CONCEPT OF INHERITANCE

Type A: Very Short Answer Questions

1	**Fill in the blanks in each of the following sentences:** **(i) A method that lacks a body is an _____ method.** **(ii) An _____ is like a class except that it contains only instance methods, no instance variables.** **(iii) Two ways for a class to inherit something in java are to _____ a class or _____ are inherited by the subclasses.** **(iv) An object can refer to itself by using _____ keyword.** **(v) A _____ method is one that does different things depending upon the object that invokes it.**
Ans.	(i) A method that lacks a body is an **abstract** method. (ii) An **Interface** is like a class except that it contains only instance methods, no instance variables. (iii) Two ways for a class to inherit something in java are to **private** a class or **public** are inherited by the subclasses. (iv) An object can refer to itself by using **this** keyword. (v) A **overloaded** method is one that does different things depending upon the object that invokes it.
2	**Define inheritance. What is the inheritance mechanism in Java?**
Ans.	**Inheritance** is the capability of one class to inherit properties from another class. Object-oriented programming allows classes to inherit commonly used state and behavior from other classes. Inheritance is a compile-time mechanism in Java that allows you to extend a class (called the base class or superclass) with another class (called the derived class or subclass). In Java, inheritance is used for two purposes: 1. **class inheritance** - create a new class as an extension of another class, primarily for the purpose of code reuse. class derived-class-name extends base-class-name { .... } 2**. interface inheritance** - create a new class to implement the methods defined as part of an interface for the purpose of subtyping.public public class class-name implements interface-name { .... }
3	**Inheritance is a way to** **(i) Make general classes into more specific classes.** **(ii) Pass arguments to objects of classes.** **(iii) Add features to existing classes without rewriting them.** **(iv) Improve data-hiding and encapsulation.**
Ans.	Add features to existing classes without rewriting them.
4	**Advantages of inheritance include** **(i) Providing class growth** **(ii) Avoiding rewriting of code** **(iii) Data abstraction** **(iv) Simulation of real-world entities.**
Ans.	Avoiding rewriting of code Simulation of real-world entities.
5	**What is inheritance? Discuss its various forms.**
Ans.	**Inheritance** is the capability of one class to inherit properties from another class. (i) Single inheritance (ii) Multilevel inheritance (iii) Hierarchical.
6	**Define Base class and Derived class. How are these related?**
Ans.	The class from which the subclass is derived is called a super class (also a base class or a parent class). A class that is derived from another class is called a subclass (also a derived class, extended class, or child class). A derived class is formed from the base class. It can use its methods and variables (depend on specifier) and allow reusability. In a subclass, you can add new methods and new fields.
7	**True or False?**

	(i) Adding a derived class to a base class requires fundamental changes to the base class. **(ii) Derivation of classes facilitates transitivity.** **(iii) Use if inheritance saves on efforts and time.** **(iv) The way a derived class member function can access the protected and public members of base class, in the same way, the base class can also access protected and public members of derived class.** **(v) The size of a derived class object is equal to the sum of sizes of data numbers in base class and the derived class.**
Ans.	(i) False (ii) True (iii) True (iv) False (v) True

Type B: Short Answer Questions

1 **Discuss various reasons that support the concept of inheritance in Object Oriented Languages.**

Ans.

1. Inheritance is capable of expressing the inheritance relationship of real-world models. 'Men' inherit from 'Person'; 'Women' inherit from 'Person', etc.
2. Inheritance facilitates the code reusability. Additional features can be added to a class by deriving a class from it and then by adding new features to it. Class ones written and tested need not be rewritten or redefined.
3. Inheritance is capable of simulating the transitive nature of real-world's inheritance, which in turn saves on modification time and efforts, if required.

2 **How does the visibility mode control the access of members in the derived class? Explain with examples.**

Ans. Visibility mode controls the access of member in derived class by using access specifier. Below table shows different visibility modes.

