JBK1006-Assignment: Inheritance

Program for single inheritance

```
Class A{
  public void methodA()  {
   System.out.println("Base class method");
  } }

Class B extends A{
  public void methodB() {
   System.out.println("Child class method");
  }
  public static void main(String args[]) {
   B obj = new B();
   obj.methodA(); //calling super class  method
   obj.methodB(); //calling local method
 } }
```

Program for single inheritance

```
public class Vehicle {
      String color;  int speed;   int size;
          void display() {
          System.out.println("Color : " + color);
          System.out.println("Speed : " + speed);
          System.out.println("Size : " + size);
        }   }
public class Car extends Vehicle{
        int CC;
        int gears;
        void displayCar() {
          System.out.println("Color of Car : " + color);
          System.out.println("Speed of Car : " + speed);
          System.out.println("Size of Car : " + size);
          System.out.println("CC of Car : " + CC);
         System.out.println("No of gears of Car : " + gears);
```

```
        } }
public class Test {
public static void main(String[] args) {
            Car b1 = new Car();
          b1.color = "Blue";
          b1.speed = 200 ;
          b1.size = 22;
           b1.display();
          b1.CC = 1000;
          b1.gears = 5;
          b1.displayCar();
      } }
```

Another example of Single Inheritance

```
class Vehicle{
   String vehicleType;
}
public class Car extends Vehicle {

   String modelType;
   public void showDetail() {
      vehicleType = "Car";      //accessing Vehicle class
member
      modelType = "sports";
      System.out.println(modelType+" "+vehicleType);
   }
   public static void main(String[] args)  {
      Car car =new Car();
      car.showDetail();
   } }
```

Program for single inheritance with getter setter

```
public class Vehicle {
```

```
        String color;
          private int speed;
          private int size;
          public int getSize() {
            return size;
} public int getSpeed() {
            return speed;
}  public void setSize(int i) {
            size = i;
}   public void setSpeed(int i) {
            speed = i;
        }   }
public class Car extends Vehicle{
          int CC;
           int gears;
          int color;
}  public class Test {
public static void main(String[] args) {
              Car b1 = new Car();
                 b1.color = 500;
              b1.setSpeed(200);
              b1.setSize(22);
              b1.CC = 1000;
              b1.gears = 5;
             System.out.println("Color of Car : " + b1.color);
System.out.println("Speed of Car : " + b1.getSpeed());
System.out.println("Size of Car : " + b1.getSize());
System.out.println("CC of Car : "  + b1.CC);
System.out.println("No of gears of Car : " + b1.gears);
        }          }
```

Program for multilevel inheritance

```
Class X{
  public void methodX() {
```

```
    System.out.println("Class X method");
  }}
Class Y extends X{
public void methodY(){
System.out.println("class Y method");
}}
Class Z extends Y{
  public void methodZ()  {
    System.out.println("class Z method");
  }
  public static void main(String args[]) {
    Z obj = new Z();
    obj.methodX(); //calling grand parent class method
    obj.methodY(); //calling parent class method
    obj.methodZ(); //calling local method
 }}
```

Program for multilevel inheritance

Program 1:

```
public class Car {
      public Car(){
            System.out.println("Constructor of class
Car");
      }
public void vehicleType(){
            System.out.println("Vehicle Type : Car");
      }   }
public class Maruti extends Car {
      public Maruti(){
            System.out.println("Constructor of class
Maruti");
      }
public void brand(){
```

```
            System.out.println("Brand : Maruti");
      }
public void speed(){
            System.out.println("Max speed: 90Kmph");
      }    }

public class Maruti800 extends Maruti {
      public Maruti800(){
            System.out.println("Constructor of class
Maruti800");
      }
public void speed(){
                  System.out.println("Max speed:
80Kmph");
            }
public static void main(String[] args) {
            Maruti800 obj=new Maruti800();
            obj.vehicleType();
            obj.brand();
            obj.speed();
            }  }
```

Program 2:

```
class User {
String name;    int age;    long ph;
 }
class Employee extends User{
      String specialization;
}
class Manager extends User{
      String department;
}

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

```
Employee e1=new Employee();
        e1.name="Candid";
        e1.age=22;
        e1.ph=123456789l;
        e1.specialization="Java";
Manager m1=new Manager();
        m1.name="java";
        m1.age=25;
        m1.ph=345789l;
        m1.department="HR";

        System.out.println(e1.name);
        System.out.println(e1.age);
        System.out.println(e1.ph);
        System.out.println(e1.specialization);

        System.out.println(m1.name);
        System.out.println(m1.age);
        System.out.println(m1.ph);
        System.out.println(m1.department);
        } }
```

Program for hierarchical inheritance

```
class A{
  public void methodA() {
    System.out.println("method of Class A");
  }}
class B extends A{
  public void methodB()  {
    System.out.println("method of Class B");
  }}
class C extends A{
 public void methodC() {
   System.out.println("method of Class C");
 }}
```

```
class D extends A{
  public void methodD() {
    System.out.println("method of Class D");
  }}
class JavaExample{
  public static void main(String args[]){
    B obj1 = new B();
    C obj2 = new C();
    D obj3 = new D();
    //All classes can access the method of class A
    obj1.methodA();
    obj2.methodA();
    obj3.methodA();
  } }
```

Program for hierarchical inheritance

```
public class A {
        public void methodA()  {
            System.out.println("method of Class A");
         } }
public class B extends A{
        public void methodB() {
            System.out.println("method of Class B");
         }    }
public class C extends A {
        public void methodC() {
         System.out.println("method of Class C");
        }     }
public class D extends A {
        public void methodD() {
            System.out.println("method of Class D");
         }  }
public class Test {
                public void methodB() {
            System.out.println("method of Class B");
```

```
        }
 public static void main(String[] args) {
          B obj1 = new B();
          C obj2 = new C();
          D obj3 = new D();
          obj1.methodA();
          obj2.methodA();
          obj3.methodA();
}        }
```

Program that illustrates protected variable use in java using shape class

```
public class Shape{
  protected int sides;
public Shape() {
   sides = 3;
 }
public int getSides() {
   return sides;
  }
public printSides()  {
    System.out.println("This object has " + sides + "
sides." );
  }  }
public class Square extends Shape{
  public Square(int nSides)  {
   sides = nSides;  // dont need to call super class
constructor due to protected type of variable.
  } }
class ProtectedVariableDemo{
  public static void main(String args[]) {
     Square sObj = new Square(10);
     sObj.printSides();
  }  }
```