructor.

```
class Student
{
public Student ( )
{
System.out.println ("Hello");}
}
class smain
{
publicstatic void main (String args[])
{
Student s = new Student ( );}}
```

Program of parameterize constructor.

```
class sum
{
public sum(int x)
{
int z = 0;
z = x*x;
System.out.println ("The square value is" +z);}
}
class smain
{
publicstatic void main (String args [])
{
sum s = new sum (5);
}
}
```

Program for function return value (using return keyword).----------

```
class sum
{
int disp (int x){
int z = 0;
z = x*x;
return z;
}}
class smain
{
public static void main (String args [ ]){
sums = new sum ( );
int c = s. disp (5);
System.out.println ("the square value is" +c);
}}
```

if else construct: -

It is known as conditional control structure or selection structure because it checks for the given condition and performs a specific task depending on whether the (statement) condition is true or false.

Syntax of if else structure is;

if (condition), Statement, else, Statement

Nested if structure: -

If inside another if condition is called nested if structure.

Syntax of nested if structure is;

if (condition), Statement, else if (condition), Statement, else, Statement

Switch statement: -

A switch statement is used when multiple comparison for a condition have to be performed. It also substitutes long series of nested if else statements. Condition in the switch statement can be expression or variable. *Syntax* of switch is;

```
switch (condition)
{
case'constant value 1';
    Statements;
    break;
case 'constant value 2';
```

```
                Statements;
                break;
                ……————————
                ……————————
        default:
                Statement:
        }
```

Some important points to be remember while writing conditions and constant value is switch.

1. The result of the condition must be type compatible with the constant value given with the "case" become both are exactly matched.
2. With "case" only constant values should be given and no variable or expressions are permitted.
3. No two constant values is one switch can be same.

Program of switch for 1 to 7 days accepted by users.

```
import java.io.*;
class Mukesh
{
publicstatic void main (String args[])
{
int a;
BufferedReader br = new BufferedReader (new
InputStreamReader (System.in));
try
{
System.out.println("Enter Days Name Value
Between 1 to 7");
a = Integer.parseInt(br.readLine());
        switch (a)
{
case 1:
        System.out.println("Sunday");
         break;
case 2:
         System.out.println("Monday");
        break;
case 3:
                System.out.println("Tuesday");
                 break;
case 4:
                 System.out.println("Wednesday");
                 break;
case 5:
                 System.out.println("Thursday");
                 break;
case 6:
                System.out.println("Friday");
                 break;
case 7:
                 System.out.println("Saturday");
                 break;
default:
                 System.out.println("Invalid No");
                 break;
                }
}
                 catch (Exception e)
{
                 System.out.println("Error");
                        }
                 }
}
```

Loops: -

The process ofrepetitively executing a blocks of statement is known as looping. The statements in blocks may be executed any number of times from zero to infinitive. If the loop continues forever, it is called an

infinite loop. Java supports such looping features, which enable us to develop concise programs containing repetitive process without using unconditional branching statements.

While loop: -

While loop is also known as repetitive loops or iteration loop because it executes a set of statements till the condition evaluates to true. In while loop, first check the condition if the condition is true then statement is print otherwise exit from loop.

Syntax of while loop is;

While (condition) { statements; increment / decrement ;}

Program for example of while loop.

```
class ss
{
publicstatic void main (String args [])
{
int I = 1;
while (I < = 10)
{
System.out.println (I);
I + +;
}
}
}
```

Program to display series 1 to 10 and their sum.

```
class Mukesh
{
public static void main (String args[])
{
int I =1, s = 0;
while (I < =10)
{
System.out.println (I);
s = s + I;
I + +;}
System.out.println ("sum is" +s);}}
```

For loop: -

For loop is the compact form of while loop as it combine installation of variable, condition checking and incrementing and decrementing value of the variable for iteration in a single statement. In a for loop first check the condition then after statement is print.

Syntax of for loop is;

for (intvar; condition; increment or decrement of variable)

Example of for loop.

```
class Mm
{
public static void main (String args [] )
{
inti;
for (i=1; i< = 10; i+ +)
{
System.out.println (i) ;
}
}
}
```

Program to display 10 -3 using for loop.

```
class Mm
{
public static void main (String args [ ])
{
int I;
for (I=10;I > =3; I - -)
{
System.out.println (I);
}
}
}
```

Program to display (1, 3, 5, 7, 9).

```
class Mm
{
publicstatic void main (String args[])
{
int I;
for(I=1;I < =9; I + = 2)
{
System.out.println (I);
}
}}----------------------
```

Program to display (2, 4, 6, 8,10)

```
class Mm
{
public static void main(String args[])
-------------------------------------------
{
int I;
for(I=2;I < =10; I + = 2)
{
System.out.println (I);
}}
}-------------------------------------------
```

Program to display (1, 1, 2, 3, 5).

```
class mm
{
publicstatic void main (String args[])
{
int a=1, b;
System.out.println (a);
for (a=1; a<=3; a+ +)
{
System.out.println (a);}
if (a==3)
{
b= a+2;
System.out.println (b);}}}
```

Program for display following output by nested for loop.

```
class Mu
{
public static void main (String args [ ])
{
int I, j;
for (I =1; I <= 5; I ++)
{
for (j = 1; j< =I; j ++)
{
System.out.print (j);}
System.out.println ( );
}}}
```

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

Program for display following output by nested for loop.

```
class Mm
{
public static void main (String args [ ])
{
int I, j;
for (I =1; I < = 5; I + +)
{
for (j = 1; j< =I; j + +)
{
System.out.print (I);}
System.out.println ( );}}}
```

```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

Program for display following output by nested for loop.

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

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

Program for display following output by nested for loop.

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

```
1
1 1
1 1 1
1 1 1 1
1 1 1 1 1
```

Do – while loop: -

Do – while loop works similar to the while loop except that do – while loop executes at list once even if the condition is not true. In do –while loop first statement is print then after condition is checked. *Syntax* of do – while loop is;

Do{statements; increment / decrement ;}while (condition);

Example for do – while loop is.

```
class ss
{
public static void main(String args[])
{
int I =1;
do
{
System.out.println (I);
I + +;
}
while (I < =10);
}}
```

Method: -

A method is set of executable statements. Methods are also interface to the data of the object. Methods also help to provide a structured approach to programming. Programs can be divided into different methods, which is nothing but logical grouping of related executable statements. It also help while debugging the program as the debugger can directly jump to a particular method and make necessary correction. Methods are also called as function.

Advantage of methods: -

1. Methods are provided to give access to the data of the class. No one can directly deal with the data of an object. Access to the data is only through methods.
2. A program can be divided logically.
3. No need to repeat the same set of statements again and again as separate method can be declared which will be called as and when required.
4. Programs become easy to debug.

Program of method to display add, mul, div, in a class.

```
class student
{
int a=10, b=5, c=0, d=0, e=0;
public void add()
{
c = a + b;
System.out.println ("sum is" + c);}
public void mul ( )
{
d = a * b;
System.out.println ("mul is" + d);}
public voiddiv ( )
{
e = a / b;
System.out.println ("div is" + e);}}
class smain
{
public static void main (String args [ ])
{
student s = new student ( );
s.add ( );
s.mul ( );
s.div( );
System.out.println ("program is complete");}}}
```

Passingargument to method from parameterized method.

```
class ss
{
```

```
public static disp (int a, int b)
{
int x =0;
x = a *b;
System.out.println ("multiplication is" + x);}}
classSK {
publicstatic void main (String args [])
{
ss s = new ss ( );
s.disp (10, 20);}
}
```

Program to display odd and even from passing arguments to method.

```
class student
{
public void disp (int x)
{
if (x% 2 = = 0)
System.out.println ("even") ;
else
System.out.println ("odd");}}
class smain
{
public static void main (String args [ ]){
student s = new student ( );
s.disp (5);}}}----------
```

Program to display prime no. from passing argument to method from parameterized method.

