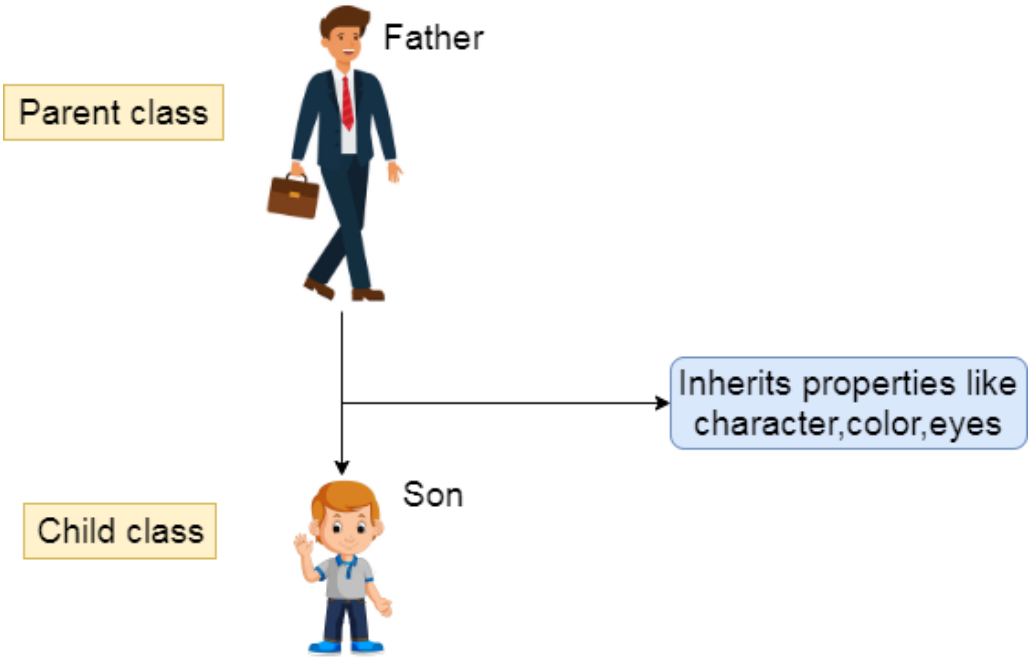
Inheritance in Java Programs

1. What is Inheritance?

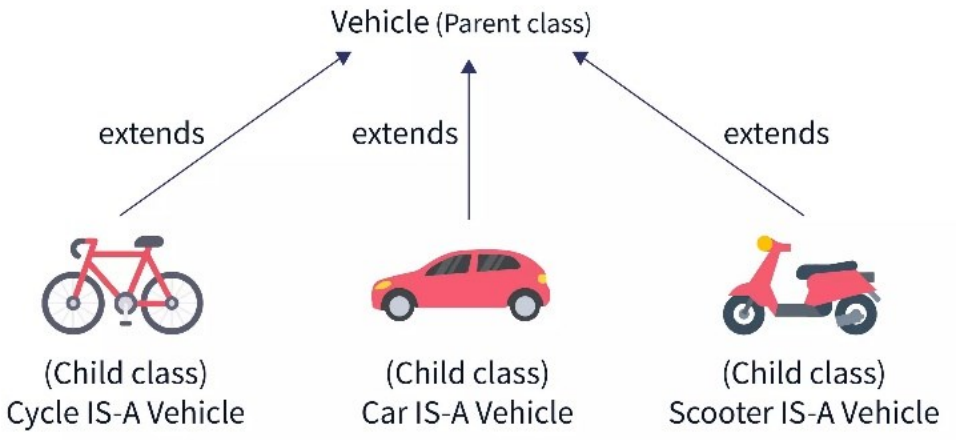
It is the capability or feature of one class so that it can inherit properties from another class.

- **Base Class:** It is the class whose properties are inherited by another class. It is also called Super Class or Parent Class.
- **Derived Class:** It is the class that inherits properties from the base class. It is also called a sub-class or child class.

