Week 7 Java Assignment

1.To develop a vehicle class hierarchy in Java to demonstrate the concept of polymorphism.
Step 1:-Declare a super class called vehicle with data elements doors,wheels and seats.

Step 2:-Derive another class called car and invoke a function tostring() to display the variables.

Step 3:-Derive another class calledmotorcycle with same data and method called setseats() .

Step 4:-Declare another sub class called Truck with 2 constructors and finally assign values to variables.

Step 5:-In the main function, create an object for class motorcycle and display all details of sub classes through object.

Sourcecode:-

```
//This is the class that will be
inheritedpublic class Vehicle
{
publicint doors;
publicint seats;
publicint wheels;
Vehicle()
{
wheels=4;
doors=4;
seats=4;
}
}
//This class inherits
Vehicle.javapublic class Car
extends Vehicle
{
public String toString()
{
return "This car has "+seats+" Seats, "+doors+" Doors
"+"and "+wheels+" wheels.";
}
```

```
}
//This class inherits Vehicle.java
public class MotorCycle extends Vehicle
{
MotorCycle()
{
wheels=2;
doors=0;
seats=1;
}
voidsetSeats(intnum)
{
seats=num;
}
public String toString()
{
return "This motorcycle has "+seats+" Seats, "+doors+"
Doors "+"and "+wheels+" wheels.";
}
}
//This class inherits Vehicle.java
public class Truck extends
Vehicle
{
booleanisPickup;
Truck()
{
isPickup=true;
}
Truck(booleanaPickup)
{
this();
isPickup=aPickup;
}
Truck(int doors, int seats, intinWheels, booleanisPickup)
{
this.doors=doors;
this.seats=seats;
wheels=inWheels;
this.isPickup=isPickup;
}
public String toString()
{
```

```
return "This "+(isPickup?"pickup":"truck")+
" has "+seats+" Seats, "+doors+" Doors "+"and
"+wheels+"wheels.";
}
}
//This class tests the classes that inherit Vehicle.java
```

```
public class VehiclesTest
{
public static void main(String args[])
{
MotorCycle mine = new
MotorCycle();
System.out.println(mine);
Car mine2 = new Car();
System.out.println(mine2);
mine2.doors=2;
System.out.println(mine2);
Truck mine3 = new Truck();
System.out.println(mine3);
Truck mine4 = new
Truck(false);mine4.doors=2;
System.out.println(mine4);
}
}
```

Output:-

```
This motorcycle has 1 Seats, 0 Doors and 2
wheelsThis car has 4 Seats, 4 Doors and 4 wheels
This car has 4 Seats, 2 Doors and 4 wheels
This pickup has 4 Seats, 4 Doors and 4
wheelsThis truck has 4 Seats, 2 Doors and 4
wheels
```

2.What is super class and subclass?

Inheritance (Subclass and Superclass)

In Java, it is possible to inherit attributes and methods from one class to another. We group the "inheritance concept" into two categories:

- **subclass** (child) - the class that inherits from another class
- **superclass** (parent) - the class being inherited from

To inherit from a class, use the extends keyword.

In the example below, the Car class (subclass) inherits the attributes and methods from the `Vehicle` class (superclass)

Example:

```
class Vehicle {

  protected String brand = "Ford";

  public void honk() {

    System.out.println("Tuut, tuut!");

  }

}



class Car extends Vehicle {

  private String modelName = "Mustang";

  public static void main(String[] args) {

    Car myFastCar = new Car();

    myFastCar.honk();

    System.out.println(myFastCar.brand + " " + myFastCar.modelName);

  }

}
```

OUTPUT:

```
Tuut,tuut !

Ford mustang
```

3.Can you overload a final method in Java?

```
class SumTest {

    public final void sum(int num1, int num2) {
        System.out.println(num1 + num2);
    }
    public void sum(int num1, int num2, int num3) {
        System.out.println(num1 + num2 + num3);
    }
 }

public class Main
{
      public static void main(String[] args) {
            SumTest sumTest = new SumTest();
            sumTest.sum(10, 5);
            sumTest.sum(10, 5, 20);
      }
}
```

Output

```
15
35
```

<u>Final method can not be overridden. See the below example.</u>

```
class SumTest {

    public final void sum(int num1, int num2) {
        System.out.println(num1 + num2);
    }

    public final void sum(int num1, int num2, int num3) {
        System.out.println(num1 + num2 + num3);
    }
 }

public class Main extends SumTest
{
    public void sum(int num1, int num2) {
        System.out.println(num1 + num2);
    }
      public static void main(String[] args) {
            SumTest sumTest = new SumTest();
            sumTest.sum(10, 5);
            sumTest.sum(10, 5, 20);
            Main main = new Main();
            main.show();
      }
}
```

Output

```
Main.java:14: error: sum(int,int) in Main cannot override sum(int
SumTest
    public void sum(int num1, int num2) {
                ^
  overridden method is final
Main.java:22: error: cannot find symbol
           main.show();
               ^
  symbol:   method show()
  location: variable main of type Main
2 errors
```