```
class student
{
public void disp (int n)
{
int I;
if (n = =1 ||n = =2|| n==3)
System.out.println ("prime");
for (I=3; I< n-1; I + +)
if (n % I= =0){
System.out.println ("not prime"+n);
break;
}
else
{
System.out.println ("prime"+n);break;}}
class smain
{
public static void main (String args [ ])
{
student s =new student ( );
s.disp (5);}}
```

Types of method: -

1. **Over loaded method: -**

   Over loaded methods are those methods, which are in the same class and have same name but different parameter list. **OR**

   Over loaded means in a single class same name of methods but passing with different parameters.

Example of over loaded method.

```
class jpro
{
void disp ( )
{
System.out.println("using overloading method");}
void disp (int x)
{
int s = 0;
s = x * x;
System.out.println ("square value is" +s);}
void disp (int x, int y)
{
if (x > y)
System.out.println ("the greatest no. is" +x);
else
System.out.println ("the greatest no. is" +y);}}
```

```
class smain{
public static void main (String args [ ]){
jpro p = new jpro( );
p.disp ( );
p.disp (5);
p.disp (5, 10);}}
```

Program of over loading method.

```
class jpro
{
void disp ( )
{
System.out.println ("it follow the concept of over loading method") ;}
void disp (int x)
{
if (x % 2 = = 0)
System.out.println ("even is" + x);
else
System.out.println ("odd is" + x);}
public void disp (int x, int y)
{
int z = x + y;
System.out.println ("sum is" + z);}}
class smain
{
public static void main (String args [ ]){
jpro p= new jpro ( );
p.disp ( );
p.disp (5);
p.disp (5, 10) ;}}
```

Over ridden method: -

Over ridden methods are those methods, which are in super class as well as in sub class. **OR**
In an over ridden method, the method in a super class (parent class) is the same in the sub class (child class) inheritance.

Example of over ridden method is.

```
class student
{
public void disp ( )
{
System.out.println ("hello, using an over ridden method");
}
}
class ss extends student
{
public void disp ( )
{
System.out.println ("this is sub program");
super.disp ( );
}}
class smain
{
public static void main (String args[ ])
{
ss s= new ss ( );
s.disp ( );
}
}
```

Inheritance: -

Inheritance is help to inherit the properties of another entity or class. Inheritance in java is implemented by super class and sub class relationship. Super class is that class which is being inherited and sub class is that which inherits super class. When inheritance is implemented, sub class gets properties of super class plus it's on properties.

In java "extends" keyword is used to inherit a class. In java there are following different types of inheritance.

1. Single levelinheritance
2. Multi level inheritance
3. Multiple inheritance

Note: -Java does not support the multiple inheritances.

Program of single level inheritance: -

```
class jpro
{
public void show ( )
{
System.out.println("using single level inheritance");
}
}
class student extends jpro
{
public void disp (int x)
{
int z = 0;
z = x*x*x;
System.out.println ("cube value is" + z);
}
}
class smain
{
public static void main (String args [ ])
{
student s = new student ( );
s.show ( );
s.disp (5);
}}
```

Program of multilevel inheritance.

```
class jpro
{
public void show ( )
{
System.out.println("Using multilevel inheritance");}}
class student extends jpro
{
public void disp (int x){
int z = 0;
z = x*x*x;
System.out.println ("cube no is." + z);}}
class ss extends student{
public void dis (int x, int y){
int z = x + y;
System.out.println ("sum is" + z);}}
class smain
{
public static void main (String args [ ]){
ss s = new ss ( );
s.show ( );
s.disp (5);
s.dis (5, 10);}}
```

Program of constructor over loaded.

```
classshow
{
public show (int x)
{
int z = 0;
z = x*x;
System.out.println ("the square value is" + z);}
public show (int a, int b) {
int c = a + b;
System.out.println ("sum is" + c);}
public show (int d)
{
if (d%2 = = 0)
System.out.println ("even"+d);
else
System.out.println ("odd"+d);}}
class conover
{
public static void main (String args [ ])
{
show s= new show (5);
show s1=new show (5, 10);
shows2= new show (11);}}
```

---

this Keyword: -

This keyword is associated with an object. It is use to indicate current object. For example,this. a = a; indicates that variable ‘a’ of this object should assign the value of variable ‘a’ which is passed through the constructor.

The keyword super is used to indicate super class object in the same way keyword this is used to indicate current objects.

Example of this keyword.

```
class sup
{
int a;
public sup ( )
{
a = 10;
}
public void disp ( )
{
System.out.println (“the value is” + a);}}
class sub extends sup{
int a;
public sub (int a)
{
this.a = a;}
public void disp ( ){
super.disp ( )
System.out.println (“the value is” + a);}
class smain{
public static void main (String args [ ]){
sub s = new sub (5);
s.disp ( );}}
```

Program of this keyword.

```
class sup
{
int a;
a = 10;
public void dsip ( )
{
System.out.println(“the value is” + a);}}
class sub extends sup
{
int a;
{
public void disp ( ){
this.a = a;
System.out.println (“the value is” + a);
super.disp ( )
}
}
class smain
{
public static void main (String args [ ])
{
sub s = new sub (5);
s.disp ( );}}
```

Inner classes: -

Inner classes is a class embedded (add) in a outer class. A class defined in a class is called as an inner class. The class called the inner class is called as outer class.

Example of inner class.

```
class outer
{
String str;
Boolean OCA;
public outer ( ){
str = new String (“outer class variable”);
OCA = true;
System.out.println (str);
outer.inner in = new outer.inner ( );
System.out.ptintln (“Outer Class Accessible” + OCA);
System.out.println(“Inner Class Accessible” +in. INA);}
```

```
class inner{
String str;
Boolean INA;
public inner ( ){
str = new string ("Inner Class Accessible");
INA = true;
System.out.println (str);
System.out.println ("Inner Class Accessible" +INA);}}}
class indemo{
public static void main (String args [ ])
{
outer out = new outer ( );}}
```

Access modifier: -

Modifiers are keywords that give additional information or meaning to the code and classes. There are two types of modifiers;

1. Access modifier
2. Non – access modifier

The access modifiers are;

a. Public
b. Protected
c. Private

Public access modifiers: -

Features of a class are available to other classes within the same package or in a different package, only when the public access modifier is used. The public access modifier makes the class features publically available to any class.

Example of public access modifier.

1.
```
package mypackage;
public class Cal {
public double volume (int a, int b, int c)
{
return (a*b*c);
}
public int sqr (int x)
{
return (x*x);
}}
```

2.
```
import java.io.*;
import my package.Cal;
public class student{
public static void main (String args [ ]){
Cal c = new Cal ( );
int s = c.volume (5, 10, 20);
System.out.println ("volume is" +s);
int z = c.sqr (5);
System.out.println ("square volume is" + z);}}
```

Package: -

In java, to reuse the already existing code, we make use of package. In java all, re-usable code is put into package is collection of classes, interfaces and other packages. Packages are essentially a means of organizing classes together as groups;

a. Packages allow you to organize your classes into smaller units and make it easy to locate and use the appropriate class file.
b. Packages allow you to protect your classes, data and methods in a large way then a class to class basis.
c. Packages names can be used to identify your classes.

Note: -When we create package we must use package keyword to create package.

Example of package.

```
package mm;
public class cal
```

```
{
public int sum (int x, int y)
{
return (x + y);
}}
```

```
import mm.cal;
class student
{
public static void main (string args [ ]){
cal c = new cal ( );
int s = c .sum (5, 10); SOP("sum is"+s);
}}
```

import java.io.*;

Pass any two values with the comp function and display which is the greatest no. by using package.

1. package mm;

```
public class cal
{
public int comp (int x, int y)
{
if (x > y)
return (x);
else
return (y);
}
}
```

2. import java.io.*;

```
import mm.cal;
class student
{
public static void main (String args [ ])
{
cal c = new cal ( );
int s = c.comp (5, 10);
System.out.println("the value is greatest" + s);
}}
```

Program of factorial using package.

1. package mm;

```
public class cal
{
public int fact (intx){
int I, m= 0;
m = n;
for (I = n-1; I > = 1; I - -){
m = m*I;}
return (m);}}
```

2. import java.io.*;

```
import mm.cal;
class student
{
public static void main(String args[])
{
cal c = new cal ();
int s = c.fact(5);
System.out.println ("fact value is" +s);
}}
```

Abstract modifier: -

The abstract modifier can be used with classes and methods this keywords when used with class indicates that the class cannot be initiated and when it uses in method indicates that the implementation of the method most be provided in this abstract class.

Example of abstract modifier is.