Specifier	class	subclass	Package	World
private	Y			
protected	Y	Y	Y	
public	Y	Y	Y	Y
default	Y		Y	

```
class BaseClass {
     public int x = 10;
     private int y = 10;
     protected int z = 10;
     int a = 10; //Implicit Default Access Modifier
     public int getX() {
          return x;
     }
     private int getY() {
          return y;
     }
     private void setY(int y) {
          this.y = y;
     }
     protected int getZ() {
          return z;
     }
     int getA() {
          return a;
     }
}
```

	`public class SubclassInSamePackage extends BaseClass {`
	`  public static void main(String args[]) {`
	`  BaseClass rr = new BaseClass();`
	`  rr.z = 0;`
	`  SubclassInSamePackage subClassObj = new SubclassInSamePackage();`
	
	`  //Access Modifiers - Public`
	`  System.out.println("Value of x is : " + subClassObj.x);`
	
	`  //Access Modifiers – Private`
	`  // If comment is removed error will come as trying to access`
	`  //private member(s).`
	`  /* System.out.println("Value of y is : "+subClassObj.y);`
	`  subClassObj.setY(20);`
	`  System.out.println("Value of y is : "+subClassObj.y);*/`
	
	`  //Access Modifiers - Protected`
	`  System.out.println("Value of z is : " + subClassObj.z);`
	
	`  //Access Modifiers - Default`
	`  System.out.println("Value of x is : " + subClassObj.a);`
	`  }`
	`}`

3	**What is the difference between protected and private members?**
Ans.	**Protected members** are accessible inside their own class as well as in all subclasses of their class, regardless of whether subclasses exist in the same package or any other package whereas; **private members** are accessible only inside their own class and nowhere else.
4	**How are arguments sent to the base constructors in multiple inheritance? Whose responsibility is it?**
Ans.	Java doesn't support multiple inheritance, so it is not possible. If we use interface for then also it is not possible because interface doesn't have constructor. But if we use inheritance which is supported by Java then Super() is responsible for passing arguments to the base class constructor.
5	**If a base class as well as its derived class contain a member functions with the same name, say func(), can an object of the derived class access both these functions i.e., of base class and of derived class?**
Ans.	No, an object of derived class ca not access both these functions i.e., of base class and of derived class.
6	**If a base class contains a member function func(), and its derived class does not contain a function with this name, can an object of the derived class access func()?**
Ans.	No, an object of the derived class cannot access func().
7	**If a derived class doesn't add any data members to the base class, does the derived class require constructors?**
Ans.	Every class in Java has a default constructor in any condition/situation. If we need to instantiate object of class then constructor(s) is must, whether it is a base class or derived class
8	**If a base class and a derived class each include a member function with the same, which member function will be called by an object of the derived class, assuming the scope-resolution operator is not used? To call the other function, what can be done?**
Ans.	If a base class and a derived class each include a member function with the same, member function of derived class will be called by an object of the derived class. To call the base class function, we can use **super** keyword. If the member function name is func() then we can write the following statement: super.func()
9	**Assume a class Derv derived from a base class base. Both classes contain a member function func() that takes no arguments. Write the definition for a member function of Derv that calls both the func()s.**

Ans.	`class base`
	`{`
	`  void func()`

```
        {
                System.out.println("this is base class");
        }
}
class derv extends base
{
        void func()
        {
                System.out.println("this is derived class");
                super.func();
        }
}
class test
{
        public static void main(String[] args)
        {
                derv dr=new derv();
                dr.func();
        }
}
```