Note: A non final method can not be overloaded as well as overridden with final declaration.

```
class SumTest {

    public void sum(int num1, int num2) {
        System.out.println(num1 + num2);
    }

    public final void sum(int num1, int num2, int num3) {
        System.out.println(num1 + num2 + num3);
    }
 }

public class Main extends SumTest
{
    public final void sum(int num1, int num2) {
        System.out.println(num1 + num2);
    }
      public static void main(String[] args) {
           SumTest sumTest = new SumTest();
           sumTest.sum(10, 5);
           sumTest.sum(10, 5, 20);
           Main main = new Main();
           main.sum(10, 5);
      }
}
```

Can an Abstract class be Final in Java

There is a proverb in Hindi, "Haath Kangan ko Aarshi Kya, Padhe Likhe ko Parsi kya", which means instead of saying just show. So, why not we instead of saying that whether we can make an abstract final in Java or not, just code it and see what happens.

}

Let's try that, the following is a sample code that uses both abstract and final modifier together at class declarations when you write this code in Eclipse, the IDE will suggest this is wrong by giving an error as shown below:

```
abstract final class ImAnAbstractClass{

}
```

Error: "The class ImAnAbstractClass can be either abstract or final, not both"

Unlike many others, this error is both clear and concise and shows that it's not possible to make a class both final and abstract at the same time in Java.

Btw, If you compile the above Java source file using Java compiler (javac) from the command line you will receive a similar error. Eclipse uses a slightly different compile but it follows almost all rules of the Java compiler.

Can we make an Abstract method final in Java

Now that, you know there is no way you can make an abstract class final in Java, let's try to make an abstract method final in Java. As I said above, this time also Java compiler should complain because both final and abstract are mutual exclusive keywords:

```
abstract class ImAbstract{

public final abstract void anAbstractMethod();
}

class ImConcrete extends ImAbstract{
```

```
@Override

public void anAbstractMethod() {

// TODO Auto-generated method stub



}



}
```

This will give the error "The abstract method anAbstractMethod in type ImAbstract can only set a visibility modifier, one of public or protected" when you write this code in Eclipse, as shown in the following screenshot:

Can an abstract class be final in Java

We will also receive the error "Cannot override the final method from an abstract" on the subclass, as shown in the following screenshot which shows cannot override the final method from an abstract class.

Is it possible to make an abstract method final in Java

5.Write the syntax for creating the subclass of a class?

Creating Subclasses

declare that a class is the subclass of another class within The Class Declaration. For example, suppose that you wanted to create a subclass named SubClass of another class named SuperClass. You would write:

```
class SubClass extends SuperClass {
    . . .
}
```

This declares that SubClass is the subclass of the Superclass class. It also implicitly declares that SuperClass is the superclass of SubClass.

A subclass also inherits variables and methods from its superclass's superclass, and so on up the inheritance tree.

For purposes of making our discussion easier, when this tutorial refers to a class's superclass it means the class's direct ancestor as well as all of its ascendant classes.

A Java class can have only one direct superclass. Java does not support multiple inheritance.

Creating a subclass can be as simple as including the extends clause in your class declaration (such as in the declaration in ImaginaryNumber above).

However, you usually have to make other provisions in your code when subclassing a class, such as overriding methods.

class can access a hidden member variable through its superclass. Consider this superclass and subclass pair:

```
class Super {
   Number aNumber;
}
class Sub extends Super {
   Float aNumber;
}
```

The aNumber variable in Sub hides aNumber in Super. But you can access aNumber from the superclass with:
super.aNumber

super is a Java language keyword that allows a method to refer to hidden variables and overriden methods of the superclass

6.Write a program to get the personal details of students such as name, register number, age and department from the base class Student and display those information in a class named Student Display.

```
class Personal

{

  String name=new String();

  String fname=new String();

  String add=new String();

  String phno=new String();
  Personal(String a,String b,String c,String d)
  {
     name=a;
     fname=b;
    add=c;
    phno=d;
   }

  void display()
  {
   System.out.println("Name is "+name);
    System.out.println("Address  "+add);
    System.out.println("Father's Name is "+fname);
   System.out.println("Contact number "+phno);
 }
}
class Education extends Personal
{
  int roll,age;
  char section;
  String branch=new String();
  Education(String a, String b, String c, String d, int e, int f, char  g, String h)
  {
   super(a,b,c,d);
    roll=e;
   age=f;
  section=g;
   branch=h;
  }
void display2()
```

```
 {
   super.display();
   System.out.println("Roll Number="+roll);
   System.out.println("AGE="+age);
   System.out.println("SECTION="+section);
   System.out.println("Branch is "+branch);

 }

}

class Hello
{
  public static void main(String args[])
  {
    Education e=new
Education("KRISH","RAM","VIZAG","9885098850",10,19,'B',"IT");
    e.display2();
   }
}
```

OUTPUT

```
D:\>javac Hello.java

D:\>java Hello
Name is KRISH
Address  VIZAG
Father's Name is RAM
Contact number 9885098850
Roll Number=10
```

AGE=19

SECTION=B

Branch is IT