

Disadvantage of C language

- I. As C as the support for global Variables, hence C is not a very secure programming language because data stored in global variables can be easily modified by any function.
- II. C is a platform dependent language.

Idea Of OOPS

- A programming language which deals with concept of object is called an Object oriented programming language.

What do you mean by Object?

An object is a representation in real world entity, with unique identity, embedded property and the ability to interact with other objects.

What are the features of object oriented Programming?

- ❖ Encapsulation
- ❖ Data Abstraction
- ❖ Polymorphism

❖ Inheritance

Encapsulation

- Encapsulation is the mechanism by which the variables and the function which work with those variables are group together in a single unit. In an object oriented programming Encapsulation is done using Class.

Data Abstraction

- Data Abstraction means hiding the working complexity of Data.
Encapsulation is the Mechanism and the Outcome is Data Abstraction.

Polymorphism

- The Term Poly means many the morph means form. It is mechanism where the same interface is used in order to achieve multiple task.

- In an oops, polymorphism is used in function like:-function Overloading ,Function Overriding, Constructor Overloading.

Inheritance

Inheritance is the Mechanism by which one class acquires the features of another class. Through Inheritance OOPS supports the concept of code reusability.

Types of OOPS

- ❖ Partial OOPS
- ❖ Fully OOPS
- ❖ Complete OOPS

Partial OOPS

- A partial OOPS is that Programming is that Programming where all the four features are supported but any one or two features are partially supported.
e.g. C++

Fully Object Oriented

- In a Fully OOPS all the four features of OOPS supported FULLY.
e.g. JAVA

Complete Object Oriented

For a language to be complete Object Oriented:-

- First it has to be fully object oriented.
- It should not have support for primitive data type.
e.g. Small talk.

Software:- It is basically a collection of **interrelated** programs.

Program:- It is a collection Of Instructions.

Instruction:- It is basically a command which we give to our computer in order to

achieve a particular task using a specific language.

Introduction to JAVA:-

- Java language was developed by James Gauslin and Patrick Naughton at Sun Microsystem.
- Java was Initially called as Oak Language.
- Java was basically an object oriented programming language in early 90's.
- The language itself takes much of its syntax from C and C++ but has a simpler object model and eliminates low-level tools.
- It is platform independent. Early implementations of Java had the slogan "Write once, run anywhere".

Features Of JAVA

- ❖ Platform Independent
- ❖ Fully Object Oriented

- ❖ Case Sensitive
- ❖ Compiled and interpreted
- ❖ Strongly Typed
- ❖ Very much Secure
- ❖ Support Multitasking
- ❖ Do not support Pointers and Multiple Inheritance.

Why Java is called a platform Independent Programming Language?

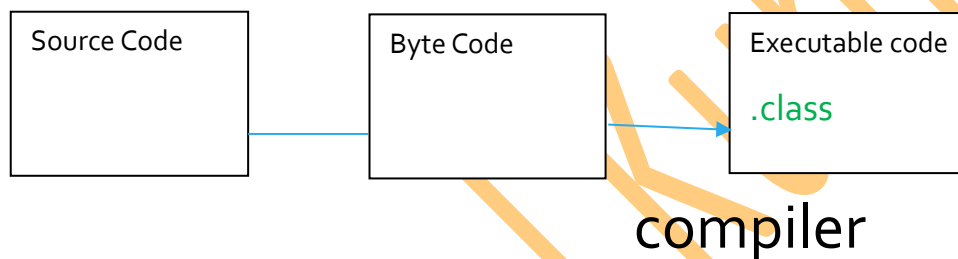
When we compiled a java program it creates a highly optimizable code which is called a byte code. This Byte code can run on any computer with different hardware and software architecture. That's why JAVA is platform independent language.

Why Java is called a compiled as well as interpreted programming Language?

- As both compiler and interpreter are required for compiling and running a java

program hence java is called compiled and interpreted programming language.

- The java compiler translate the java source code to the corresponding byte and the java interpreter correspond into byte code.



Why JAVA language is strongly Typed?

For a Language to be strongly type it must follow two rules:-

- Every variable should be declared before they can be used in the program.
- We can not assign mismatch data type to one another.

Hence, JAVA follow two rules strictly that's why JAVA is strongly type.

Why JAVA program are so much secure?

- All JAVA programs runs in a close environment which is called JAVA runtime Environment(JRE).
- No other programs outside the environment will be able to access the program inside.

Multitasking

- JAVA is The programming to support the concept of multitasking using the concept called thread.

NO Pointer

JVM

- JVM stands for JAVA VIRTUAL MACHINE. JVM is interpreter for byte code. It accepts byte code as input Executable code and runs it simultaneously.
- JVM is platform dependent in nature.

JDK

- JDK stands for Java Development Kit.
- JDK is a software which contains various application files which are required to compile, debug and run a java program.

Tokens in Java

Token is the smallest individual unit that makes up a programming language.

- **Identifiers:-** It is the names given to various programming elements.
e.g. Class name, variable name.
- **Variables**
- **Constants:-** It can be classified into two types:-

- ✓ Numeric constant
- ✓ Non-Numeric constant

Numeric constant can be classified into two types:-

- Integer
- Float

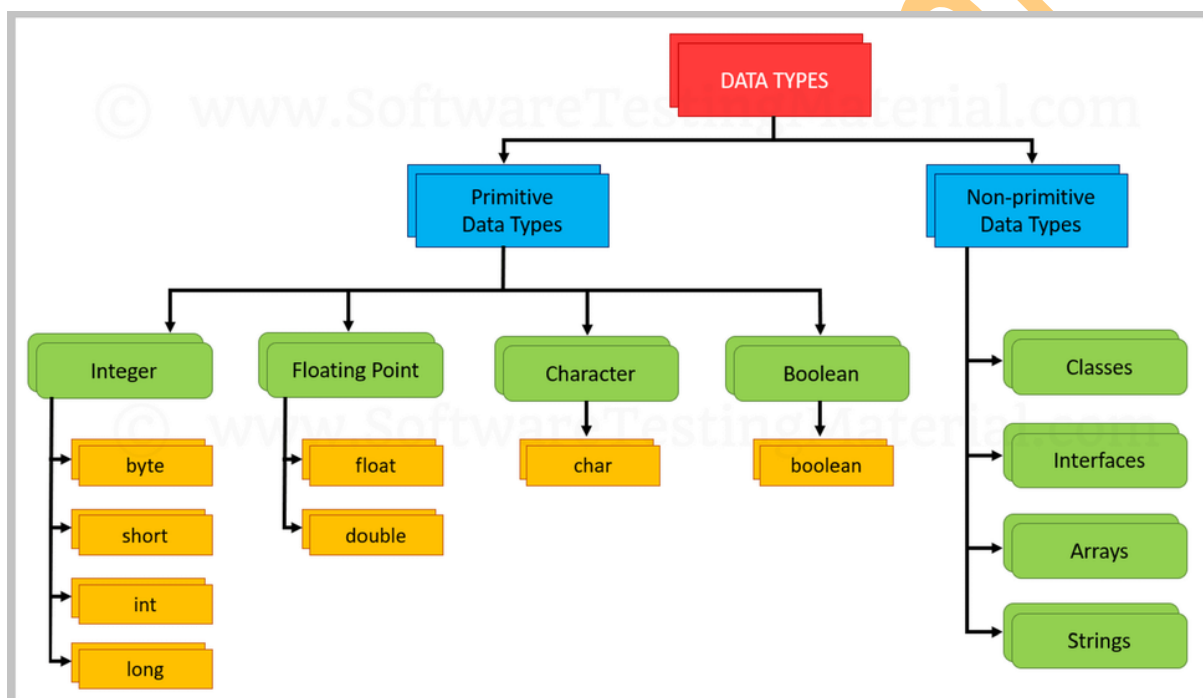
Integer can be classified into three types:-

- Hexa-Decimal
- Octal
- Decimal

Non-Numeric constant can be classified into two types:-

- **Character constant:-** A character constant represent a single character enclosed with single quotes.
- **String Constant:-** A string constant is a sequence of character enclosed with double quotation.

The Data type defines the type of data which can be stored in the computer's memory.



- Boolean takes one bytes of memory
- Char takes two bytes of memory.
- Int four bytes.
- Float 4 bytes.

Why in Java a Character Occupies 2 bytes of memory?

- Java supports Unicode character set which is a collection of all human known languages characters all over the globe. In order to cooperate char data types in Java takes Two bytes of memory.

Steps for setting paths in JAVA:-

- Step 1: copy the entire path upto bin which contains the java compiler or interpreter.
- (C:\Program Files\Java\jdk-1.8\bin)
- Steps 2: Right click on this PC option.
- Steps 3: click on the properties option.
- Steps 4: click on the advanced system settings options.
- Steps 5: click on the environment variables options.

- Steps 6: In the system variable option click on the path option.
- Step 7: click on the Edit Option.
- Step 8: click on the new option.
- Step 9: paste the copy path over there.
- Step 10: OK..Ok..Ok..

Java program to display your name?

```
public class test {  
    public static void main(String  
x[]) {  
        System.out.println("Welcome  
Ujjawal");  
    }  
}
```

Commands for command prompt:-

- For change drive c to d we use "d:"
- To access folder "cd folder name"
- To compile file "javac filename.java"
- To run file "java file name"

Use of Class and Object:-

Class

- A class is used to group variables and methods which access those variables as a single unit.