10	**Define an abstract class and abstract methods.**
Ans.	**Abstract classes** are classes that contain one or more abstract methods. **An abstract method** is a method that is declared, but contains no implementation.
11	**What is an interface?**
Ans.	**An interface** defines a protocol of behavior. It dictates common behavior among objects from diverse classes.
12	**We know that a private member of a base class is not directly accessible by a subclass. Is it anyway possible for the objects of a derived class to access the private members of the base class? If yes, how? Remember, the base class cannot be modified.**
Ans.	Since we know that private member of a base class cannot be inherited and it is not available for the derived class directly. This can be done by the modifying the visibility limit of the private member by making it public. This would make it accessible to all the other function of the program, thus taking away the data hiding.
13	**A class One with data members int a and char b inherits from another class Two with data members float f and int x. write definition for One and Two for the following situations:** **(i) Objects of both the classes are able to access all the data members of both the classes** **(ii) Only the members of class one can access all the data members of both the classes.**
Ans.	(i)

```
class Two
     {
        float f=0.0f;
        int x;
   }
   class One extends Two
   {
        int a=9;
        char b;
   }
```

(ii)

```
class Two
      {
        float f=0.0f;
        int x;
      }
      //final keyword restrict further inheritance.
       final class One extends Two
      {
        int a=9;
```

```
        char b;
      }
```

14 **Find out the errors in following code fragment:**

```
interface one{
      public and Method1();
}
public class test extends one{
     public void Method();
}
```

Ans.

```
interface one{
  public and Method1();
}
//implements should be used in place of extends
public class test extends one{
          public void Method();
}
```

15 **A class Cola has to inherit from a class SoftDrink and implement an interface Drinks. Write code to accomplish this. Make necessary assumption regarding the members of the classes and interface.**

Ans.

```
interface Drinks
{
      public void drink1();
      public void drink2();
      public void drink3();
}
 class SoftDrink implements Drinks
{
      String dr1="Cola";
      String dr2="sprite";
      String dr3="maza";
      public void drink1()
      {
            System.out.println("Drink1");
      }
      public void drink2()
      {
            System.out.println("Drink2");
      }
      public void drink3()
      {
            System.out.println("Drink3");
      }
}
class cola extends SoftDrink
{
      void drink()
      {
            System.out.println("cola class="+dr1);
      }
}
public class test
{
      public static void main(String[] args)
      {
            cola c=new cola();
            c.drink();
```

```
            SoftDrink sd=new SoftDrink();
            sd.drink1();
            sd.drink2();
            sd.drink3();
      }
}
```

16 What will happen when the following program is executed?

```
public class Superclass
{
   private int iamprivate=10;
   protected int iamprotected=20;
   public int iampublic=30;
   private void privateMethod()
   {
      System.out.println("I am private method in superclass");
   }
   protected void protectedMethod()
   {
      System.out.println("I am protected method in superclass");
   }
   public void publicMethod()
   {
      System.out.println("I am public method in superclass");
   }
}
public class subclass extends Superclass
{
   public void accessTest()
   {
      System.out.println(iamprivate);
      System.out.println(iamprotected);
      System.out.println(iampublic);
      privateMethod();
      protectedMethod();
      publicMethod();
   }
   public void Method()
   {
      System.out.println("I am public method in the class subclass");
   }
}
public class Test
{
      public static void main(String[] args)
      {
            subclass obj=new subclass();
            obj.accessTest();
            obj.Method();
            obj.publicMethod();
            obj.protectedMethod();
            obj.privateMethod();
      }
}
```

Ans. Above code will generate many compilation errors

i. Trying to define or declare a Superclass and Subclass as public class in a file which does not have the same name

	as the public class or interface (plus a .java extension). ii. Trying to access private member iamprivate and privateMethod().

Type C: Long/Practical Answer Questions

1	**Write a student class with following specifications:** **(i) Two private variables: first name, last name** **(ii) Constructor with two arguments.** **(iii) Void Method print Data() to print first + last name.**

Ans.	`public class Student`
	`{`
	`  private String fname;`
	`  private String lname;`
	`  Student(String fname, String lname)`
	`  {`
	`  this.fname=fname;`
	`  this.lname=lname;`
	`  }`
	`  void printData()`
	`  {`
	`  System.out.println(fname+lname);`
	`  }`
	`  public static void main(String[] args)`
	`  {`
	`  Student st=new Student("abc", "def");`
	`  st.printData();`
	`  }`
	`}`

