Java Classes and Objects – 2

1. Create a class called Date that includes three pieces of information as instance variables—a month (typeint), a day (typeint) and a year (typeint). Your class should have a constructor that initializes the three instance variables and assumes that the values provided are correct. Provide a set and a get method for each instance variable. Provide a method displayDate that displays the month, day and year separated by forward slashes(/).Write a test application named DateTest that demonstrates classDate's capabilities.

2. Create a class called Invoice that a hardware store might use to represent an invoice for an item sold at the store.An Invoice should include four pieces of information as instance variables-a part number(type String),a part description(type String),a quantity of the item being purchased (type int) and a price per item (double). Your class should have a constructor that initializes the four instance variables. Provide a set and a get method for each instance variable. In addition, provide a method named getInvoice Amount that calculates the invoice amount (i.e., multiplies the quantity by the price per item), then returns the amount as a double value. If the quantity is not positive, it should be set to 0. If the price per item is not positive, it should be set to 0.0. Write a test application named InvoiceTest that demonstrates class Invoice's capabilities.

3. Write a method named swapPoints that accepts two Points as parameters and swaps their x/y values. Consider the following example code that calls swapPoints.

```
Point p1 = new Point(5, 2);
Point p2 = new Point(-3, 6);
Point.swapPoints(p1, p2);
System.out.println("(" + p1.x + ", " + p1.y + ")");
System.out.println("(" + p2.x + ", " + p2.y + ")");
```

The output produced from the above code should be:

```
(-3, 6)
(5, 2)
```

4. Suppose that the following two classes have been declared:

```
public class Car {
    public void m1() {
        System.out.println("car 1");
    }

    public void m2() {
        System.out.println("car 2");
    }
```

```
    public String toString() {
        return "vroom";
    }
}

public class Truck extends Car {
    public void m1() {
        System.out.println("truck 1");
    }

    public void m2() {
        super.m1();
    }

    public String toString() {
        return super.toString() + super.toString();
    }
}
```

Write a class `MonsterTruck` whose methods have the behavior below. Don't just print/return the output; whenever possible, use inheritance to reuse behavior from the superclass.

Method	Output/Return
m1	monster 1
m2	truck 1 car 1
toString()	"monster vroomvroom"