Syntax:-

```
class classname{  
    Instance variable declaration  
    Method Definition  
}
```

Here “classname” is a valid identifier.

Object

- An Object is defined as an instance of a class.
- Steps for creating an object of a class:-
 - Step 1: Create a reference variable of a class.
 - Step 2: Create an object of a class and assign to the reference variable.

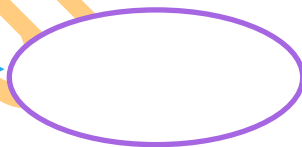
```
class T{  
Class
```

```
.....
```

```
.....
```

```
}
```

ob



1. T ob;-Creating a reference variable of the class.
2. Ob=new T();
3. T ob=new T();

Write a program to implement classes and object?

```
class Test1{
    int a,b;//instance variable
    void input(){
        a=10;
        b=20;
    }
    void disp(){
        System.out.println("The values
are "+a +" "+b);
    }
}

public class Test2 {
    public static void main(String
x[]){
        Test1 ob=new Test1();
        ob.input();
        ob.disp();
    }
}
```


Write a program in java to accept two values in two instance variables. Find their sum and display the result.

```
class test1{
    int a,b,c;
    void input(){
        a=5;
        b=10;
        c=0;
    }
    void calculate(){
        c=a+b;
    }
    void display(){
        System.out.println("Sum is:-" +
c);
    }
}

public class test3{
    public static void main(String
x[]){
        test1 ob=new test1();
        ob.input();
        ob.calculate();
        ob.display();
    }
}
```

```
}  
}
```

Write a program to display passing the value b/w the two class?

```
class test1{  
    int a,b,c;  
    void input(int x,int y){  
        a=x;  
        b=y;  
        c=0;  
    }  
    void calculate(){  
        c=a+b;  
    }  
    void display(){  
        System.out.println("Sum is:-" +  
c);  
    }  
}  
class test5{
```

```
public static void main(String
x[]){
    test1 ob=new test1();
    test1 ob1=new test1();
    ob.input(5,10);
    ob.calculate();
    ob.display();
    ob1.input(51,10);
    ob1.calculate();
    ob1.display();
}
```

Constructor

- A constructor is a special method which has the same name as that of the class within which it resides.
- A constructor does not have any return type not even void.
- A constructor is called automatically at the time of creation of object.

- A constructor is used in order to initialize an object.

Write a program to display use of Constructor?

```
class Constructor1{
    int a,b;
    Constructor1(){//constructor
        a=10;
        b=20;
    }
    void display(){
        System.out.println(a+" "+b);
    }
}
class test6{
    public static void main(String
args[]){
        Constructor1 ob=new
Constructor1();//here constructor
called automatically
        ob.display();
    }
}
```

Types of Constructor

A constructor can be classified into mainly two types:-

- Default Constructor
- Parametrized Constructor

Default Constructor

- A constructor which can not take any argument is called Default Constructor.

e.g. Constructor1(){//constructor

```
a=10;
```

```
b=20;
```

```
}
```

Parametrized Constructor

- A constructor which takes argument is called Parametrized Constructor.

Write a program in java to display use of parametrized constructor.

```
class Test1{
    int a,b;
    Test1(int x,int y){//Parametrized
constructor
    a=x;
    b=y;
}
    void display(){
        System.out.println(a+" "+b);
    }
}
    class seventh{
        public static void main(String
args[]){
            int x=16,y=32;
            Test1 ob=new Test1(x,y);//here
constructor called autmatically
            ob.display();
        }
    }
```

Why main function is public static void?

- The main method is declared as public so it is accessible to any outside code. Here outside code is JVM.
- Public is not important for compiling, But it is important for running.
- The main method is declared as static so that JVM does not have to create an object of the class, In order to call main method which is present inside the class.
- The main method is declared as void because it does not return any value to JVM.

Why a non-parametrized constructor is called default constructor?

- If we as a coder does not provide a constructor to a program the java compiler will automatically provide a non-parameterized constructor by default at the time of compilation. That's why a non-parameterized constructor is called default constructor.
- The purpose of default constructor is to provide default value to the instance variable depends on the data type.

If data type is int then it give "0", float->"0.0", char->"null", Boolean->"false".

Method Overloading/Function Overloading

- In Java, It is possible to have more than one method with the **same name** within the **same class**. This mechanism is called as method overloading. The methods should differ in their numbers or types of arguments.

- The methods are called as overloaded methods.

Write a program to display method overloading in java?

```
class fun_overload{
    void disp(){
        System.out.println("No
arguments");
    }
    void disp(int x){
        System.out.println(x);
    }
    void disp(int x,int y){
        System.out.println(x+" "+y);
    }
}
class m_overload{
    public static void main(String
args[]){
        fun_overload ob=new
fun_overload();
        ob.disp(50,60);
        ob.disp();
        ob.disp(67);
    }
}
```

```
}  
}
```

- Using method of overloading Java supports the concept of polymorphism.

Constructor Overloading

- In Java it is possible to have multiple constructors in a class.
- This mechanism is called constructor overloading and the constructors are called as overloaded constructors.

**Write a program to display
Constructor overloading in
Java?**

```
class CO{  
    int a,b;  
    CO(){  
        a=20;
```

```
        b=40;
    }
    C0(int x,int y){
        a=x;
        b=y;
    }
    C0(int x){
        a=x;
        b=180;
    }
    void disp(){
        System.out.println("The values of
a & b are:-"+a+" "+b);
    }
}

class const_over{
    public static void main(String
args[]){
        C0 ob1 =new C0(50);
        C0 ob2=new C0(50,80);
        C0 ob3=new C0();
        ob1.disp();
        ob2.disp();
        ob3.disp();
    }
}
```

- Through Constructor Overloading Java supports the concept of Polymorphism.

Command Line Argument

- Some times in Java we need to pass values through command lines. This can be achieved using concept of command line argument. When values are passed using command line there passed in the form of array or string.

Program to display using
command line program?

```
class Command_L_A{  
public static void main(String a[]){
```

```
for(int i=0;i<a.length;i++){  
    System.out.println(a[i]);  
}  
}  
}
```

Java Command_L_A Ujjawal Kumar

Output:-

Ujjawal

Kumar

In Java there is a wrapper class called integer that wrapper class ends with parseInt which is used to convert string value to integer value.

Syntax:-

```
int x=Integer.parseInt(a[0]);
```

```
int y=Integer.parseInt(a[1]);
```

```
int z=x+y;
```

code Example:-

```
class sum{
    public static void main(String
a[]){
    for(int i=0;i<a.length;i++){
    System.out.println(a[i]);
    }
    int x=Integer.parseInt(a[0]);
    int y=Integer.parseInt(a[1]);
    int z=x+y;
    System.out.println(z);
    }
}
```

Use of "this" key word in Java

- In a Java Program if an Instance variable has the same name as that of local variable then there will be a naming collision b/w local and instance variable.

- To resolve this name space collision we have to use this keyword.
- The “this” keyword always referred an instance of class.

Write a program to use “this” keywords?

```
class this_eg{
    int a,b;//intance variable
    this_eg(int a,int b){
        //local variable
        this.a=a;
        this.b=b;
    }
    void disp(){
        System.out.println(a+" "+b);
    }
}
class this_main{
    public static void main(String
args[]){
        this_eg ob=new this_eg(20,40);
        ob.disp();
    }
}
```

```
}
```

Inheritance

- Inheritance is the mechanism by which one class inherits the features of another class.
- Through Inheritance Java supports the concept of code reusability.

Types of Inheritance

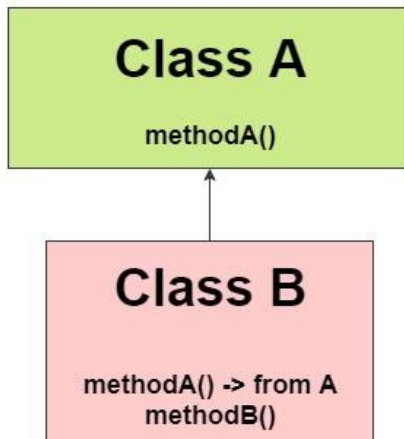
Inheritance can be classified into following types:-

- Singlelevel Inheritance
- Multilevel inheritance
- Multiple inheritance
- Hierarchical Inheritance
- Hybrid Inheritance

Single Level inheritance

- In a single level Inheritance one class is derived from another class.