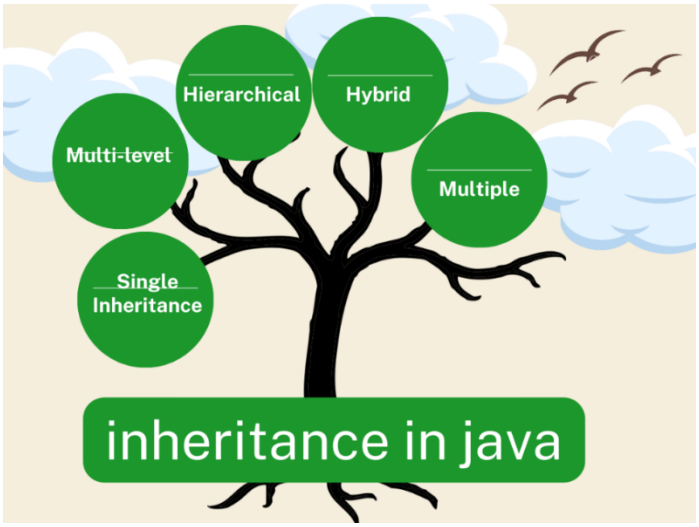
Pictorial representation of Inheritance:



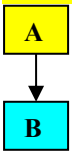
It is one of the features of OOPS, where we can reuse an existing class (parent class) with the help of a newly created class (child class). Inheritance is a form of 'IS-A' relationship.



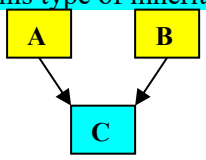
2. FORMS OF INHERITANCE



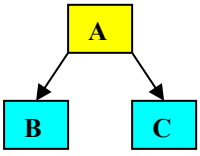
2.1 Single Inheritance: It is the inheritance hierarchy wherein one derived class inherits from one base class. In this arrangement, there is only one base class and one derived class. (Given below diagram of single inheritance where A is the base class and B is the derived class.



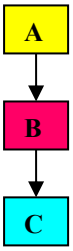
2.2 Multiple Inheritance: It is the inheritance hierarchy wherein one derived class inherits from multiple base class(es). In this arrangement, there are more than one base class and one derived class. (Given below diagram of multiple inheritance where A and B are the base classes and C is the derived class. (Note: Java does not support this type of inheritance)



2.3 Hierarchical Inheritance: It is the inheritance hierarchy wherein multiple subclasses inherit from one base class. In this arrangement, there is one base class and more than one derived class. (Given below diagram of hierarchical inheritance where A is the base class and B and C are the derived classes.

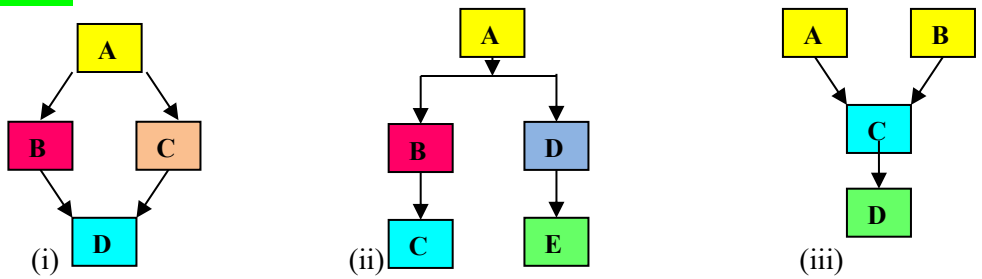


2.4 Multilevel Inheritance: It is the inheritance hierarchy wherein a subclass acts as a base class for other classes. In this arrangement, there is one base class from which one derived class is produced and from that derived class another class is derived. (Given below diagram of multilevel inheritance where A is the base class and class B is derived from class A, thereafter class C is derived from class B.



(here B is the child of Class A as well as the parent of Class C)

2.5 Hybrid Inheritance: The inheritance hierarchy that reflects any legal combination of the other four types of inheritance.



In the above diagram (i) hybrid inheritance is formed by the combination of hierarchical and multiple inheritances, similarly in the diagram (ii) hybrid inheritance is formed by the combination of hierarchical and multi-level inheritances, and in diagram (iii) hybrid inheritance is formed by the combination of multiple and multi-level inheritances. (Note: As in Java Multiple inheritance is not allowed, so, (i) and (iii) will become invalid here)

In Java programming, only three forms of inheritance are possible and they are – single inheritance, hierarchical inheritance and multilevel inheritance. Multiple inheritance is not possible in Java.