```
abstract class student
{
abstract void disp ( );
}
class mark extends student{
void disp ( ){
…….
…….
}}
class emp extends student{

void disp ( ){
```

```
……————————
……————————
}}
classmain{
public static void main (String args[ ]){
mark s = new mark ( );
emp y = new emp ( );
s.disp ( );
y.disp( );}}
```

Final modifier: -

Java provides user with a unique modifier named ‘final’. A method declared ‘final’ cannot be overridden in the sub class. Variable defined as a ‘final’ is a constant (fixed). A class declared as ‘final’ cannot be sub classed (not to inherit).

Example of final modifier.

```
class Mukesh
{
public void disp (int a, int b)
{
int c = a + b;
System.out.println (“sum is” + c);
}}
class Amit extends Mukesh
{
public void dsip (int d, int e, int f)
{
int g = d*e*f;
}
}
class smin
{
public static void main (String args [ ])
{
Amit k =new Amit ( );
k. disp (2, 4, 6);
k. disp (4, 5);
}}
```

Static modifier: -

Static modifier is different kind of modifier. It can be used with a variable, a method or block of code, a static variable or method or block of code in the class is not instance specific that is it can be used with the class name and there is no need to create an object of the class to the access the static feature of the class.

Example of static modifier.

```
class student
{
static int a= 10;
int j;
student ( )
{
j = 20;
}
}
class main
{
public static void main (String args[ ])
{
System.out.println (“the value of A is” + student
.a);
student s =new student ( );
System.out.println( “the value is “ + s.j);
}
}
```

Array: -

An array is group of variables which have a same name, same data types and same size. Arrays of any type can be created and may have one or more dimensions. A specific element in an array is accessed by its index. Arrays offer a convenient means grouping related information. Arrays are categories into two types.

1. One dimensional array
2. Multi-dimensional array

One dimensional array: -

One dimensional array is essentially a list of like typed variable. To create an array you first most create an array variable of the desired type. The *syntax* is;

Type var_name [ ];

Here, type declares the base type of array. The based type determines the data type. Example, *int a [5];*

Program of one dimensional array is.

```
class aa
{
public static void main (String args [ ])
{
double nums [ ] = {10, 20, 30, 40, 50};
double res = 0;
int I;
for (I = 0; I < 5 ; I + +){
res = res + nums [I];
System.out.println("array is" + res/5);}}
```

Multi – dimensional array: -

In java, multi – dimensional arrays are actually arrays of arrays. There are couples of two arrays. For example, *int a [ ] [ ] = new int [4][5];*

Multi – dimensional array used to store the data in tabular format where row and column are defined as matrix form.

Example of multi – dimensional.

```
class demo
{
public static void main (String args[]){
int a[][]=new int[4][5];
int i, j, k=0;
for (i=0;i<4;i++){
for (j=0;j<5;j++)
a[i][j]=k;
k++;}
for (i=0;i<4;i++){
for (j=0;j<5;j++)
System.out.print (a[i][j]+"");
System.out.println () ;
}}}
```

Program of multi – dimensional array.

```
class demo
{
public static void main (String args[]){
int a[][]=new int[3][4];
int i, j, k=0;
for (i=0;i<3;i++)
for (j=0;j<4;j++){
a[i][j]=k;
k+=2;}}
for (i=0;i<3;i++){
for (j=0;j<4;j++)
System.out.print (a[i][j]+"");
System.out.println ();
}}}
```

Multi – dimensional: -

When user allocate memory for a multidimensional array user need only specific the memory for the first dimension. You can allocate the remaining dimensions separately.

Example of multi-dimensional.

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

```
int TD []=new int [4][];
TD[0]=new int[1];
TD[1]=new int[2];
TD[2]=new int[3];
TD[3]=new int[4];
int i,j,k=0;
for(i=0;i<4;i++)
for(j=0;j<i+1;j++)        {
TD[i][j]=k;
k++;}
for(i=0;i<4;i++){
for(j=0;j<i+1;j++)
System.out.print (TD[i][j]+"");
System.out.println ();     } }}
```

Exception handling (error handling): -

An exception is an abnormal condition that arises in a code sequence at runtime. In a computer language that do not support exception handling, errors must be checked & handling manually. An exception is run time error.

A java exception is an object that describes an exception. In (error) condition that has occurred in piece of code. When an exception all condition arises, an object representing that exception is created & thrown in the method that caused the error. Exception can be generated by the java runtime system, or they can be manually generated by your code. Exception thrown by java reboot to fundamental error that violet the rules of the java language or the contents of java execution environment; manually generated exception are typically used to report some error condition to the collar of a method. Java exception handling is managed through five keyword; try, catch, throw, throws and finally.

Program statements that you want to monitor (check) for exceptions are contained within a try block. If an exception occurs within the try block, it is thrown. Your code can catch this exception (using catch) and handle it in some rational manner. System generated exceptions are automatically thrown by the java runtime system. To manually through an exception, uses the keyword throw.

Any exception that is thrown out of a method must be specified as such by a throws keyword. Any code that is absolutely must be executed before a method returns is put in a finally block.***Syntax*** is;

```
try
{
// block of code
}
catch (Exception type1 ex)
{
// exception handler
}
catch (Exception type2 ex)
}
finally
{
// block of code to be executed before
// try blocks ends
}
```

Program of exception handling.

```
class Exception_Handaling
{
public static void main(String args[]){
int a,b=0,c;
a=20;
try
{
c=a/b;
System.out.println ("The no. is"+c);}
```

```
catch (Exception e)
{
System.out.println ("cannot divided by zero");}
finally
{
System.out.println ("program is completed");}}}
```

Exception types: -

All exception types are sub class of the built in class throwable. Thus, throwable is at the top of the exception class. Immediately below throw able are two sub classes that partition exceptions in to two branches;

1. **Exception: -**
   This class is used for exceptional condition that user program should catch. This also the class that you will sub class to create your own custom exception types. There is an important sub classes of exception, called runtime exception. Exceptions of this type are automatically defined for the program that you write and include things such as division by zero and invalid array indexing.

2. **Error: -**
   Error defines exceptions that are not expected to be caught under normal circumstances by your program. Exception of type error are used by java runtime system to include errors having to do with the run time environment, itself, stack over fallow is an exception of such error.

Program of using try and multiple catch block.