Single Inheritance



to implement single level inheritance?

to implement single

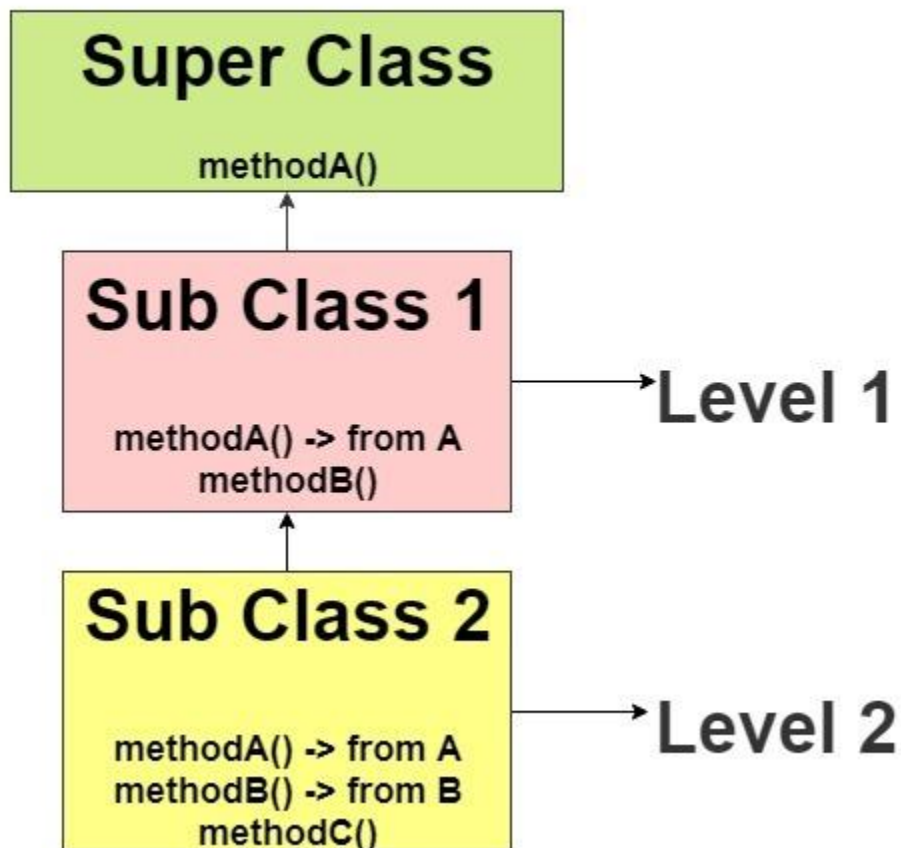
```
class A{
    int i=10;
}
class B extends A{//inheriting
parent class
{
    void disp(){
        System.out.println(i);
    }
}
class si{
    public static void main(String
a[]){
        B ob=new B();
```

```
ob.disp();  
}  
}
```

Multilevel inheritance

- In Multilevel inheritance one class is derived from another class.

Multi-Level Inheritance



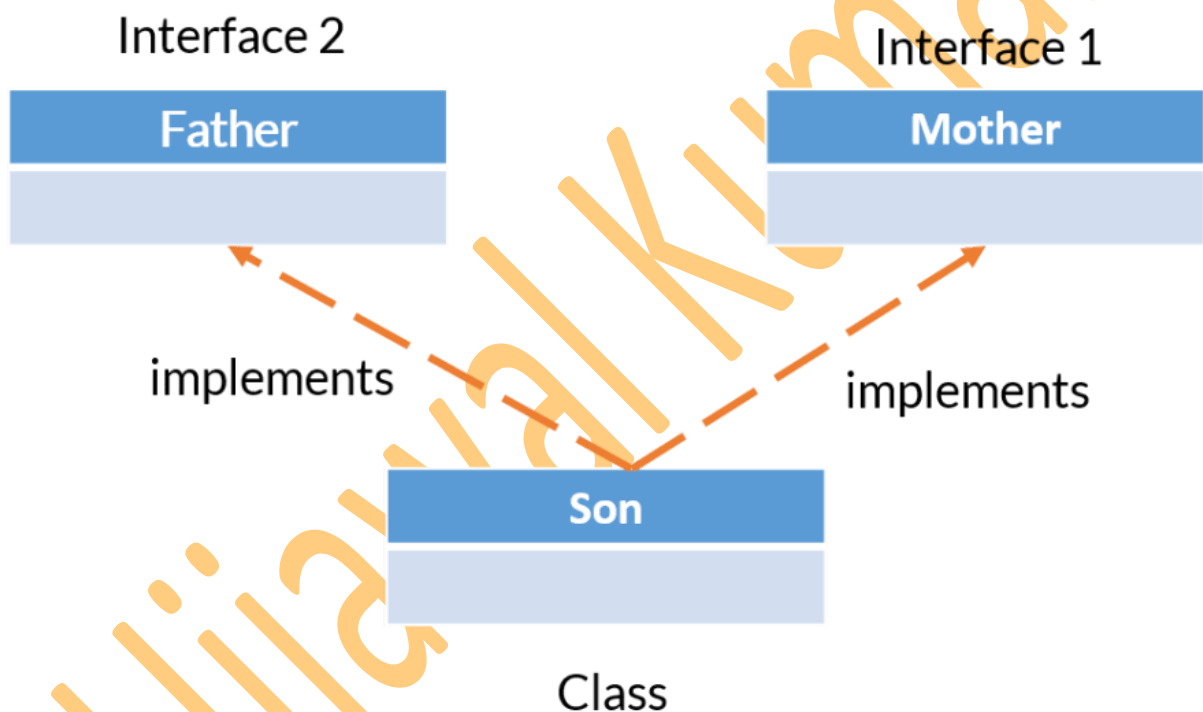
Write a program to display multilevel Inheritance?

```
class A{  
    int a=50;
```

```
    }  
    class B extends A//B is the child  
class of A  
    {  
        int b=a*a;  
    }  
    class C extends B //C is the  
child of B & grandchild of A  
    {  
        void display(){  
            System.out.println("The value of  
a="+a);  
            System.out.println("The value of  
b="+b);  
        }  
    }  
  
    class multi_inherit{  
        public static void main(String  
args[]){  
            C ob =new C();//creating object  
of class C to //call the values of  
functions and variables from all  
parent class A&B  
            ob.display();  
        }  
    }
```

Multiple Inheritance

- In multiple Inheritance one class is derived from



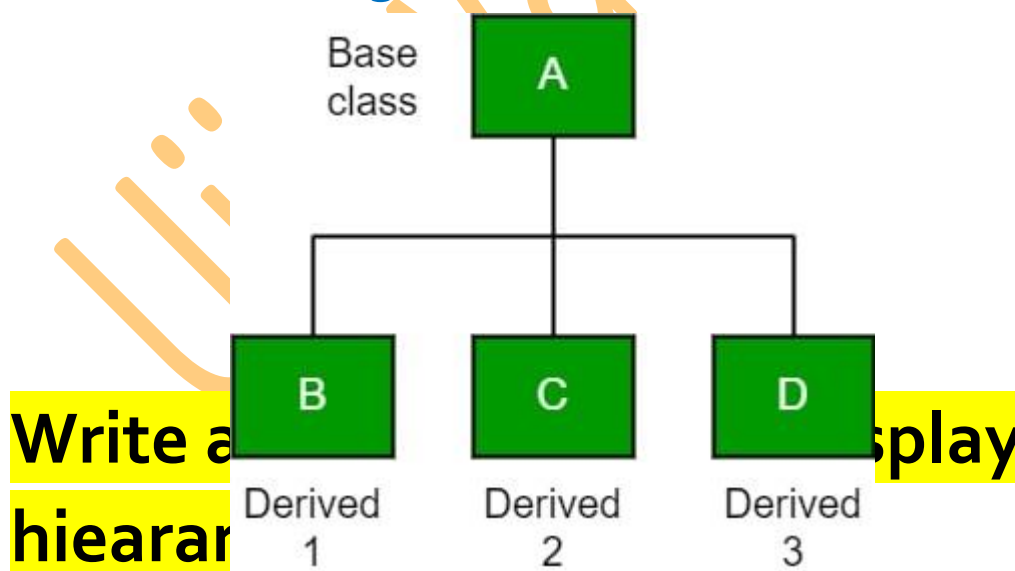
- Java does not support the concept of multiple inheritance.
- If multiple path exist b/w grandparent and grand child the grandparent variable

unnecessary replicated in the grand child class.

- In order to prevent this Java does not support the concept of multiple Inheritance.
- In Java Multiple inheritance concept is supported indirectly by using concept interface.

Hierarchial Inheritance

- In this inheritance one class is derived from a single base class.



```
class A{  
    int a=50;  
    int b=100;
```

```

    }
    class B extends A//B is the child
class of A
    {
    int c=a*b;
    void display1(){
    System.out.println("The value of
c="+c);
    }
    }
    class C extends A //C is the
child of A
    {
    int d=b/a;
    void display2(){
    System.out.println("The value of
d="+d);
    }
    }

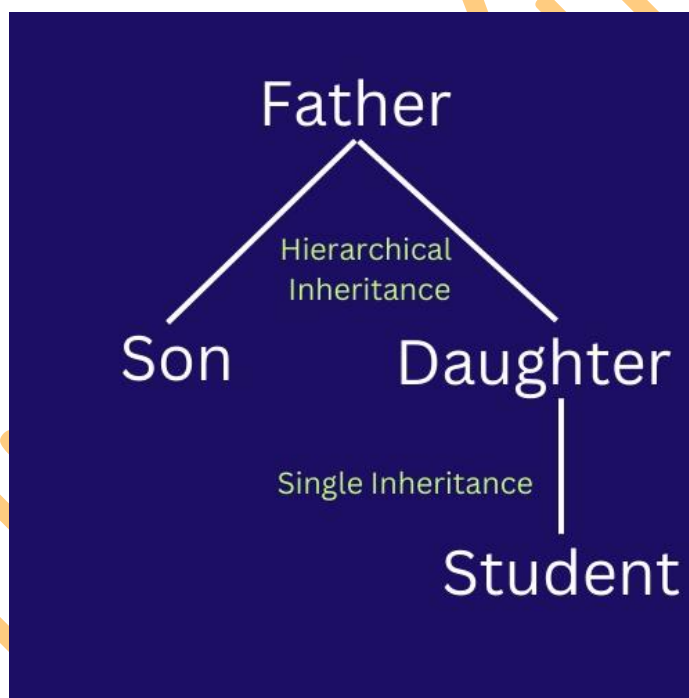
    class hie{
    public static void main(String
args[]){
    C ob1 =new C();
    B ob2=new B();
    ob1.display2();
    ob2.display1();

```

```
}  
}
```

Hybrid Inheritance

- It is a collection of all the above inheritance except multiple.