3. VISIBILITY MODE

It is the keyword that controls the visibility and availability of inherited base class members in the derived class. It can be either private, protected or public.

- **Private Inheritance:** It is the inheritance facilitated by private visibility mode. In private inheritance, the protected and public members of the base class become private members of the derived class.

- **Public Inheritance:** It is the inheritance facilitated by public visibility mode. In public inheritance, the protected members of the base class become protected members of the derived class and public members of the base class become public members of the derived class.

- **Protected Inheritance:** It is the inheritance facilitated by protected visibility mode. In protected inheritance, the protected and public members of the base class become protected members of the derived class.

Base Class Visibility	**Derived class visibility**		
	Public derivation	**Private derivation**	**Protected derivation**
Private	Not inherited	Not inherited	Not inherited
Protected	Protected	Private	Protected
Public	Public	Private	Protected

Advantages of using inheritance in Java programming.

(i) Reusability: Inheritance helps the code to be reused in many situations. The base class is defined and once it is compiled, it need not be reworked. Using the concept of inheritance, the programmer can create as many derived classes from the base class as needed while adding specific features to each derived class as needed.

(ii) Saves Time and Effort: The above concept of reusability achieved by inheritance saves the programmer time and effort. Since the main code written can be reused in various situations as needed.

(iii) Increases Program Structure which results in greater reliability.

4. INHERITANCE & CONSTRUCTOR

A constructor of the child class gets invoked first then its base class constructor is invoked when we create an object of the child class.

Execution of base class constructor

Method of inheritance	Order of execution using objects

class A
{ int x;
public A(int i)
{ x=i; }
..
public static void main( )
{
A ob=new A(10);
..
}
}
	A()//base constructor
	X=10

class B extends public A
{ int y;
public B(int i, int j)
{ super(i); //invoking base constructor
y=j;
}
..
public static void main( )
{
B ob=new B(10,15);
..
}
}
	B()//derived constructor
	A()//base constructor
	X=10
	Y=15

class C extends public B
{ int z;
public C(int i, int j, int k)
{ super(i, j); //invoking it's base class
z=k;
}
..
public static void main( )
{
C ob=new C(10,15,20);
..
}
}
	C()//derived constructor
	B()//derived constructor
	A()//base constructor
	X=10
	Y=15
	Z=20

Access Specifiers in Java

		public	private	protected	default
Same Package	Class	YES	YES	YES	YES
	Sub class	YES	NO	YES	YES
	Non sub class	YES	NO	YES	YES
Different Package	Sub class	YES	NO	YES	NO
	Non sub class	YES	NO	NO	NO

Programs to solve:

Question 1. Create two classes named Library and Issue. The class library will have details of the books. Another class Issue will inherit class Library purchases that will store the number of days late in returning the book and fine calculated.

Class: Library

Member data: bookno as Integer
Bookname as String
Studentname as String

Member function

Library(….) parameterized constructor to give initial values to member data

void display( ) to display the member data

Class: Issue (it will inherit the class Library)

Member data: daysLate as Integer
fine as Double

Member function

Issue(….) parameterized constructor to give initial values to the Superclass data members

void return_book( ) to receive the no. of days late to return the book by the student and compute the fine if no. of days is late according to the given rule otherwise no fine.

No. of days late <= 10 days **Fine of 1 rupee per day**

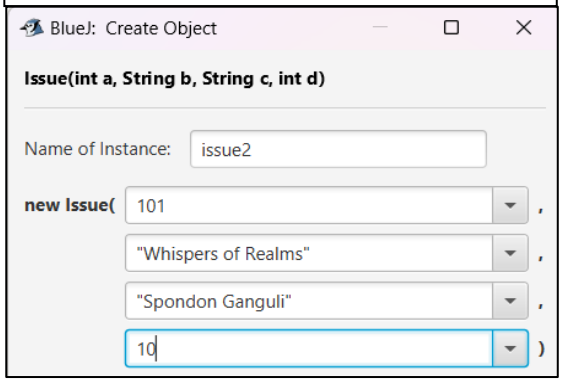
Otherwise **Fine of 1.5 rupees per day**

void display( ) to display the details of the student who issued the book along with the total amount as fine.