2	**Write a Graduate class that inherits from Student class defined in question 24.** ▪ **Add Private variables, Stream, Degree** ▪ **It must have a constructor method.** ▪ **Method PrintGrade() to print Stream, Degree along with inherited first and last name.**

Ans.	`class student`
	`{`
	`  private String stream;`
	`  private String degree;`
	`  String fname,lname;`
	`  public student(String stream,String degree)`
	`  {`
	`  this.stream=stream;`
	`  this.degree=degree;`
	`  }`
	`  void printGrade()`
	`  {`
	`  System.out.println("Stream="+stream);`
	`  System.out.println("degree="+degree);`
	`  }`
	`}`
	`class Graduate extends student`
	`{`
	`  public Graduate(String stream, String degree,String fname,String lname)`
	`  {`
	`  super(stream,degree);`
	`  this.fname=fname;`
	`  this.lname=lname;`

http://cbsecsnip.in

```
      }
      void printGrade()
      {
            System.out.println("First name="+fname);
            System.out.println("Last name="+lname);
            super.printGrade();
      }
}
class stud
{
      public static void main(String[] args)
      {

            Graduate gd= new Graduate("Science","Bsc","priya","patel");
            gd.printGrade();
      }
}
```

3

Assume that a bank maintains two kinds of accounts for customers, one called as savings account and the other as current account. The savings account provides compound interest and withdrawal facilities but not cheque book facility. The current account provides cheque book facility but no interest. Current account holders should also maintain a minimum balance and if the balance falls below this level, a service charge is imposed.
Create a class Account that stores customers name, account number and opening balance.
From this derive the classes current and savings to make them more specific to their requirements.
Include necessary member functions in order to achieve the following tasks:

(i) Deposit an amount for a customer and update the balance.
(ii) Display the account details
(iii) Compute and deposit interest.
(iv) Withdraw amount for a customer after checking the balance and update the balance.
(v) Check for the minimum balance (for current account holders), impose penalty, if necessary, and update the balance.

Implement these without using any constructor.

Ans.

```
import java.util.Scanner;     //Used for input from key broad
import java.text.DecimalFormat;   //Used for decimal place formatting
class Account{
   private String Cust_Name;
   private int Acc_No;
   private double Open_Bal;
   Scanner input = new Scanner(System.in);
   public double get_Data(){
        System.out.print("Enter the Customer Name : ");
        Cust_Name = input.next();
        System.out.print("Enter the Account No. : ");
        Acc_No = input.nextInt();
        System.out.print("Enter the Opening Balance. : ");
        Open_Bal = input.nextDouble();
        return Open_Bal;
   }
}

class Saving extends Account{
  double calamt,obal,amount;
  public double operation(){
    //Inherited from Account class for storing detail of customer
    obal=get_Data();
    Scanner input = new Scanner(System.in);
```

```
    System.out.println("Enter D for Deposit or W for Withdraw->");
    String ch=input.next();
    switch(ch){
    case "D":
    case "d":
       System.out.println("Enter Deposit Amount ");
       amount=input.nextDouble();
       obal=obal+amount;
       calamt=obal*Math.pow(1+8.5/400,4);  //Qtr Compound Interest Calculation
       break;
    case "W":
    case "w":
       System.out.println("Enter Withdrwal Amount ");
       amount=input.nextDouble();
       if(amount >=obal)
       {
          System.out.println("Insufficient Balance");
       }
       else
       {
          obal=obal-amount;
          calamt=obal*Math.pow(1+8.5/400,4);
       }
       break;
    default: System.out.println("Wrong Choice");
       break;
    }
    return calamt;
  }
}
class Current extends Account{
  public double operation(){
    //Inherited from Account class for storing detail of customer
    obal=get_Data();
    Scanner input = new Scanner(System.in);
    System.out.println("Enter D for Deposit or W for Withdraw->");
    String ch=input.next();
    switch(ch) {
     case "D":
     case "d":
       System.out.println("Enter Deposit Amount ");
       amount=input.nextDouble();
       obal=obal+amount;
       calamt=obal*Math.pow(1+8.5/400,4);  //Qtr Compound Interest Calculation
       break;
     case "W":
     case "w":
       System.out.println("Enter Withdrwal Amount ");
       amount=input.nextDouble();
       if(amount >=obal || (obal-amount)<500) //minimum balance assumed Rs.500
       {
          System.out.println("Insufficient Balance");
       }
       else
       {
           obal=obal-amount;
```

```
            calamt=obal*Math.pow(1+8.5/400,4);
        }
        break;
     default: System.out.println("Wrong Choice");
              break;
    }
    return calamt;
  }
}
public class Bank{
    public static void main(String[] args){
         double final_amt=0.0;
         DecimalFormat df=new DecimalFormat("#######.##");
         Scanner input = new Scanner(System.in);
         System.out.println("Enter S for Saving or C for Current ->");
         String ch=input.next();
         switch(ch)
         {
              case "S":
              case "s":
                   Saving saving = new Saving();
                   final_amt = saving.operation();
                   System.out.println(df.format(final_amt));
                   break;
              case "C":
              case "c":
                   Current current = new Current();
                   final_amt = current.operation();
                   System.out.println(df.format(final_amt));
                   break;
              default: System.out.println("Wrong Selection");
                   break;
         }
    }
}
```

//code is for 1 customer for multiple customers some other logic/technique have to use.

4	**Write a java program to read and display information about employees and managers. Employee is a class that contains employee number, name, address and department. Manager class contains all information of the employee class and a list of employees working under a manager.**
Ans.	Now try this code take idea from above solution
5	**A financial Institution has only two kinds of accounts, Simple and Compound.** **The class Account has two protected attributes accountNumber (a positive integer) and principal (a double precision real number to store the money deposited initially). The interest for a Simple account is calculated by using the simple interest formula:** **SI=PRT/100** **Where SI – Simple Interest, P – Principal, R – Rate of interest per annum,T – Time in years. The interest for a compound account is calculated by using the compound interest formula:** **CI = $P[1+R/100]^T$ –P** **Where CI –Compound Interest, P – Principal, R – Rate of interest per annum,T – Time in years Rate and Time for two kinds of accounts may vary.** **Model the above situation by defining classes Account, Simple and Compound where Simple and Compound are derived classes from class Account.** **Write appropriate constructors for passing various attributes for the three classes.** **Write the functions double interest () that calculate the appropriate interest and return the interest for the**

classes Simple and Compound.
Also write the functions display () for the thre classes which print out all the attribute values as
account number = …….
principal = …….
rate = …….
time = …….
interest = …….
with each attribute on a single line. Make sure that each display () function is correctly placed in the appropriate class. You do not need to write the main function.

Ans.

```
class Account
{
      protected int accno;
      protected double principal;
      int rate;
      int time;
      public double intt;
      public Account(int accno,double principal)
      {
            this.accno=accno;
            this.principal=principal;
      }
}
class Simple extends Account
{
      public Simple(int accno,double principal,int rate,int time)
      {
            super(accno,principal);
            this.rate=rate;
            this.time=time;
      }
      double interest()
      {
            intt = principal*rate*time/100;
            return intt;
      }
      void display()
      {
            System.out.println("Account number="+accno);
            System.out.println("Principal="+principal);
            System.out.println("Rate="+rate);
            System.out.println("Time="+time);
            System.out.println("Interest="+intt);
      }
}
class Compound extends Account
{
      public Compound(int accno,double principal,int rate,int time)
      {
            super(accno,principal);
            this.rate=rate;
            this.time=time;
      }
      double interest()
      {
             intt = principal * Math.pow((1 + rate/100),time);
```