Variable Overriding

- If the variable in the parent class has the same name as that of the variable in the child class then the child class variable

tends to high the parent class variable in the child class. This mechanism is called Variable Overriding.

- Inorder to view the hidden parent class variable in the child class with the help "Super" keyword.

Write a program to display Variable overriding in Java?

```
class A{
    int a=10;
}
class B extends A
{
    int a=20;
    void disp(){
        System.out.println(a);//20
        System.out.println(super.a);//10
    }
}
class C{
    public static void main(String
ab[]){
    B ob =new B();
```

```
ob.disp();  
}  
}
```

Using Variable overriding java supports the concept of dynamic polymorphism.

Method Overriding

- If a base class variable has a method with the same name as that of its derived class then the methods in the derived class tends to hide the methods of base class in the derived class. This mechanism is called method overriding and the methods are called overridden methods.
- In order to access the parent class hidden methods in the child class we require "super" keywords.

**Write a program to display
method overriding in Java?**

```

class parent{
    int a=15,b=30;
    void display(){
        int c=a+b;
        System.out.println("The sum
is"+c);
    }
}

class child extends parent{
    void display(){
        super.display();//calling the
method display from parent class
        int d=a*b;
        System.out.println("The product
is"+d);
    }
}

class M_overrid{
    public static void main(String
args[]){
        child ob=new child();
        ob.display();
    }
}

```

What is the use of "Super" Key word?

- If the parent class member is hidden or overridden by a child class member in order to access the hidden parent class member in the child class, So we require "super" Key word.

What is the difference b/w method overloading and method Overriding?

Method Overloading.

- Here all the methods belong to the same class.
- In method Overloading argument should mismatch either typewise or no. wise.

Method Overriding.

- Here the methods belong to the parent or child class.

- In method overriding, the methods in the parents or child class will have same prototype.

Use of super method

- A parent class constructor is not inherited in its child class.
- Inorder to access the parent class constructor from the child class we require the help of super method.
- Rules for using super method
- A super method is called should be present in the child class constructor.
- The super method call should be the first statement in the child class.

Write a program to display super method?

```
class A{  
    int x,y;  
    A(int a,int b){
```

```
x=a;
y=b;
}
void disp1(){
System.out.println(x+" "+y);
}
}
class B extends A{
int c,d;
B(int p,int q,int r,int s){
super(p,q);
c=r;
d=s;
}
void disp2(){
System.out.println(c+" "+d);
}
}
class super_const{
public static void main(String
args[]){
B ob =new B(10,20,30,40);
ob.disp1();
ob.disp2();
}
}
```

Use of final key word

- The final key word is use to create a variable acting like a constant.
- If the final keyword is given prior to variable declaration then the value of the variable will not change through out the program.

Syntax:

```
final datatype variable_name=value;
```

```
final double pi=3.1415;
```

To prevent Method Overriding

- If we use the final keyword while defining a method in the parent class then that method will not overridden in the child class.

Syntax:-

```
final returntype methodname(argumentlist){
```

.....

.....

}

class T1{

final void disp(){

.....

}

}

class T2 extends T1{

void disp()//Error

{

.....

}

}

- To prevent inheritance of a class.
- If we use final keyword prior to a class name then that class will not be inherited by any child class.

Syntax:-

```
final class classname{  
.....  
}  
  
final class A{  
.....  
}  
  
class B extends A //Error  
{  
.....  
}
```

Abstract Class

- An Abstract class is a class which has abstract keyword applied to it.
- An Abstract class may contain normal methods as well as abstract method.
- Abstract method is the method which does not have body.
- We cannot create an object of the abstract class.
- An abstract class always act as a super class.
- The child class which extend the abstract class has to provide body to all the abstract methods present in the abstract class.

Write a program to display use of abstract class?

```
abstract class Parent{  
    abstract void  
display1();//abstract method
```

```
void display2()//normal method
{
    System.out.println("Testing for
abstract class");
}
}
class Child extends Parent{
void display1(){
    System.out.println("This body is
provided by child class");
}
void display3(){
    System.out.println("Child");
}
}
class Abst_Testing{
public static void main(String
a[]){
    Child t=new Child();
    t.display1();
    t.display2();
    t.display3();
}
}
```

Dynamic Method Dispatch

- A parent class reference variable can referred to any of the child class object but the vice versa is not true.
- Dynamic method dispatch is a mechanism where calls two overridden methods are resolve at run time.
- Using Dynamic method dispatch Java supports the concept of run time polymorphism.

Write a program to display dynamic method dispatch?

```
//Example of run time
//polymorphism done by method
//overriding(DMD)
class parent{
    void display(){
        System.out.println("This is
parent class method");
    }
}
```

```
class Child1 extends parent{
void display(){
System.out.println("This is the
Child1 class method");
}
}
class Child2 extends parent{
void display(){
System.out.println("This is
Child2 class method");
}
}
class DMD{
public static void main(String
args[]){
parent p=new parent();
p.display();
Child1 ob1=new Child1();
Child2 ob2=new Child2();
ob1.display();
ob2.display();
p=ob1;//the object child1 has
been referred
//By parents reference variable p
p.display();
p=ob2;//the object child has been
referred
```

```
//by parent reference variable P  
p.display();  
}  
}
```

In Java polymorphism can be classified into two types:-

- Compile time Polymorphism
- Run time Polymorphism
- A compile time polymorphism is known as early binding.
- A compile time polymorphism is achieved by concepts like methods overloading, constructor overloading;
- A Run time polymorphism is also called as late binding.
- A run time polymorphism is achieved by using dynamic method dispatch concepts like method overriding.

Disadvantage of Abstract Class

- As we can keep a normal method in abstract class hence an abstract class does not provide a blue print of a particular code. So in order to create a pure blue print and also to support the concept of multiple inheritance indirectly, A new concept is introduced Interface.

Interface

- An Interface is a special type of class which may contain variables as well as methods.
- In an interface all variables are by default static and final.
- In an Interface the methods are by default public and abstract.
- An interface is actually implemented by a class.

Syntax:-

```
interface interfacename{
```

static and final variable declaration

public and abstract methods

}

Syntax of a class implementing interface

class classname implements

interfacename1,interfacename2,.....

{

Body of the class

}

We can not create an interface of an object.

**Write a program to display
use of interface?**

```
interface i1{  
    void display1();  
}  
interface i2{  
    void display2();  
}
```



```

class Inter implements i1,i2{
    public void display1(){
        System.out.println("Display for
interface1");
    }
    public void display2(){
        System.out.println("Display for
interface2");
    }
}
class Interx{
    public static void main(String
args[]){
        i1 ob1=new Inter();//creating
object for class Inter &
//
assigning it to interface i1
        ob1.display1();
        i2 ob2=new Inter();
        ob2.display2();
    }
}

```

Garbage Collection

- In Java unreferenced objects are called as garbage .When an object does not have any reference then that object is considered to be no longer in use.
- JVM keeps track of such type of unreferenced object and remove them from memory automatically.
- This mechanism of removal of unreferenced objects by JVM is called garbage collection.
- The time of garbage collection can not be predicted in advance.

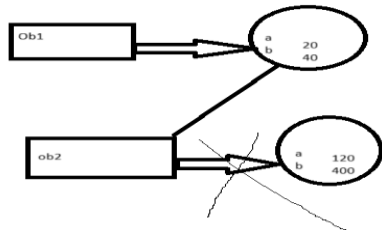
Note:-

- A particular object in Java can have more than one reference.
- But, a single reference variable can not point to two object at same time.