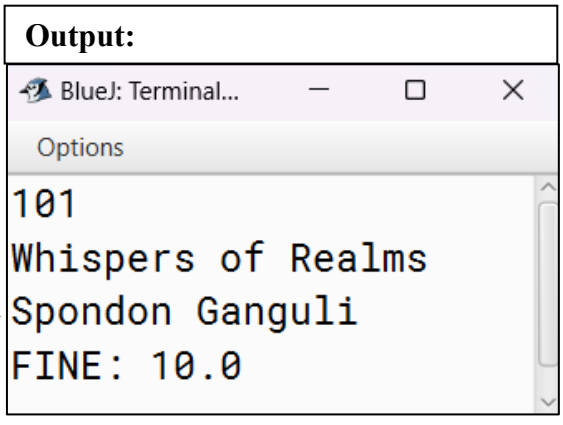
Solution:

```
class Library//base class
{
   int bookno;
   String Bookname;
   String Studentname;
   Library(int a, String b, String c)//parameterised constructor
   {
       bookno=a;
       Bookname=b;
       Studentname=c;
   }
   void display()//base class method
   {
      System.out.println(bookno+"\n"+Bookname+"\n"+Studentname);
   }
}
class Issue extends Library //Issue is the child class that is inheriting the Library class
{
   int daysLate;
   double fine;
   Issue(int a, String b, String c, int d)//parameterised constructor
   {
      super(a,b,c);//invoking the base class constructor at the time of the child class's object creation
      daysLate=d;
      fine=0.0;
   }
   void return_book()
   {
      fine=daysLate*((daysLate<=15)?1.00:1.50);
   }
   void display()//method of child class
   {
      super.display();//super keyword used to call the base class method
      System.out.println("FINE: "+fine);
   }
}
```

Input:



Output:



Question 2. Two classes Worker and Wages will compute the wages of a worker on the no. of hours worked and rate of wage using the concept of inheritance.

Class: Worker

Member data:	Name as String
	Basic as double

Member function

Worker(…)	parameterized constructor
Display( )	to display the member data

Class: Wages (it will inherit the class Worker)

Member data:	Hrs as integer
	Rate as integer
	Wage as double

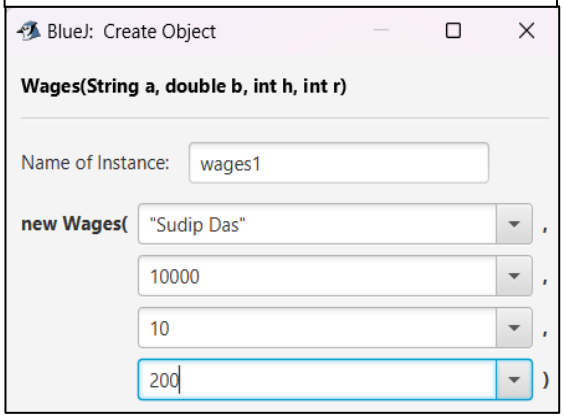
Member function

Wages(…)	parameterized constructor to give initial values to the Superclass data members
Ovrtm( )	to compute the overtime as Hrs*Rate
Display( )	to display the details of the Worker and his/her wages

Solution:

```
class Worker
{
   String Name;
   double Basic;
   public Worker(String a, double b)
   {
      Name=a;
      Basic=b;
   }
   public void display()
   {
      System.out.println("Name of the worker is "+Name);
      System.out.println("Basic pay of the worker is "+Basic);
   }
}
class Wages extends Worker
{
 int hrs, rate;
 double wage;
 public Wages(String a, double b, int h, int r)
 {
    super(a,b);
    hrs=h;
    rate=r;
    wage=0.0;
 }
 public double calcOvrtm()
 {
    double ovrtm;
    ovrtm=hrs*rate;
    return ovrtm;
 }
 public void display()
 {
    wage=calcOvrtm()+Basic;
    super.display();
    System.out.println("No. of hrs are"+hrs+"\n"+ "rate per hour is "+rate+"\n"+" wage of the worker is "+wage);
 }
}
```

Input:



Output: