**Inheritance** : It is a process of acquiring the characterstics and behavior by a subclass from the base class.

**TYPES OF INHERTANCE:**

1. Single level :

BASE CLASS

SUBCLASS

1. MULTIPLE INHERTANCE

BASE CLASS 1

BASE CLASS 2

SUBCLASS

1. HIERARCHIAL INHERITANCE:

BASE CLASS

SUB CLASS 3

SUB CLASS 2

SUB CLASS

1. MULTILEVEL INHERITANCE

X

Y

Z

1. HYBRID INHERITANCE

B

Z

Y

X

A

What is the use of **extends** keyword ?

Keyword extends is used to inherit the properties from base class by subclass

public class Base

{

int a ;

}

public class Subclass extends Bae

{

}

Q Define method overriding?

It is a process of defining a function in the sub class with the same function signature as it is defined in base class.

public class Base

{

public void showDetails( )

{

System.out.println(“Base class”);

}

}

public class Subclass extends Base

{

public void showDetails( )

{

System.out.println(“sub class”);

}

}

What is the use of super keyword ?

Super keyword is used to access properties and function , in base class from derived class ,which are hidden due to overriding by the derived class.

Q1. A superclass Number is defined to calculate the factorial of a number. Define a subclass Series to find the sum of the series S = 1! + 2! + 3! + 4! + ………. + n! [5]  
The details of the members of both classes are given below:  
Class name: Number  
Data member/instance variable:  
n: to store an integer number  
Member functions/methods:  
Number(int nn): parameterized constructor to initialize the data member n=nn  
int factorial(int a): returns the factorial of a number  
(factorial of n = 1 × 2 × 3 × …… × n)  
void display()  
Class name: Series  
Data member/instance variable:  
sum: to store the sum of the series  
Member functions/methods:  
Series(…) : parameterized constructor to initialize the data members of both the classes  
void calsum(): calculates the sum of the given series  
void display(): displays the data members of both the classes  
Assume that the superclass Number has been defined. Using the concept of inheritance, specify the class Series giving the details of the constructor(…), void calsum() and void display().  
The superclass, main function and algorithm need NOT be written.

Q2: 29-7-2022

A superclass Product has been defined to store the details of a product sold by a wholesaler to a retailer. Define a subclass Sales to compute the total amount paid by the retailer with or without fine along with service tax. [5]  
Some of the members of both classes are given below:  
Class name: Product  
Data members/instance variables:  
name: stores the name of the product  
code: integer to store the product code  
amount: stores the total sale amount of the product (in decimals)  
Member functions/methods:  
Product (String n, int c, double p): parameterized constructor to assign data members: name = n, code = c and amount = p  
void show(): displays the details of the data members  
Class name: Sales  
Data members/instance variables:  
day: stores number of days taken to pay the sale amount  
tax: to store the sen ice tax (in decimals)  
totamt: to store the total amount (in decimals)  
Member functions/methods:  
Sales(….): parameterized constructor to assign values to data members of both the classes  
void compute(): calculates the service tax @ 12.4% of the actual sale amount  
calculates the fine @ 2.5% of the actual sale amount only if the amount paid by the retailer to the wholesaler exceeds 30 days calculates the total amount paid by the retailer as (actual sale amount + service tax + fine)  
void show (): displays the data members of the superclass and the total amount  
Assume that the superclass Product has been defined. Using the concept of inheritance, specify the class Sales giving the details of the constructor (…), void compute() ) and void show(). The superclass, main function and algorithm need NOT be written.

Q3

A superclass Bank has been defined to store the details of a customer. Define a sub-class Account that enables transactions for the customer with the bank. The details of both the classes are given below: [5]  
Class name: Bank  
Data members/instance variables:  
name: stores the name of the customer  
accno: stores the account number  
P: stores the principal amount in decimals  
Member functions/methods:  
Bank(….): parameterized constructor to assign values to the instance variables  
void display (): displays the details of the customer  
Class name: Account  
Data member/instance variable:  
amt: stores the transaction amount in decimals  
Member functions/methods:  
Account(…): parameterized constructor to assign values to the instance variables of both the classes  
void deposit(): accepts the amount and updates the principal as p=p+amt  
void withdraw(): accepts the amount and updates the principal as p=p-amt  
If the withdrawal amount is more than the principal amount, then display the message “INSUFFICIENT BALANCE”.  
If the principal amount after withdrawal is less than 500, then a penalty is imposed by using the formula.  
p=p-(500-p)/10  
void display(): displays the details of the customer  
Assume that the superclass Bank has been defined.  
Using the concept of Inheritance; specify the class Account giving details of the constructor(…), void deposit(), void withdraw() and void display() The superclass and the main function need not be written.

**INTERFACE**

An interface defines a protocol of behavior . the aim of interface in Java is to dictate common behavior among objects from diverse classes.

An interface can declare only two things:

1. Abstract method : methods which do not have body and these methods have to be declared with the modifier **public**.
2. Constant: Variables which cant be modified .they have to be declared with modifier

**public static final**

Implementing an interface allows a class to become more formal about the behavior it promises to provide. Interfaces form a contract between the class and the outside world, and this contract is enforced at build time by the compiler. If your class claims to implement an interface, all methods defined by that interface must appear in its source code before the class will successfully compile.

// HOW TO CREATE INTERFACE

public interface IRobot

{

public static final String BRAND="BMW";

public void moveForward();

public void moveBackward();

}

public interface IVanshRobo

{

public void moveLeft();

public void moveRight();

}

// HOW TO IMPLEMENT INTERFACE

public class OscorpRobo implements IRobot ,IVanshRobo

{

public void moveForward()

{

System.out.println("Moving forward");

}

public void moveBackward()

{

System.out.println("Moving backward");

}

public void moveLeft()

{

System.out.println("Moving left");

}

public void moveRight()

{

System.out.println("moving right");

}

}