Program to display creation of Garbage in Java?

```
class Test1{
    int a,b;//instance variable
    void input(int x,int y){
        a=x;
        b=y;
    }
    void disp(){
        System.out.println("The values
are"+a+" "+b);
    }
}

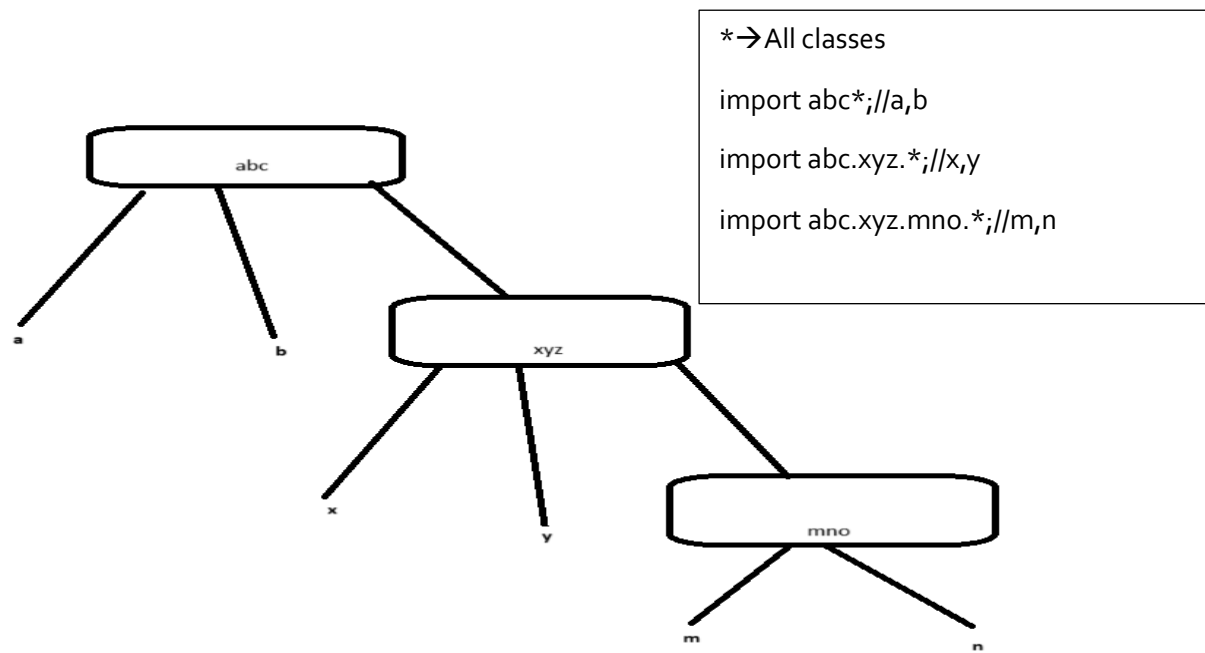
class Test2{
    public static void main(String
args[]){
        Test1 ob1=new Test1();
        Test1 ob2=new Test1();
        ob1.input(20,40);
        ob2.input(120,400);
        ob2=ob1;//Assigning one object to
another
        ob1.disp();
        ob2.disp();
    }
}
```



Garbage

Package

- A package is a collection of classes, interfaces and other packages.
- A package is used for mainly two reasons:-
 - ✓ Code reusability.
 - ✓ Security of the code.








Inorder to get all the six classes, I have to write these statements:

```
import abc.*; //a,b
import abc.xyz.*; //x,y
import abc.xyz.mno.*; //m,n
import abc.xyz.y; //y
```

Types of package Access

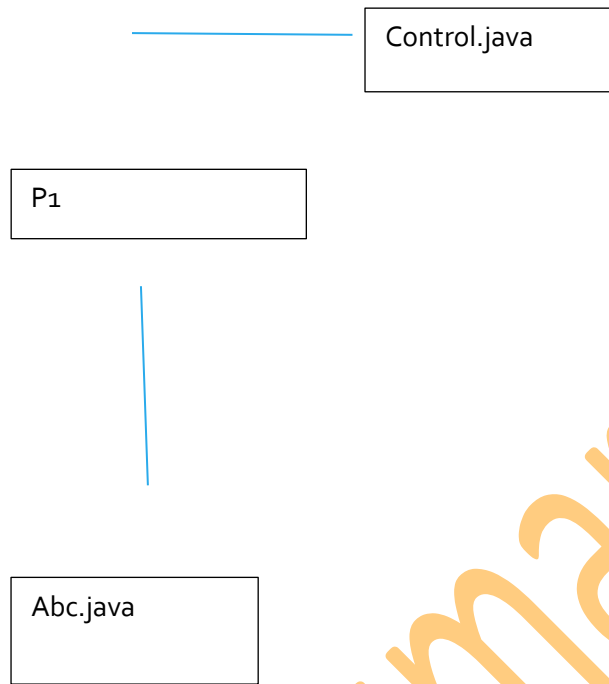
In Java package access can be classified into five types:-

-  Same package Same class
-  Same package Sub class
-  Same Package Non-sub class
-  Different Package Sub Class
-  Different Packege Non-sub Class

Same package Same class

SPSC



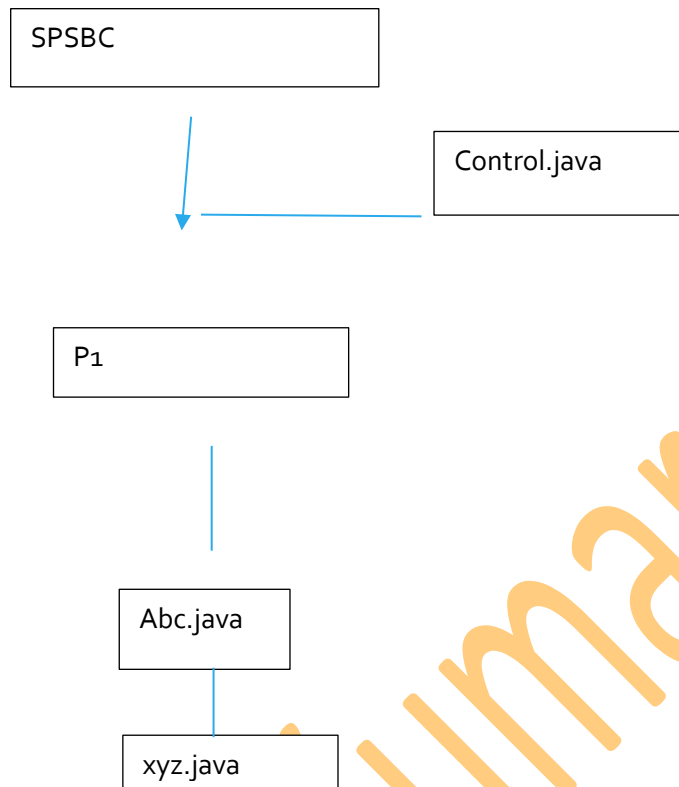


In case of same package same class all the variables no matters whatever access specifier they have are accessible.

In Java there are four access specifiers:-

- Private
- Public
- Protected
- Default

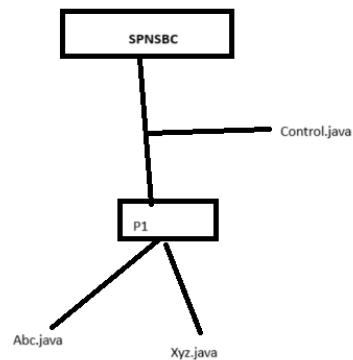
Same package Sub class



In case of same package sub class private variable is not accessible because a private variable is not inherited to the child class.

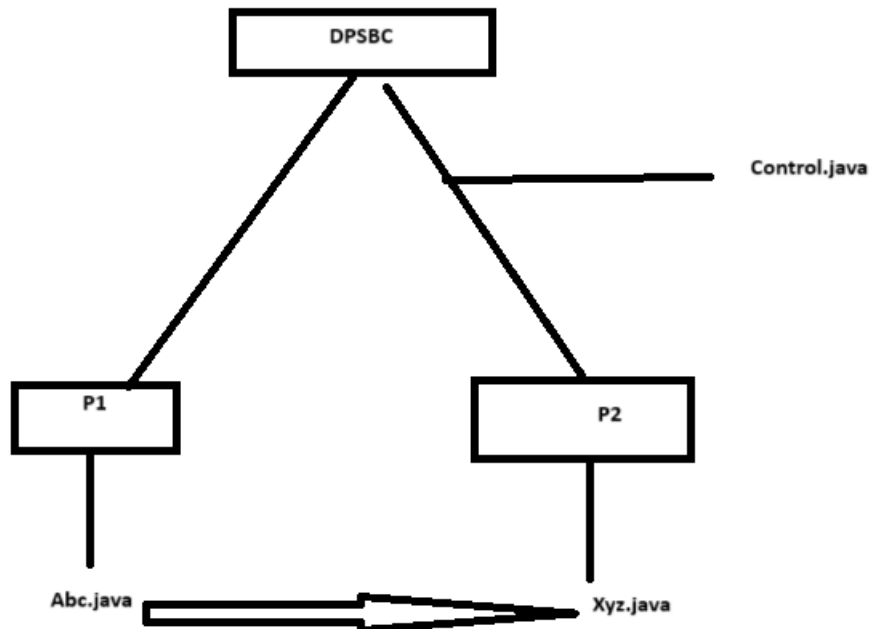
Same Package Non Sub Class

Same Package Non Sub Class



In case of Same package Non sub class private members will not access.

Different package Sub Class



1. If a variable has private access then that variable is only accessible by the function which belongs to the same class.
2. If a variable has protected access then it will be accessible by the function if and only if any one of these two situations are true:-
 - The class which contains the variables and the class which contains the methods both belongs to same package.
 - There is a parent child relationship b/w the class which contains the variables and the class which contains the method.
3. A variable which has public access is accessible from anywhere inside the code.
4. A variable which does not have any access specifiers applied to it is said to have default access or package access.

Those variables are only accessible by methods if the class which contains the variable and the class which contains the method.






Exception Handling

In any programming language there are four types of error possible:-

- Logical Error
 - Symmentic Error
 - Runtime Error
 - Compile time Error
-
- When an exception or run time error occurs in Java It is the JVM which by default handle the exception.
 - JVM only knows to stop the program on where run time error occurs.
 - The JVM stops that execution of program at that line which generates the exception.
 - To prevent this from happening, We as a coder needs to handle the exception on

our code. That's why it is known as
Exception handling

An Exception can be handled by using 5
keywords.