```
class edemo
{
public static void main(String args[])
{
try
{
int a=args.length;
System.out.println ("a="+a);
int b=42/a;
int c[]={1};
c[42]=99;
}
catch (ArithmeticException e)
{
System.out.println("Divide by zero"+e);
}
catch (ArrayIndexOutOfBoundsException e )
{
System.out.println("Array index out of bound"
+e);}}}
```

Differences between Applet and Applications;

Applet	Application
1. import java.io.*; package is used. 2. init ( ) method is used. 3. drawString ( ) is used for print. 4. Execute through browser (JVM).	1. import java. lang. io.*; package is used. 2. public static void main (String args[]) 3. System.out.println is used for print. 4. Execute through Javac (interpreter)

Applet: -

Applet is java program that executes on web pages. Actually it is the bytes code rather than the source code, that you download and then run. To use an applet, we need a java enabled browser, which will interpret the bytes code for us. It is because of this applet, that today is the top among all the programming language.

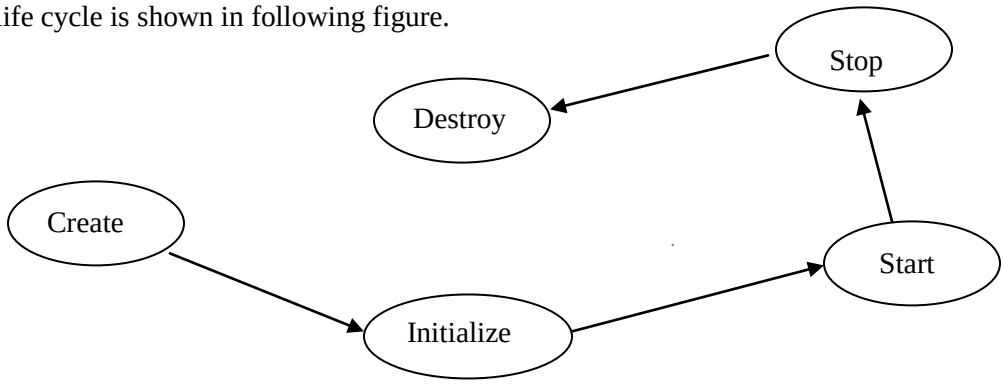
Java application are those application that rum using JVM. Java application are generally executed from a command line prompt using JDK. Applets on the other hand are executed on any browser supporting java. Internet explorer, Netscape, navigator etc are the most commonly used browsers for executing java applet. They can also be run using an applet viewer tool i.e. included in the Jdk;

Applet class: -

All applets are sun classes of applet class. The applet class is present in "java.applet" package. This is the smallest package in the java. API applet includes several methods that help in controlling the execution of applet.

Life cycle of applet: -

Java applet life cycle is shown in following figure.



Step of life cycle of applet: -

1. First an applet is created.
2. The next step is initialization. In this step init ( )method is used. This step occurs when an applet is loaded into memory. This can also include creating the objects the applet names. init ( ) method is called once the applet viewer or browser when an applet is loaded for execution.
3. Once an applet is initialized it is started. An applet can also start even if the applet was stopped previously. The difference between initialization and starting is that, an applet can be started many times but initialization can happen only once. Started method is over ridden to provide start of behavior to the applet.
4. Once the user leaves the page or the page is minimized, the stop method is called. This method performs any tasks that are required to suspend the applet execution.
5. Finally we have the destroy method. This method is over ridden to provide a cleanup behavior for the applet. This method is called when the applet is being removed from memory. This method performs any tasks that are required to destroy resources allocated to the applet.

Example of applet.

How to save applet program?

"Mukesh.html"

< Html >

<applet code = "aa.class" width = 200 height = 200 >

< / applet>

< / Html >

How to convert the following?

```
import java.awt.*;
import java.applet.*;
public class aa extends Applet
{
public void paint (Graphics g)
{
g.drawString ("welcome to Applet", 10, 50);
}
}
```

Program of sum using applet.

```
import java.awt.*;
import java.applet.*;
public class aa extends Applet
{int a, b, c;
public init ( )
{
a= 5;
b = 10;
c = a + b;
}
public void paint (Graphics g)
{
g.drawString ("the sum is"+String.valueOf(c),
10, 50);}}
```

Program of factorial using applet.

```
import java.applet.*;
public class fact extends Applet
{
int m, n, I;
public void init ( )
{
m= 1;
n = 5;
for (I = 1; I < = n; I + +)
{
m = m * I;
}
}
public void paint (Graphics g)
{
g.drawString ("the fact is"+String.valueOf(m),
10, 50) ;
}}
```

Passing parameter in applet: -

Parameters are the variable which value pass from html file to applet program. *Syntax*is;<***PARAM* >.**This indicates the parameter. It is also known as tag. Get – parameters function is used to retrieve the name of parameter.

Program of passing parameter in applet.

```
import java.awt.*;
import java.applet.*;
public class ma extends Applet{
Button b1;
public void init ( ){
String str = getParameter ("ms");
if (str = = null){
b1 = new Button ("default");}
else
b1 = new Button (str);}
add (b1);}}
```

How to savepassing parameterapplet program?

"Mukesh.html"

< Html >

<applet code = "aa.class" width = 200 height = 200 >

< PARAM NAME = ms value = "java pro" >

< / Applet >

< / Html >

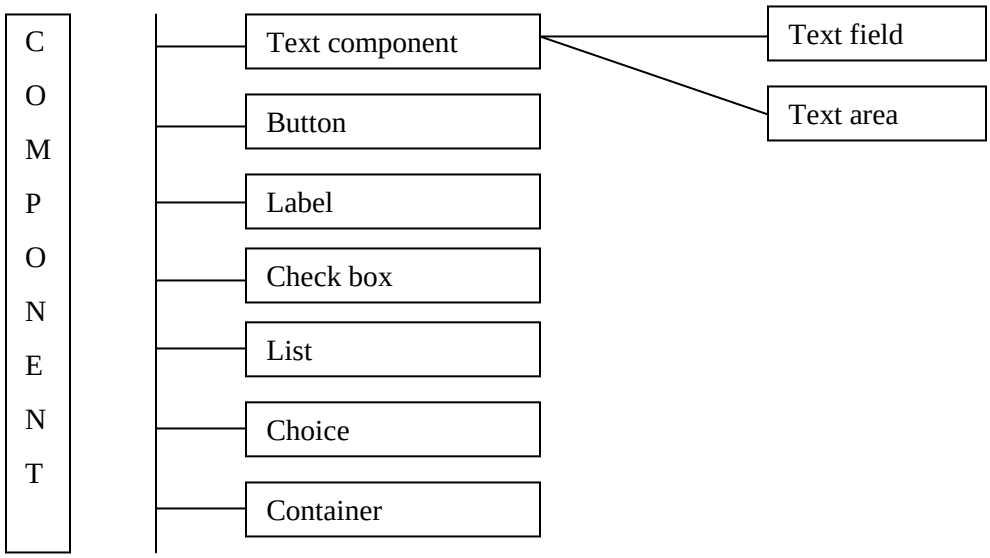
Program of user defined exception in applet.

```
import java.awt.*;
import java.applet.*;
classMexception extends Exception
{
private int d;
Mexception (int a)
{
d = a;
public String  toString ( )
{
return “Myexception [“+ dt+”]”;}}
class Edemo
{
public static void compute (int a) throws
Mexception
{
System.out.println (“called compute ( “+
at+”)”);
if (a > 10)
throw new Mexception (a);
System.out.println (“Normal Exit”);}
public static void main(String args [ ])
{
try
{
compute (1);
compute (20);
}catch (Mexception e)
{
System.out.println (“Error” +e);}}}
```

GUI (Graphic User Interface) components: -

With the help of GUI, we can create graphical interface to an application. This help to developed more efficient programs that are easy to work with. The user can interact with the GUI components.

A GUI component is an object, which is a visual object, and a user interacts with this through a mouse or keyboard. The components such as buttons, labels, check boxes, radio buttons etc. used in the application or applet can be actually seen on the screen. Any operation i.e. common to all the GUI components are found in the class component and it is supported by java.awt package.



Label: -The label is used to display a string.

java .awt.label

Example of label.

import java. awt.*;

```
import java.applet.*;
public class mLabel extends Applet
{
Label L1, L2;
public void init ( )
{
L1 = new Label ("java program");
L2 = new Label ("Awt components");
add (L1);
add (L2);
}
}
```

Test field: -Text field is used to accept the value from user and also help to calculate the value.

Example of text field.

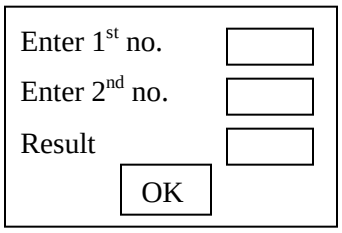
```
import java. awt.*;
import java.applet.*;
public class Tdemo extends Applet
{
Label L1, L2;
TextField T1, T2;
Button b1;
public void init ( ){
L1 = new Label ("Name");
L2 = new Label ("Address");
T1 = new TextField (20);
T2 = new TextField (20);
b1 = new Button ("ok");
add (L1);
add(T1);
add (L2);
add (T2);
add(b1);}}
```

Button class: -

Puss button are the component that are used to trigger a specific action when clicked. The text that describes the button is called label of button or caption of button. Each button should be unique and perform a particular task.

Example of button.

```
import java.awt.*;
import java.applet.*;
public class tdemo Applet
{
Label L1, L2, L3;
TextField T1, T2, T3;
Button b1;
public  void init ( ){
L1 = new Label ("Enter 1st no");
L2 = new Label ("Enter 2nd no");
L3 = new Label ("Result");
T1 = new  TextField (20);
T2 = new TextField (20);
T3 = new TestField (20);
b1 = new Button ("ok");
add (L1);
add (T1);
add (L2);
add (T2);
add (L3);
add (T3); add(b1);}}
```

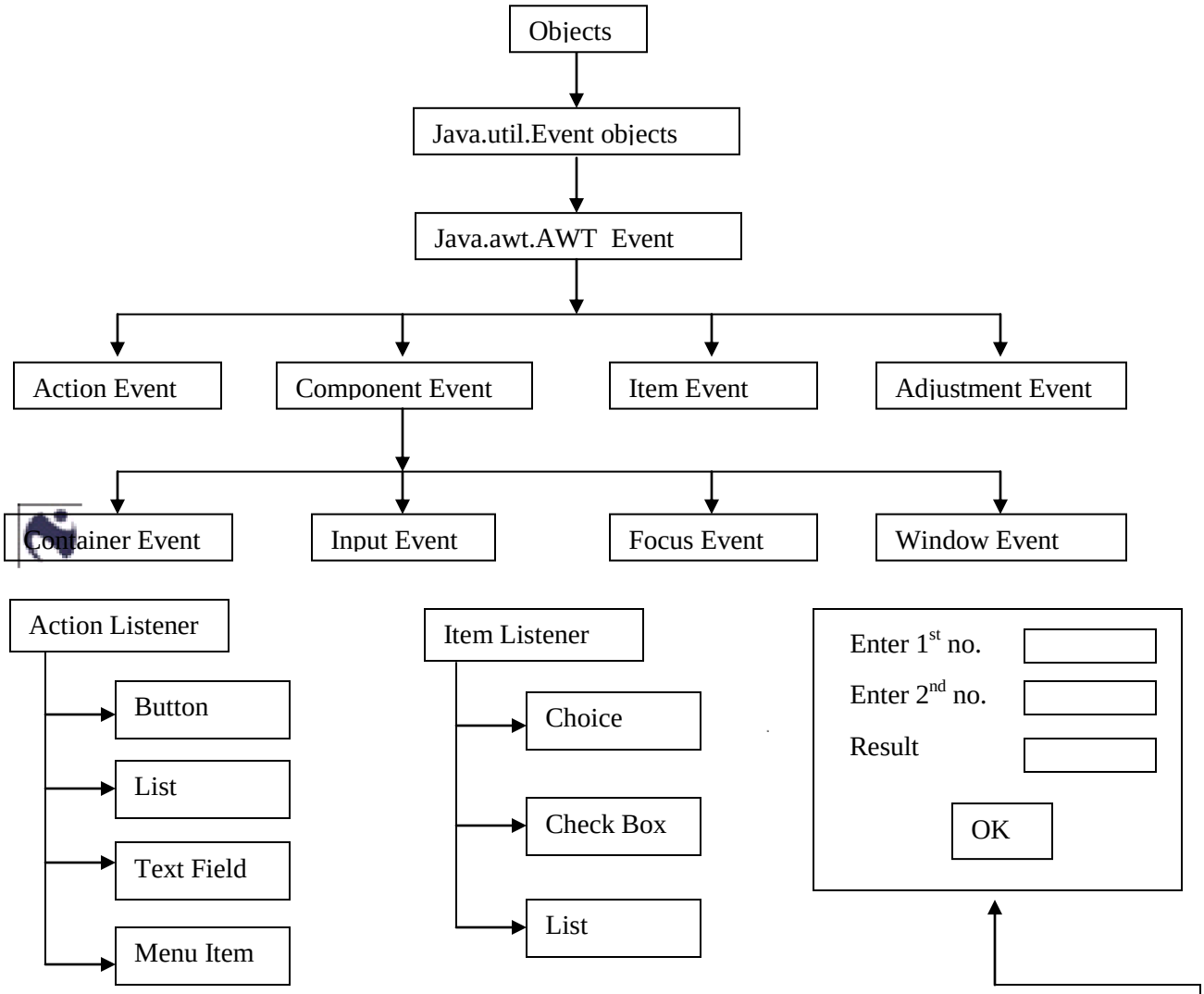


Event: -

Event performs the action. GUI system handles all the user interaction with the help an event driven model. When the user performs an action, such as moving the mouse, pressing a key, releasing the key and does on, an event is generated.

The way these events are handled depends the type of application. Some events are handle by the awt. The event processing is a process whereby, the application allows registering handlers called listeners, with the objects. The handlers are automatically called when a suitable events take place. An event listeners listen to a particular event generated by an object this in turn calls the methods call "event handler" that handle the event. Each event listeners provides methods that handle these events.

Types of event: -



Interface: -

Interface is like a class but not a class, in java the method in interface can be declared but not initialize like an abstract class. Class can be inherit with the help of extends keyword where as interfaces can be inherited by the implements keywords.

Example of interface.

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
public class aa extends Applet implements ActionListener.
{
Label L1, L2, L3;
TextField  T1, T2, T3;
Button b1;
public  void init ( ){
L1 = new Label ("Enter 1st no");
L2 = new Label ("Enter 2nd no");
L3 = new Label ("Result");
T1 = new TextField (20);
T2 = new TextField (20);
T3 = new TextField (20);
b1 = new Button ("ok");
b1.addActionListener (this);
add (L1);
```

```
add (T1);
add (L2);
add (T2);
add (L3);
add (T3);
add (b1);}}
public void actionPerformed(ActionEvent e){
if(e.getSource( ) ==b1){
int n = Integer.parseInt (T1.getText ( ));
int m = Integer.parseInt(T2.getText ( ));
int x = m + n;
T3.setText (String.valueOf (x));}}
```

Check box: -

The check box is used to create two types of components. They are; check boxes and radio buttons. Java.awt.check box package is used.

Program of check box.

```
import java.awt.*;
import java.applet.*;
public class mdemo extends Applet
{
Checkbox cb1, cb2;
Button b1;
public void init ( ){
setBackground (Color.cyan);
setBackgroud(Color.black);
b1 = new Button (“java”);
cb1 = new Checkbox (“bold”);
cb2 = new Checkbox(“Italic”);
add (b1);
add(cb1);
add(cb2);}}
```

Program to create radio button.

```
import java.awt.*;
import java.applet.*;
public class mdemo extends Applet
{
Checkbox cb1, cb2, cb3;
CheckboxGroup cb;
public void init ( ){
cb = new CheckBoxGroup ( );
cb1 = new Checkbox (“Bold”,cb,false);
cb2 = new Checkbox (“Italic”,cb,false);
cb3 = new Checkbox (“Plain”,cb,true);
add(cb1);
add(cb2) ;
add (cb3);
}}
```

Choice: -

The choice object provides a list of items from which an item can be selected by the user. It is also called dropdown list.

Example of choice.

```
import java.awt.*;
import java.applet.*;
public class mdemo extends Applet
{
Label L1;
Choice cb;
public void init ( ){
L1 = new Label (“using choice”);
cb = new Choice ( );
cb.add(“Button”);
cb.add(“Label”);
cb.add(“List”);
cb.add(“TextField”);
add (L1);
add (cb);}}
```

List: -

List is a collection of items from which the user may select one item or some timers more than one items. Java.awt.list.*; package is used.

Program of list.

```
import java.wt.*;
import java.applet.*;
        public class mdemo extends Applet
{
        Label L1;
        List co;
String str [ ] = {"Java", "C + +", "VB", "VB.Net", "Java Beans"};
        public void init ( )
{
        co = new List (2, false);
        for (int I = 0; I < str.length; I + +)
{
        co.add (str[I]);
}
        add(co);
}
}
```

Text area: -

The text area class provides on area where multiple text lines are visible and can be manipulated. Java.awt.TextArea package is used.

Example of text area is.