http://cbsecsnip.in

```
            return intt;
      }
      void display()
      {
            System.out.println("Account number="+accno);
            System.out.println("Principal="+principal);
            System.out.println("Rate="+rate);
            System.out.println("Time="+time);
            System.out.println("Interest="+intt);
      }
}
public class calculate
{
            public static void main(String[] args)
             {

      Simple si=new Simple(1234,15.00,5000,2);
      double sim=si.interest();
      System.out.println("Simple interest");
      si.display();
      Compound co=new Compound(1234,15.00,5000,2);
      double cmp=co.interest();
      System.out.println("Compound interset");
      co.display();
            }
}
```

6

A class D2Point defines the coordinates of a point in a plane while another class D3Point defines the coordinates of a point in space. The details of both the classes are given below:

Class name:	**D2Point**
Data members	**double x, y To store the x and y coordinates**
Functions/methods:	
D2Point ()	**Constructor to assign 0 to x**
D2Point (double nx, double ny)	**Constructor to assign nx to x and ny to y**
double distance 2d (D2Point b)	**to return the distance between the point b and the current point in a plane.**
Class name	**D3Point**
Data members	**double z To store the z coordinate.**
Functions/methods:	
D3Point ()	**constructor to assign 0 to z**
D3Point (double nz)	**constructor to assign nz to z**
double distance3d (D3Point b)	**to return the distance between the point b and the current point in space.**

If x1, y1, z1, x2, y2, z2 be the coordinates of the two points, then the distance between he points in a plane is given by the formula:

and the distance between the points in space is given by the formula:

Specify the class D2Point giving the details of the two constructors and functions double distance2d (D2Point b). Using the concept of inheritance specify the class D3Point giving the details of the two constructors and function double distance3d (D3Point b). you do not need to write the main function.

Ans.

```
public class D2Point
{
      protected double x;
      protected double y;
      public D2Point()
      {
```

```
            x=y=0;
      }
      public D2Point(double nx, double ny)
      {
            x = nx;
            y = ny;
      }
      double distance2d(D2Point b)
      {
            double d=Math.sqrt((b.x-x)*(b.x-x)+(b.y-y)*(b.y-y));
            return d;
      }
}
public class D3Point extends D2Point
{
      private double z;
      public D3Point()
      {
            super();
            z=(0.0);

      }
      public D3Point(double nx, double ny,double nz)
      {
            super(nx,ny);
            z=nz;
      }
      double distance3d(D3Point b)
      {
            double d = Math.sqrt((b.x-x)*(b.x-x)+(b.y-y)*(b.y-y)+(b.z-z)*(b.z-
z));
            return d;
      }
}
```

7	**Function not returning any value has return type as:** **(a) int** **(b) char** **(c) float** **(d) void.**
Ans.	void
8	**Which of the following function-definitions are overloading the method given below:** **int sum(int x, int y) { }** **(a) int sum(int x, int y, int z) { }** **(b) float sum (int x, int y) { }** **(c) int sum (float x, float y) { }** **(d) int sum (int a, int b) { }** **(e) float sum (int x, int y, float z) { }**
Ans.	int sum(int x, int y, int z) { } int sum (int a, int b)
9	**What is function overloading? What is the significance of function overloading in java?**
Ans.	A function name having several definitions that are differentiable by the number of types of their arguments is known as function overloading. Following are significance of function overloading: 1. Overloaded methods are bonded using static binding in Java. Which occurs during compile time i.e. when you

	compile Java program. During compilation process, compiler bind method call to actual method. 2. Overloaded methods are fast because they are bonded during compile time and no check or binding is required during runtime.
10	**What is the role of a function's signature in disambiguation process?**
Ans.	A method signature is part of the method declaration. It is the combination of the method name and the parameter list. The Java supports overloading methods. Overloaded methods are differentiated by the number and the type of the arguments passed into the method. Java can distinguish between methods with different method signatures. This means that methods within a class can have the same name if they have different parameter lists.