-  try
-  catch
-  throw
-  throws
-  finally

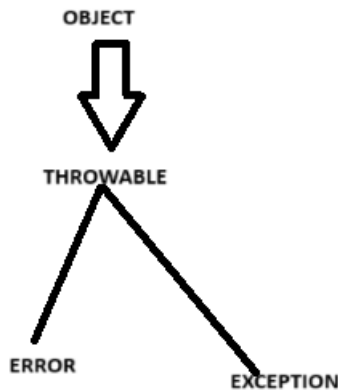
try

- try is a block which contains those statement which may or may not generates exception.
- A try block should always be follows atleast one catch block.
- When an exception occurs in the try block then the program control comes out the the try block and goes to the matching catch block.

- It is inside the catch block where we have to write exception handling code.

Write a program to display try and catch block?

```
class try_catch{
    public static void main(String
args[]){
    int x=12;
    int y=0;
    try{
    int z=x/y;
    System.out.println("The result is
+z);
    }
    catch(Exception e){
    System.out.println(e);
    }
    }
}
```



The Error class is responsible for handling any run time error happening due to hardware issue.

The Exception class is responsible for handling any kind of run time error happening due to software issue.

Syntax try and catch:-

```
try{  
.....  
}  
  
catch(Exception_classname reference name){  
.....  
}
```

A single try block may be followed by multiple catch block.

When an exception occurs in the try block, then based on the nature of the exception appropriate catch blocks are called.

Write a program to display multiple catch against a single try block?

```
class try_mul_catch{
public static void main(String
args[]){
int x=12;
int y=6;
int p[]={10,4,6,2};
try{
int z=x/y;
System.out.println("The result
is"+z);
p[17]=10;
}
catch(Exception e){
System.out.println(e);
}
```

```
}  
catch(ArithmeticException e){  
System.out.println("This results to  
infinity");  
}  
}  
}
```

Note

In case of multiple catch against the single child the catch block which contain the exception class should always be written in last, otherwise it give syntactical error.

Use of throw

Sometimes in Java we need to generates an exception manually. This can be achieved using the help of throw keyword.

Syntax of throw:-

```
throw new any_exception _class_name();
```


Program to display throw keyword?

```
class Throweg{
void check(String s1){
char ch;
ch=s1.charAt(0);
if(ch=='A'){
System.out.println("valid ID");
}
else{
try{
throw new ArithmeticException();
//generating an exception
}
catch(ArithmeticException e){
System.out.println(e);
}
}
}
}

class TCT{
public static void main(String
args[]){
Throweg ob=new Throweg();
ob.check(" ");
}
```

```
}  
}
```

Use of throws keyword

Some times a method which is capable of generating an exception does not handle the exception on its own.

In such case inorder to inform such behavior to all the other calling method, It uses the throws keyword.

Syntax for throws:-

```
class classname{  
    returntype  
    methodname(argument_list)throws  
    exception1,exception2,.....  
    .....  
}
```

```
}
```

Write a program to display use of throws keywords?

```
import java.io.*;
class Throwseg{
public static void main(String
args[]) throws IOException{
DataInputStream ds=new
DataInputStream(System.in);
System.out.println("Input 3 nos:");
int
a=Integer.parseInt(ds.readLine());
float
b=Float.parseFloat(ds.readLine());
double
c=Double.parseDouble(ds.readLine());
System.out.println("Enter a string");
String ch=ds.readLine();
System.out.println("string="+ch);
System.out.println("3 nos="+a+" "+b+"
"+c);
}
}
```

Write a program to accepts three number from the user and find their sum and average using concept of classes and object?

```
import java.io.*;
class find{
int a,b;
void sum(int x,int y){
a=x+y;
}
void average(int x,int y){
b=(x+y)/2;
}
void display(){
System.out.println("Sum is:-"+a);
System.out.println("Average is:-"+b);
}
}
class soln{
public static void main(String
args[])throws IOException{
DataInputStream ds=new
DataInputStream(System.in);
System.out.println("Please Enter 1st
no.");
```

```
int
x=Integer.parseInt(ds.readLine());
System.out.println("Please Enter 2nd
number");
int
y=Integer.parseInt(ds.readLine());
find ob=new find();
ob.sum(x,y);
ob.average(x,y);
ob.display();
}
}
```

Use of finally keyword

Finally is a block which is always executed whether an exception occur or not.

Finally block is always written after try and catch block.

Syntax:-

```
try{
```

```
.....
```

```
.....  
  
.....  
  
.....  
}  
catch(){  
.....  
}  
finally{  
.....  
}
```

**Write a program to display
finally keyword?**

```
class try_catch{  
public static void main(String  
args[]){  
int x=12;  
int y=0;  
try{
```

```
int z=x/y;
System.out.println("The result is
"+z);
}
catch(ArithmeticException e){
System.out.println("Sorry divison is
not possible");
}
finally{
System.out.println("Thankyou");
}
}
}
```

User defined Exception:-

In java we can create our own exception class provided that exception class extends the Exception class.

Write a program to display user defined exception?

```
class nomatch extends Exception{
}
class Throweg{
void check(String s1)throws nomatch{
char ch;
ch=s1.charAt(0);
if(ch=='M'){
System.out.println("valid Id");
}
else{
throw new nomatch();
}
}
}
class use_exp{
public static void main(String
args[]){
Throweg ob=new Throweg();
try{
ob.check(args[0]);
}
catch(nomatch e){
```



```
System.out.println("Invalid id");
}
finally{
System.out.println("Thanks");
}
}
}
```

Thread In Java

- ✚ In Java thread, thread is used in order to support multitasking.
- ✚ Using thread we optimized the performance of the CPU.
- ✚ In Java a thread is created using a class called thread.
- ✚ Every thread contains a name and priority level.
- ✚ The priority level of thread ranges from one to ten where the one is the minimum and the ten will be the

maximum. If we does not mention the priority level of a thread the default priority level is five.

- ✚ This name priority level combination is unique for each thread.
- ✚ Main itself is a thread.
- ✚ In any java program the main thread is also called as thread zero and the priority of main thread is five.

Write a program to display main thread in java?

```
class thread_eg{  
public static void main(String  
args[]){  
Thread ob=new Thread();  
ob.currentThread();  
System.out.println(ob);  
ob.setName("Ujjawal");  
ob.setPriority(10);  
System.out.println(ob);  
}  
}
```

Various Ways for creating a thread?

A thread can be created in two ways:-

- ❖ By implementing Runnable interface.
- ❖ By extending the Thread class.

Steps for creating a thread By implementing Runnable interface:-

Step1: create a class which implements the Runnable interface

Step2: Create an object of the thread class.

Step3: call the start method.

Step4: The start method calls the run method. It is inside the run method where we have to write the code for another thread.

Program to display creation of thread By implementing Runnable interface?

```
class Thread1 implements
Runnable//step1
{
public void run()//step 4
{
for(int i=0;i<10;i++){
System.out.println("Thread is i="+i);
}
}
}
class Thread2{
public static void main(String
args[]){
Thread1 ob=new Thread1();
Thread t=new Thread(ob);//step 2
t.start();//step3
for(int i=0;i<10;i++){
System.out.println("Thread main's
i="+i);
}
}
```

```
}
```

e.g. Operating System

What is basically a thread?

A thread is a smallest unit of a process which can run independently.

Write a program to create four threads including main by implementing runnable interface?

```
class Thread1 implements
Runnable//step1
{
public void run()//step 4
{
for(int i=0;i<3;i++){
System.out.println("Thread1 is
i="+i);
}
```

```
}  
}  
class Thread2 implements  
Runnable//step1  
{  
public void run()//step 4  
{  
for(int i=0;i<3;i++){  
System.out.println("Thread2 is  
i="+i);  
}  
}  
}  
class Thread3 implements  
Runnable//step1  
{  
public void run()//step 4  
{  
for(int i=0;i<3;i++){  
System.out.println("Thread3 is  
i="+i);  
}  
}  
}  
  
class Thread4{
```

```
public static void main(String
args[]){
Thread1 ob1=new Thread1();
Thread2 ob2=new Thread2();
Thread3 ob3=new Thread3();
Thread t1=new Thread(ob1); //step 2
Thread t2=new Thread(ob2);
Thread t3=new Thread(ob3);
t1.start(); //step3
t2.start();
t3.start();
for(int i=0;i<3;i++){
System.out.println("Thread main's
i="+i);
}
}
}
```

**Program to create a thread
by implementing runnable
interface but in a different
way?**

```
class Thread1 implements Runnable{
```

```
Thread1(){
Thread t=new Thread(this);
t.start();
}
public void run(){
for(int i=0;i<10;i++){
System.out.println("Thread 1 is
i="+i);
}
}
}
}
class Thread2{
public static void main(String
args[]){
new Thread1();
for(int i=0;i<=20;i++){
System.out.println("Main i="+i);
}
}
}
}
```

Steps for Creating a thread by extending the thread class?

Step1: Create a class which extend the thread class.

Step2: call the start method.

Step3:the start method calls the run method and it is inside the run method where we have to write the code for another thread.