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
public class mdemo extends Applet{
Button b1;
TextArea T1,T2;
String msg = "This class can also \n" + "implements text area class \ n" + "in java program.\n";
public void init ( ){
B1 = new Button ("copy");
T1 = new TextArea (msg, 5, 14);
T2 = new TextArea (5, 14);
T2.setEditable (false);
B1.addActionListener (this);
add (T1);
add (B1);
add (T2);}
public void actionPerformed (Actione Event e){
str = T1.getSelectedText ( )+ "\n";
T2.setText (str);}}
```

Layout Manager: -

The Layout manager arranges the components as user required. It is physically arrangement of components is applet or frame or top level containers. All the components that we created used the default layout manager. The default layout of an Applet is the FlowLayout. The FlowLayout manager automatically arranges our components within the window by using some types of algorithm. All the components are placed in a container and are arranged with the associated layout manger. The layout manger is set with setLayout ( ).

The Layout Manager present in the java programming language are:

- ➔ FlowLayout
- ➔ BorderLayout

- GridLayout
- CardLayout
- GridBagLayout
- Null Layout

Flow Layout: -

This is the default layout manager for Panels and applets. It implements a simple layout style. The 'FlowLayout' manager arranges components in horizontal rows (left to right, top to bottom).The components arranged from upper Left corner, Left to right and top to bottom. With a no. of components, it adjusts these components row wise and left to right.

Example of flow layout is.

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
public class mm extends Applet
implements ActionListener
{
Button b1, b2, b3;
public void init ( ){
b1 = new Button ("Left alignment");
b2  = new Button ("Right");
b3 = new Button ("Centre");
b1.addActionListner (this);
b2.addActionListner (this);
b3.addActionListner (this);
add (b1);
add (b2);
add (b3);}
public void
actionPerformed(ActionEvent e)
{
int val;
if (e.getSource ( ) = = b1)
{
val = flowlayout.LEFT;}
else if (e.getSource ( ) = = b2)
{
val = flowlayout.RIGHT;}
else
{
val = flowlayout.CENTER;}
setLayout (new FlowLayout (val));
validate( );}}
```

Border layout: -

The border layout manager helps to arrange up to five components in a container. These components can be assigned to north, east, west, south and centre of the container.

a. NORTH: - Correspond to the top of the container.
b. EAST: -Right of the container.
c. SOUTH: -The button of the container.
d. WEST: -The left of the container.
e. CENTRE: -The center of the container.

Example of border layout is.

```
import java.util.*;
import java.applet.*;
import java.awt.*;
public class bdemo  extends Applet
{
Button b1, b2, b3, b4, b5;
public void init ( )
{
setBackground(Color.Red);
setForeground(Color.Blue);
setLayout (new BorderLayout (5, 5));
b1 = new Button ("East");
b2 = new Button ("West");
```

```
b3 = new Button ("North");
b4 = new Button ("South");
b5 = new Button ("Center");
add (b1, BorderLayout.EAST);
add (b2, BorderLayout.WEST);
add (b3, BorderLayout.NORTH);
add (b4, BorderLayout.SOUTH);
add (b5, BorderLayout.CENTER);}}
```

Grid layout: -

The Grid Layout helps us to divide the container into a grid. The components can be placed in rows and columns. Each grid should contain at least 1 component. All components in the layout are given equal size. It always ignores a components preferred size.

Example of grid layout is.

```
import java.util.*;
import java.applet.*;
import java.awt.*;
        public class bdemo  extends Applet
{
        Button btn [ ];
        String str [ ] = { "1", "2", "3", "4", "5", "6", "7", "8", "9'};
        public void init ( )
{
        setBackground (Color.Red);
        setLayout(new GridLayout (3, 3));
        btn = new Button[str.length] ;
        for(int I = 0; I < str.length; i++)
{
        btn [I] = new Button (str[I]);
        add (btn [1]);
}
}}
```

CardLayout Manager

The 'CardLayout' places components(usually panels) on top of each other in a stack like a deck of cards. We see only one card at a time, and we can flip through the panels by the using another control to select which panel comes to the top.

The CardLayout is a good layout to use when we need to have different panels containing different components but in one single Frame.

Frames: -

A frame is independent window. Frames are mostly used with applications which require graphical user interface (GUI). When you construct a frame. It does not have a size of the frame and the frame can be displayed by envoking the method setVisible( ).

Example of frame.

```
Import java.awt.*;
Class mfdemo extends Frame
{
Button b1, b2, n3;
Public mfdemo (String str)
{
super (str);
```

```
setSize (300, 300);setVisible(true);
setLayout (new FlowLayout ( ));
b1 = new Button ("Java");
b2 = new Button ("C + +");
b3 = new Button ("C");
add (b1);
add (b2);
add (b3);}}
Class mdemo
{
Public static void main (string args [ ] )
{
Mfdemo m = new mfdemo ("my frame");
}}
```

My Frame

Menus: -

Java program has rich collection of classes for creation of menus. There are two types of menus. Pull down and popup menu. Menu is provided to easy the use of application developed only one menu bar is a horizontal bar placed at the top and list different options for selections which is known as menu. Individual menu can contain sub options which are known as menu items.

Example of menu is.

```
Import java.awt.*;
Class mfdemo extends Frame
{
MenuItem m;
Public mfdemo ( )
{
subtitle ("Menu Example");
setSize (300, 300);
MenuBar mb = new MenuBar ( );
Set MenuBar (mb);
Menu f = new Menu ("File");
Mb. Add (f);
menuItem n = new MenuItem ("New");
f.Add (N);
MenuItem op = new MenuItem (Open");
f.add (Op);
F.addSeparator ( );
MenuItem ex = new MenuItem ("Exit");
f.Add (Ex);}}
```

program of menu.

```
Import java.awt.*;
Class mframe extends Frame
{
PopupMenu op;
Public mframe ( )
{
Op = new PopupMenu ("Options");
f.add (op);
MenuItem r = new MenuItem ("Read");
Op.add( r);
Op.addSeparator ( );
Menu fm = new Menu ("Format");
Op.add (fm);
This.ad (op);
CheckboxMenuItem cb = new
CheckboxMenuItem ("Insert",true);
Fm.add (cb);
}
```

Dialog: -

Dialog are popup window that are displayed when the programmer wishes to pass a message to the user of the program.

Example of dialog.

```
Import java.awt.*;
Import java.awt.event.*;
Public class dia extends frame implements
ActionListener
{
Button dis, ok, exit;
Dialgo di;
Public dia ( ){
setSize (300, 300)(
```

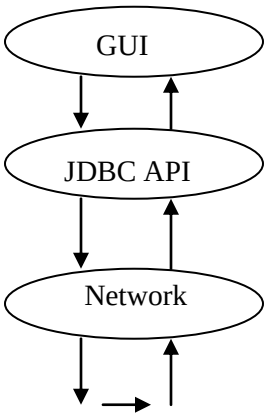
```
setLayout (new FlowLayout ( ));
dis  = new Button ("Display dialog");
dis.add ActionListener (this);
add .(dis);
exit = new Button ("Exit");
exit.add ActionListener (this);
add (exit);
ok = new Button ("ok");
add (ok);
setVisible (true);}
Public void actionPerformed (ACctionEvent e)
{
If (e.getSource ( ) = = dis){
Di = new Dialog (this, "Alert", true);
di.setLayout (new FlowLayout( ));
di.setSize (200, 100);
Label L = new Label ("click on");
di.add (L)};
di.setVisible (true);}
If (e.getSource = = exit)
{
System.exit (0);}}
Class mdemo
{
Public static void main (String args [ ] )
{
Dia m = new dia ("My dialog");
m. show( );}}
```

JDBC: -

JDBC was outlined (developed) by java soft. The applet and application written in java can access remote database with the help of JDBC. Information can be retrieving and updated if required.

JDBC API: -

Java database connectivity application programming interface is a set of specification that defines how a java program can communicate with the database. To be more specific, it defines how an application opens a connection, communicates with the database, executes SQL statements and retrieve the results. The following figure chows about the JDBC API.