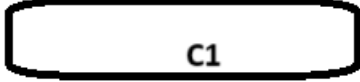
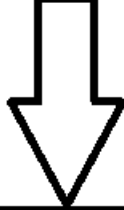
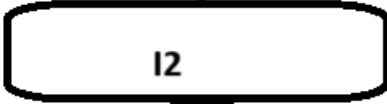
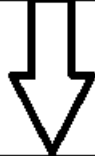
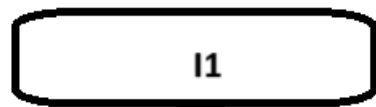
Program to display creation of the thread by extending the thread class?

```
class A1 extends Thread//step1
{
A1(){
start();//step2
}
public void run()//step3
{
for(int i=0;i<10;i++){
System.out.println("A1 is i="+i);
}
}
}
class Thread2{
public static void main(String
args[]){
```

```
new A1();  
for(int i=0;i<=20;i++){  
System.out.println("Main i="+i);  
}  
}  
}
```

Among this two ways of creating a thread which one is the best and why?

- If we create a thread by extending the thread class then that class will not be able to extend any other class if needed. Because Java does not support multiple inheritance.
- But we create by implementing the runnable interface then that class still has the option to extend any other class if needed.

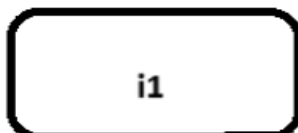


`void disp1();`

`interface I2 extends I1`

`{`

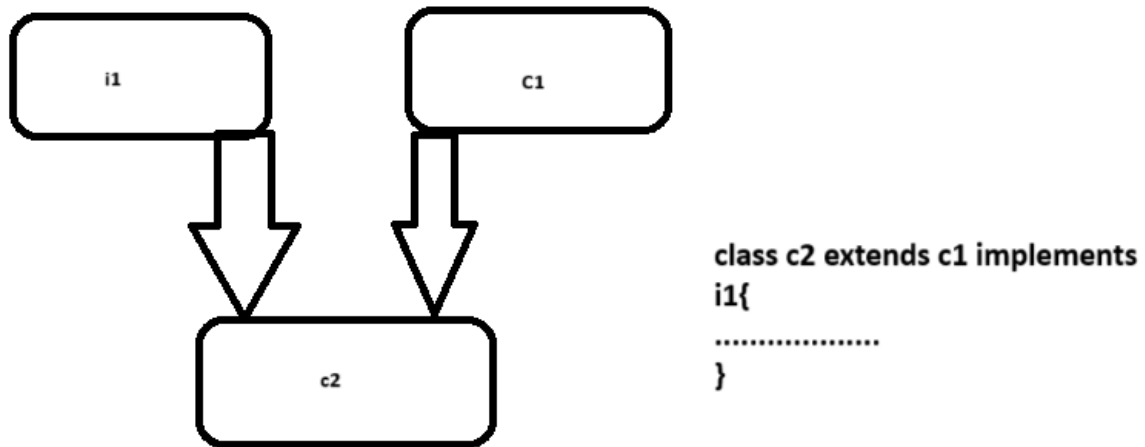
`}`



`interface i3 extends i1`

`.....`

`}`



Use of sleep in thread method

In Java there is a method in thread class sleep. Using which we can momentarily paused the execution of thread.

Takes Argument in the form of millisecond.

Write a program to display use of sleep method?

```
class Test{  
public static void main(String a[]){  
for(int i=1;i<=10;i++){  
try{  
System.out.println(i);  

```

```
Thread.sleep(2000);  
}  
catch(InterruptedException e){  
}  
}  
}  
}
```

Or

```
class Test{  
public static void main(String a[]){  
for(int i=1;i<=10;i++){  
try{  
System.out.println(i);  
Thread.sleep(2000);  
}  
catch(InterruptedException e){  
}  
}  
}  
}
```

Synchronization of a thread

- Sometimes when a common code is shared by multiple thread, then the output will be ambiguous in nature.
- To prevent this from happening we as a coder needs to synchronized the common code.
- If we synchronized the sharable code then until and unless a thread has already finished the operation with sharable thread.
- No other thread are allow to access the sharable thread.
- This mechanism is called Synchronization of a thread.

Write a program to display Synchronized thread?

```
class disp{  
void display(String s1)//common code  
{
```

```
System.out.println("Mr");
System.out.println(" "+s1);
System.out.println("Roy");
}
}
class syn implements Runnable{
    disp ob;
    String s1;
    syn(disp ob1,String s2){
        ob=ob1;
        s1=s2;
        Thread t=new Thread(this,s1);
        t.start();
    }
    public void run(){
        //synchronized(ob)
        //{
        ob.display(s1);
        //}
    }
}
class Synchronize{
    public static void main(String a[]){
        disp ob=new disp();
        new syn(ob,"Ayan");
        new syn(ob,"Sourav");
        new syn(ob,"Amit");
    }
}
```

```
}  
}
```

Various States of Thread

- New
- Active
- Running
- Blocked
- Terminated

New

This is the state where a thread is first created.

e.g.-Creating object of the thread class.

Thread ob=new Thread();

Active

This is the state where a thread is ready to run.

e.g When we call the start method.

Running

In running method means here the thread is actually running.

e.g -Execution of run method.

Blocked

The block state where execution of the thread is blocked or paused.

A block state can be achieved either by wait method or by sleep method.

Terminated

This is the state where the execution of thread is complete.

Using thread we maximized the CPU performance.

String in Java

A string may be defined as a sequence of characters which is terminated by a null character('\0').

In Java String is implemented by two classes:-

- String
- StringBuffer

String Class

- A String class is the immutable class.
- Any changes made to the existing String object actually creates a fresh new object.
- This String class contains a lot of methods using which we can manipulate a particular string.

length()

the length method is responsible for determining the total number of characters present in the string.

e.g.:-

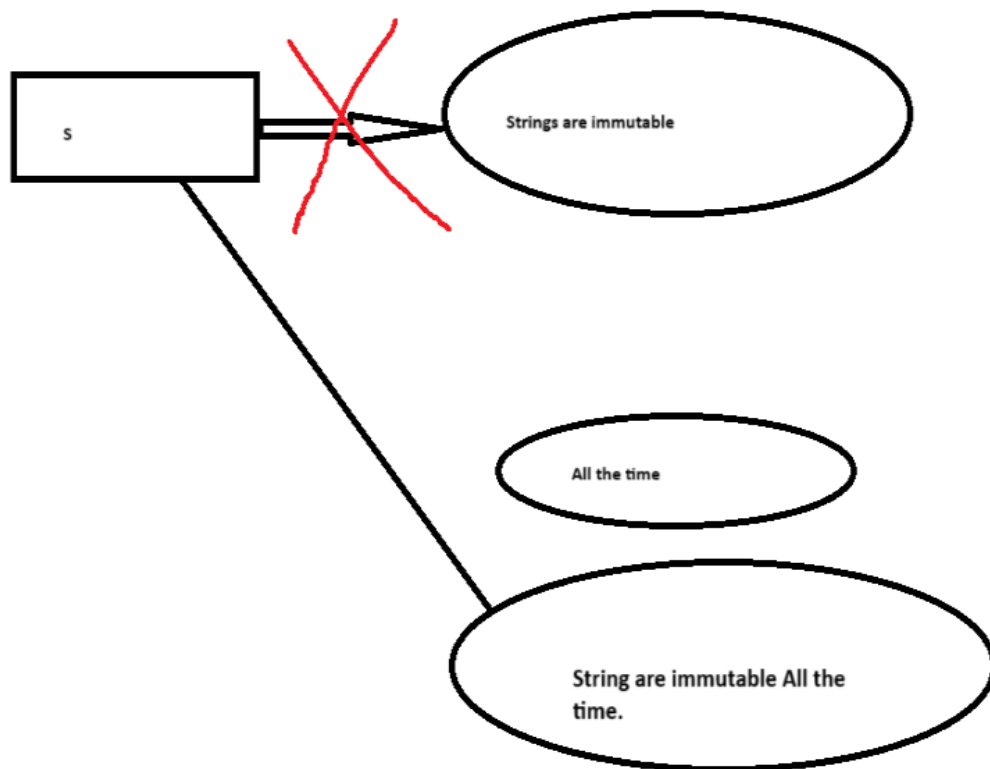
```
class StringDemo{
public static void main(String
args[]){
String str="Hello how r u";
int len=str.length();
System.out.println("String length
is:"+len);
}
}
```

concat()

This method is responsible for concatenating two strings together.

```
class Test{
public static void main(String
args[]){
String s="Strings are immutable";
```

```
s=s.concat(" all the time");  
System.out.println(s);  
}  
}
```



charAt()

charAt method is responsible for extracting a particular character from a given string based on the index position.

Program to display charAt method?

```
class Test{
```

```
public static void main(String  
args[]){  
String s="Strings are immutable";  
char result=s.charAt(8);  
System.out.println(result);  
}  
}
```

compareTo()

- ✚ This method is used to compare two strings together and return the ascii difference of the mismatch character.
- ✚ This functions returns zero if both the strings are equal.

**Write a program to display
compareTo function?**

```
class Test{  
public static void main(String  
args[]){  
String str1="Strings are immutable";  
String str2=new String("Strings are  
immutable");
```

```
String str3=new String("Integers are
not immutable");
int result=str1.compareTo(str2);
System.out.println(result);
result=str2.compareTo(str3);
System.out.println(result);
}
}
```

endsWith()

- ✚ This method is used to check whether a particular string ends with the given string or not.
- ✚ This method returns a Boolean value.

```
class Test{
public static void main(String
args[]){
String str=new String("This is really
not immutable!!");
boolean retval;
retval=str.endsWith("immutable!!");
System.out.println("returned
value="+retval);
retval=str.endsWith("immu");
```

```
System.out.println("returned  
value="+retval);  
}  
}
```

startsWith()

- ✚ The startsWith method is used to check whether a particular string start with given String.
- ✚ It also return Boolean value.

```
class Test{  
public static void main(String  
args[]){  
String str=new String("This is really  
not immutable!!");  
boolean retval;  
retval=str.startsWith("This is");  
System.out.println("returned  
value="+retval);  
retval=str.startsWith("immu");  
System.out.println("returned  
value="+retval);  
}  
}
```

Use of equals method

- This method is used to compare two strings and returns Boolean value.
- Equal method is case Sensitive in nature.

Write a program to compare two strings?

```
class Test{
public static void main(String
args[]){
String str1="Strings are immutable";
String str2=new String("Strings are
immutable");
String str3=new String("Integers are
not immutable");
boolean result=str1.equals(str2);
System.out.println(result);
result=str2.equals(str3);
System.out.println(result);
}
}
```


equalsIgnore()

It is used to compare two strings by ignoring the case.

```
class Test{
public static void main(String
args[]){
String str1="UJJAWAL";
String str2=new String("ujjawal");
boolean
result=str1.equalsIgnore(str2);
System.out.println(result);
result=str2.equalsIgnore(str3);
System.out.println(result);
}
}
```

indexOf()

indexOf method is used to determine the index position of the string.

This method returns the index within this string of the first occurrence of the specified character or -1, if the character does not occur.

```
class Test{
public static void main(String
args[]){
String str=new String("Ujjawal");
System.out.print("Found at:");
System.out.print(str.indexOf('a'));
}
}
```

lastIndexOf()

The lastIndexOf is used to determine the index of last character from right hand side.

```
class Test{
public static void main(String
args[]){
String str=new String("Ujjawal");
System.out.print("Found at:");
System.out.print(str.lastIndexOf('j')
);
}
}
```

replace()

The replace method is used to replace a particular character in a string with another character.

```
class Test{
public static void main(String
args[]){
String str=new String("Ujjawal
Kumar");
System.out.print("Return value:");
System.out.print(str.replace('l','L')
);
}
}
```

substring()

The substring method is used to extract a particular string from a given string.

Code:-

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

```
String str=new String("Ujjawal  
Kumar");  
System.out.print("Return value:");  
System.out.print(str.substring(0,4));  
}  
}
```

toLowerCase()

this method is used to convert an upper case string into corresponding lower case.

```
class Test{  
public static void main(String  
args[]){  
String str=new String("UJJAWAL  
KUMAR");  
System.out.print("Return value:");  
System.out.print(str.toLowerCase());  
}  
}
```

toUpperCase()

this method is used to convert an lower case string into corresponding upper case.

```
class Test{
public static void main(String
args[]){
String str=new String("ujjawal
kumar");
System.out.print("Return value:");
System.out.print(str.toUpperCase());
}
}
```

trim()

The trim method is used to eliminate any leading or trailing spaces in string.

```
class Test{
public static void main(String
args[]){
String str=new
String("      ujjawal kumar");
System.out.print("Return value:");
System.out.print(str.trim() );
}
}
```

StringBuffer

- StringBuffer is also used in order to manipulate a string in Java.
- Unlike a String class, StringBuffer is a mutable class.
- This StringBuffer class also contains methods for modifying a string.

append()

The append method is responsible for concatenating two strings.

```
class Test{
public static void main(String
args[]){
StringBuffer str=new
StringBuffer("hello");
str.append("Java");
System.out.print(str);
}
}
```

Use of insert()

The insert method is used to insert a particular string in a given string at particular index.

```
class Test{
public static void main(String
args[]){
StringBuffer str=new
StringBuffer("hello ");
str.insert(1,"Java");
System.out.print(str);
}
}
```

Use of replace()

The replace method is used to replace a particular character in a string with another character.

Code:-

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

```
StringBuffer str=new  
StringBuffer("Ujjawal Kumar");  
System.out.print("Return value:");  
System.out.print(str.replace(8,12,"Ga  
va"));  
}  
}
```

Use of delete()

The delete method is used to delete a sequence of character from a given string.

```
class Test{  
public static void main(String  
args[]){  
StringBuffer str=new  
StringBuffer("Ujjawal Kumar");  
str.delete(1,3);  
System.out.print(str);  
}  
}
```


reverse()

The reverse method is used to reverse the content of string .

Code:-

```
class Test{
public static void main(String
args[]){
StringBuffer str=new
StringBuffer("Ujjawal Kumar");
str.reverse();
System.out.print(str);
}
}
```

Applet

- Applet is used in order to implement graphics in java.
- An applet is a program that is typically embedded in a web page and can be run from a browser.
- We need special HTML in the Web page to tell the browser about applet.
- We don't need to supply a main method; the browser does that
- When we write an applet, we are writing only part of a program.
- We supply certain methods that the browser calls.

The hierarchy of Applet class

java.lang.Object

+-----java.awt.component

|

+-----java.awt.Container

```
|
+-----java.awt.Panel
|
+-----java.applet.Applet
|
+-----
java.swing.JApplet
D:\java prog\Applet>appletviewer index.html
```

Program to display a string in an applet.

```
import java.awt.*;
import java.applet.Applet;
public class HelloWorld extends Applet{
public void paint(Graphics g){
g.drawString("Hello World!",30,30);
```

```
}  
  
}
```

Html File linking HelloWorld Class

```
<html>  
  
<applet code="HelloWorld.class"  
height="400" width="500">  
  
</applet>  
  
</html>
```

For any Applet program two packages need to be imported.

```
import java.awt.*;  
import java.applet.Applet;
```

Graphics is inbuilt class of the paint method which is responsible for displaying anything inside an applet.

This Graphics class contains a method called `drawstring` using which we can display a String in an Applet.

Write A program to draw a line in Applet?

drawLine()

To draw a line in Applet drawLine method is used.

e.g. `drawLine(x1,y1,x2,y2);`

To draw a rectangle

`drawRect(x,y,width,height)`

x&y represents the top left point of the rectangle.

`fillRect(x,y,width,height);`

```
drawOval(x,y,width,height);
```

Colors

The java.awt package defines a class named Color.

There are predefined colors-here are their fully qualified names:-

Color.BLACK

Color.PINK

Color.GREEN

Color.DARK_GRAY

Color.RED

Color.CYAN

Color.GRAY

Color.ORANGE

Color.BLUE

Color.LIGHT_GRAY

Color.YELLOW

Color.WHITE

Color.MAGENTA

New colors

To provide custom colours in applet.

Color ob=new Color(Red,Green,Blue)

Red→0 to 255

Green→0 to 255

Blue→0 to 255

Color ob=new Color(56,32,78);

g.setColor(ob)

Life Cycle of Applet

As an applet program does not have main method hence it follows a life cycle.

An applet life cycle consists of following methods.

`public void init()`

`public void start()`

`public void stop()`

`public void destroy()`

`public void paint(Graphics)`

- The init method is also called initialization method. this method is responsible for loading the method in applet memory.
- This method is called only ones in the entire life cycle of the applet.
- The start method is called just after the init method. This method is responsible for bringing the applet on the screen.

- This method can be called multiple times.
- The stop method is responsible for running the applet in the background of the screen.
- Like a start method stop method calls multiple times.
- The paint method is responsible for drawing anything in an applet.
- The paint method can also be called multiple times.
- The paint method is called using repaint method.
- The destroy method for unloading the applet from the computers memory ones its operation is over.
- Life init method the destroyer method is called only ones in the entire life cycle of applet.

Swing in Java

Swing is an advance version of Applet.

For creating a swing program we have to extends JApplet class.

Program to display use of swing?

- For any swing program the package we need to export is `javax.swing.*`;
- The Container class is responsible for providing the layout of the swing class.
- `getContentPane` method is responsible for initializing the container class.
- For creating the Label in swing using the class called `JLabel`.

- JcheckBox class is used to create checkbox in swing.

Program to create radio button using swing?

Event Handling in Java

Event

Event may be defined as an occurrence of something.

When an event occurs in java those events are actually captured what we called listeners.

This Listeners contains abstract method which we need to define.

Whenever a particular event occurs appropriate methods will be executed.

Write a program to display key event?

- For event handling the package which is required is called java.awt.event.*;
- KeyListener is an interface which is responsible for handling key related events.
- getKeyChar is a method using which we can extract a particular character from the keyboard.

Program to display use of mouse event.

```
import java.awt.*;  
import javax.swing.*;  
import java.awt.event.*;  
  
public class mouse extends JApplet  
implements MouseListener{
```

```
JLabel l;  
  
public void init(){  
    Container cp=getContentPane();  
    l=new JLabel("Ayan",JLabel.LEFT);  
    cp.add(l);  
    addMouseListener(this);  
    requestFocus();  
}  
  
public void mouseClicked(MouseEvent e){  
    l.setText("Mouse clicked");  
}  
  
public void mouseEntered(MouseEvent e){  
    l.setText("Mouse Entered");  
}  
  
public void mousePressed(MouseEvent e){  
    l.setText("Mouse Pressed");
```

```
}  
  
public void mouseReleased(MouseEvent e){  
    l.setText("Mouse released");  
}  
  
}
```

For handling mouse realated event we require mouse listener.