JDBC API was developed by java soft. Many of the JDBC API concepts are taken from other sources, Microsoft OBC (open database connectivity). Both ODBC and JDBC are based on call, level interface. But ODBC is well known and is one of the widely used database interface.

It is generally thought that JDBC is a part of ODBC to java programming language. But the major difference between them is ODBC is written in C language where as pointer but java does not have pointers. More than this is the design aspect of ODBC and JDBC. JDBC was designed to be compact and focused only on the execution

of SQL statements and thereby retrieving the results, but ODBC has multiple mechanism for performing a single task and has additional data handling capabilities.

JDBC API is much smaller and easier to implement as compared to the ODBC. JDBC API incorporates only those tasks that are considered essential.

Java SQL package: -

This package has collection of classes and interfaces which used for communicating with the database. This package consists of the following interfaces and classes.

Interfaces: -

1. Callable statement
2. Connection
3. Driver
4. Result set
5. Statement

Class Name: -

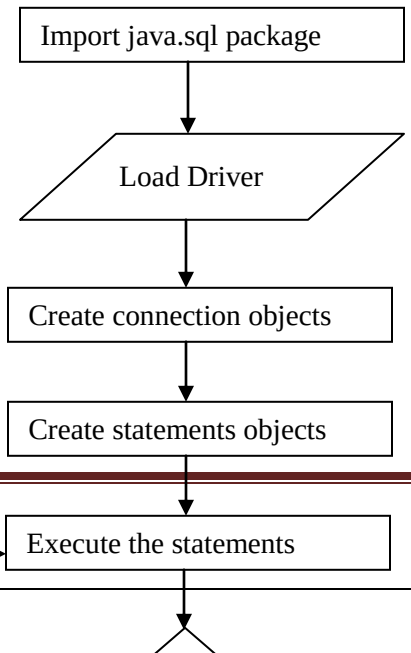
1. Data
2. Driver Manager
3. Driver Property Info
4. Time
5. Type

Basic steps of JDBC: -

There are about seven steps using JDBC to access a database. Each step is important and is required to complete a database transaction. The seven steps are;

1. Import java.sql package
2. Load and register the driver
3. Establishing a connection to the database server
4. Creating a statement
5. Execution to statements
6. Retrieving the result
7. Closing the statements and connection

flowchart of basic step of JDBC.



Foundation class: -

A no. of software platform today provides us with a new facility called foundation classes. The foundation classes simplify the designing process and reduce the time taken to code. With the help of foundation classes, the design process gets simplified and the final product has a better look and fills.

Microsoft foundation classes (MFC) and java foundation classes (JFC) has two popular used foundation classes.

Q. Describe about JFC.

The java foundation classes extend the original awt by adding s set of GUI class libraries. Java foundation class provides us with additional visual component classes and a unique way of designing the screen.

Q. what is swing (advanced form of awt)?

Swing is a set of classes under the JFC that provide light weight visual components and enable creation of an attractive GUI, swing not only contains replacement component for awt visual component but also complex thathaven't awt provide.

In swing, the main window also called top level container which contains all of the swing components that appear inside the window. Every top level container has an intermediate container called content – pane. All swing component names starts with J. the swing components are in the Javax.swing package.

Example of swing.

```
Import java.awt/*;
Import javax.swing.*;
Public class sdemo extends JApplet
{
Public void init ( )
{
getContentPane ( ).setLayout (newFlowLayout ( ));
ImageIcon ic = new ImageIcon ("a.gif");
JLabel L1 = new JLabel ("tis Java", ic, SwingConstants.LEFT);
getContentPane( ).add (L1);
}}
```

Input / output stream: -

The meaning of stream is flow. A stream is a path by which the data travels in a program. Streams are pipelines for sending and receiving information in java programs when a stream of data is being sent or received, we referred to it as writing and reading a stream respectively. While reading or writing a stream

if an errors occurs, the IO Execution is thrown. Hence, the stream statement most consists of try and cache block.

The class "java.lang.system." defines the standard input /output stream. The stream I / O stream consist of;

1. ***System.out class:*** **-**The standard output stream is used to typically display the output on the screen.
2. ***System.in class:*** **-** The standard input stream comes from keyboard and id used for reading character of data.
3. ***System.err class:*** **-** This is the standard error stream.

The java IO package: -The java IO package contains the following classes.

1. **The InputStream class: -**

The class InputStream is an abstract class. It defines how data is received. The class InputStream provides a no. of methods for reading and taking streams of data as input. These method help in creating, reading and processing Input Stream.

a. Read ( ) b. Available ( ) c. Close ( )
d. Mark ( ) e. Reset ( ) f. Skip ( )

2. **The Output Stream class: -**

The class Output Stream is also abstract. It defines the way in which outputs are written to stream. The class provides a set of methods that help in creating writing and processing output streams.

a. Write ( ) b. Flash ( ) c. Close ( )

Program of input / output stream.

```
Import java.io.*;
Class ca
{
Public static void main (String args [ ] ) throws Exception
{
Char c = 'z';
Long l = '0';
FileInputStream is = new FileInputStrem ("data.txt");
FileOutputStream os = new FileOutputStream (ndata.txt");
Byte b [ ] = new byte [is. Available ( )];
L = is. Available ( );
Is. Read (b);
System.out.println (l);
For (int = = 0; I < l; I + +)
{
c = (char) b [i];
If (c = = 'a')
{
System.out.println ("A is found");
os.write (c byte ) + + c;
}
else
{
os.Write ((byte) c) ;}}}}
```

Graphic class: -

The awt package provided by java allows drawing graphics. The class Graphics provided in awt package has a rich collection of methods. These methods are used to draw any graphical figure like lines circle, square, rectangle.

To draw any figure, a graphical background is required. A graphical background can be obtained by getGraphics ( ) or when any of the following three methods are called.

1. Paint (Graphics g)
2. Repaint ( )
3. Update (Graphics g)

Example is.

```
Import java.awt.*;
Public class ds extends frame
{
Public dis( )
{
Super ("Dra String");
setSize (300, 300);
setVisible (true);
}
Public void paint (Graphics g)
{
g.drawStrig ("Hello java", 50, 50);
}
Public static void main (String args [ ] )
{
New ds ( );}}
```

Drawing lines and ovals: -

Syntax: -

1. drawLine(int x1, int y1, int x2, int y2)
2. drawOval (int xcor, int ycor, int width, int height)
3. drawRect (int xcor, int ycor, int width, int height)
4. drawFoundRect (int xcor, int ycor, int width, int height, int arc width, int arc height)

Color control: -

Color control in java is achieved through the three primary color, red, green and blue. Java uses RGB color mode. An object of the class color contains three integer or float values for these three parameters Red, Green and Blue.

Element	Range
1. Red 2. Green 3. Blue	1. 0 – 255 2. 0 – 255 3. 0 - 255

Example, write under paint command.

Color c = new color (255, 0, 0)

g. Set color ( )

Font class: -

Fonts are used to display the txt in different forms. The class Font provided in the java.awt package allows the user of various fonts to display text. The class toolkit is provided by the language, which gives platform dependent information such as fonts available and screen size.

Example of font class.

```
Import java.awt.*;
Public class Ds extends frame
{
Public Ds ( )
{
Super (“Draw String”);
setSize (300, 300);
setVisible (true);
}
Public void paint (Graphics g)
{
Toolkit t = Toolkit.getDefaultToolkit ( );
Font f = new font (“Times New Roman”, font .PLAIN, 22);
Font Metics fm = t.get font metrics (f);
String na = fm. getFontName ( );
g.drawstrin (“Details of font” + na, 30, 50);
}
Public static void main (String args [ ])
{
New Ds ( );
}
}
```

Stream tokenizer: -

Stream tokenizer breaks up the input stream into token that are delimited by set of characters. It has used the following constructors. Stream tokenizer (Reader in stream). Stream tokenizer defines general method.

1. To reset the default of set delimiters we will used reset syntax ( ).
2. Eollos Significant ( ) to insure that new line character will be delivered as tokens, so we can count the number of lines as well as words.
3. The word chars ( ) is used to specify the ramge of character that be used in word, void word chars (int start, int end)
4. The white space characters are specified using white space chars ( ) void white space chars (int start, int end)

5. The next token is obtained from the input stream by calling next token ( ). It returns the type of token.

Vector class: -

One of the problems with an array is that we must know how big it must be when we create it. It is not always possible to determine the size of the array before creating it. The java vector class solves this problem by providing a form of resizable array that can grow as more elements are added to it.

A vector stores item of type object so that it can be used to store instants of any java class. A single vector may store different elements that are instants of different class. The following constructor used in vector class;

Constructor: -

1. Vector (int)
2. Vector (int, int)
3. Vector ( )

Methods: -

Add Element (object)
Capacity ( )
Index of (object)
Size ( )

Example of vector class is.

```
Import java.util.*;
Public class vdemo
{
Public static void main (String args [ ])
{
Vector  v = new Vector ( );
v.addElement (“One”);
v.addElement (“Two”);
v.addElement (“Three”);
v.addElement (“Four”);
System.out.println (“size” + v.size ( ));
v.remove Element (“Two”);
System.out.println (“size is” + v.size ( ));
{
System.out.println (v.ElementAt (i) + “ “);}}}
```

File dialog: -

Java provided a built in dialog box that the user specify a file. To create a file dialog box, initiate an object of type file dialog.

Example is.

```
Import java.awt.*;
Class aa extends frame
{
Public aa ( )
{
Super (tittle);
File Dialog fd =  new File Dialog (“File Dialog”);
Fd.setVisible (true);
```

…….. Do your self

Scroll bar: -

```
Import java.awt.*;
```

```
Import java.awt.event.*;
importjava.applet.*;
public class Sc extends Applet
{
Ms s;
Public void init ( )
{
S = new ms (scroll bar.
HORIZONTAL,0,1,0,100);
Add (s);}
Class Ms extends Sc
{
Public Ms (int style, int initial, int thumb, int
min, int max);{
Super (style, initial, thumb, min, max);
enableEvent (AWTEVENT.ADJUSTMENT
EVENT MASK);}
Protected void process Adjustment Event
(Adjustment Event e){
showStatus ("Adjustment" + e.getValue ( ));
setValue (getValue ( ));
super.process Adjustment Event (e);}}}
```

Canvas: -

Canvas is a drawing area which can also receiver mouse event by default canvas is not capable of drawing as the paint method of the canvas class, sets only the background color.

Example is.

```
        Import java.awt.*;
        Import java.applet.*;
        Class me extends canvas
{
        Public final static int rect = 1, oval = 2, sq = 3, line = 4;
        int fig;
        Public void paint (Graphics g)
{
        If (fig = = rect)
        g.fillRect (30, 40, 90, 100);
        else if (fig = = oval)
        g.fillOval (25, 25, 600, 160)
        else if (fig = = line)
        g.drawLine (10, 10, 60, 60);
} }
```

Destructor: -

Destructor is use to remove unusable objects from memory. Java handles destructor method automatically where as in C It is done by manually by using delete operator. The technique to clear the memory is also known as garbage collection. It works like; when no references to an object exit that object is assumed to be no longer needed and the memory occupied by the object can be reclaimed garbage collection only occurs during the execution of programs. To handle destructor use the following method.

```
Protected void finalize ( )
{
}
```

By using finalization, we can define specific actions that will occur when an object is just about to be reclaimed by the garbage collector. To add a finalize to a class, we simply the define the finalize ( ), inside the finalize method we will specify those actions that must be performed before an object is destroyed.

The garbage collector runs periodically checking for object that are no longer referenced by any running state or indirectly through other referenced object.

Example of destructor is.

```
Class student
{
Student ( )
{
……————
……————...
}
protected void finalize ( )
{
System.out.println ("Destructor is running");
}
}
Class main
{
……————..
……————..
}
}
```

JAR File: -

The Java.util.package provides the avability to read and write java archive (JAR) files. A JAR file allows you to efficiently deploy the set of classes and their associated resources.

JAR technology makes much easier to deliver and install software. The elements in a JAR file are compressed which makes downloading JAR file much faster than separately downloading several uncompressed file. Digital signature may also be associated with the individual elements in a JAR files. For example, a developer may build a multimedia application that uses various sound and image files. A set of beans can control how and when this information is presented. All of these pieces can be placed into one JAR file. The JAR keyword help to create JAR file from class files.

Program of database connectivity.

```
import java.sql.*;
class dba
{
	public static void main(String args[])
	{

		ResultSet rs;

		try
		{
			Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
			Connection con;
			Statement sa;
			con=DriverManager.getConnection("jdbc:odbc:student");
			sa=con.createStatement();

			rs=sa.executeQuery("select roll,name from student");
			while(rs.next())
								{

				System.out.print(rs.getString(1)+"   ");
				System.out.println(rs.getString(2));
				System.out.println();
			}
		}
		catch(Exception e){
			System.out.println("hello and how");
		}
	}

}
```

```
import java.awt.*;
import java.awt.event.*;
import java.sql.*;

public class da2 extends Frame implements ActionListener
{
	static Connection con;
	static Statement sa;
	static ResultSet rs;
	Label l1,l2,l3;
```

```
TextField t1,t2,t3;
Button b1,b2,b3,b4;
Panel p;
public da2()
{
        super("using database ");
        setBackground(Color.blue);

        l1=new Label("id");
        l2=new Label("name");
        l3=new Label("salary");
        t1=new TextField(10);
        t2=new TextField(10);
        t3=new TextField(10);
        b1=new Button("Exit");
        b2=new Button("select");
        b3=new Button("Insert");

        p=new Panel();
        p.setLayout(new GridLayout(6,2));
        b1.addActionListener(this);
        b2.addActionListener(this);
        b3.addActionListener(this);

        p.add(l1);
        p.add(t1);
        p.add(l2);
        p.add(t2);
        p.add(l3);
        p.add(t3);
        p.add(b1);
        p.add(b2);
        p.add(b3);

        add(p);
        pack();
        setVisible(true);
                }

public static void main(String args[])
{
        da2 d=new da2();
        try
```

```
{
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        con=DriverManager.getConnection("jdbc:odbc:mm");
        sa=con.createStatement();
        rs=sa.executeQuery("select id,name,salary from employee");
        rs.next();
        }
        catch(Exception e)
        {
        System.out.println("error"+e);

        }
        d.show(rs);
        }
        public void actionPerformed(ActionEvent e2)
        {
                if(e2.getSource()==b1)
        {
        try
        {
        //      a a1=new a();
        //      a1.setVisible(true);
                System.exit(0);
        }
        catch(Exception e)
        {
        }
                }
                else if(e2.getSource()==b2)
                {
                        try
                        {
                        //      rs.next();

                        select(rs);

                        }
                        catch(Exception e)
                        {
                        }
                        show(rs);
                }
```

```
                    else if(e2.getSource()==b3)
                    {
                        try
                        {
                        insert();
                            }
                            catch(Exception e)
                    {
                        System.out.println("error");
                    }
                    show(rs);
                }

                }
```

```
    public void insert()
    {
        try
        {

                sa.executeUpdate("insert into employee
values("+t1.getText()+","'+t2.getText()+'","+t3.getText()+")");
                        System.out.println("one record saved");
                        }
                        catch(Exception e)
                        {
                            System.out.println(e);
                        }

    }
        public void select(ResultSet rs)
    {
        try
        {
                while(rs.next())
                {
                    System.out.println(rs.getString(1)+" ");
                    System.out.println(rs.getString(2)+" ");
                    System.out.println(rs.getString(3));

                }
```

```
                                        }
                                        catch(Exception e)
                                        {
                                                System.out.println(e);
                                        }

        }
        public void show(ResultSet rs)
        {
                try
                {
                        t1.setText(rs.getString(1));
                        t2.setText(rs.getString(2));
                        t3.setText(rs.getString(3));
                }
                catch(Exception e)
                {
                //      System.out.println("error"+e);
                }
        }

}
```