King Fahd University of Petroleum & Minerals

College of Computer Science and Engineering

Information and Computer Science Department

First Semester 101 (2010/2011)

ICS 201 – Introduction to Computing II

Major Exam 1

Thursday, 28th October, 2010

Time: 120 minutes

Name:

ID#:

Please circle your section number below:

Section	01	02	03	04
Instructor	Sami	Sukairi	Sami	Sukairi
Day and Time	SMW 8 - 8:50	SMW 9 -9:50	SMW 10 - 10:50	SMW 13:10 – 14:00

Question #	Maximum Mark	Obtained Mark
1	15	
2	15	
3	30	
4	25	
5	15	
Total	**100**	

Question 1 [15 Points (5 + 10)]

a) What is the right answer?

1) What does a derived class automatically inherit from the base class?
   (a) instance variables
   (b) static variables
   (c) public methods
   (d) all of the above

2) If the final modifier is added to the definition of a method, this means:
   (a) The method may be redefined in the derived class.
   (b) The method may be redefined in the sub class.
   (c) The method may not be redefined in the derived class.
   (d) None of the above.

3) Java does not use late binding for methods marked as:
   (a) final
   (b) static
   (c) private
   (d) all of the above

4) A class that implements an interface but only gives definitions for some of the method headings given in the interface is called a/an:
   (a) concrete class
   (b) abstract class
   (c) discrete class
   (d) friendly class

5) If a method throws an exception, and the exception is not caught inside the method, then the method invocation:
   (a) terminates
   (b) transfers control to the catch block
   (c) transfers control to the exception handler

b) Given the following Shoe, TennisShoe, and WhiteTennisShoe classes:

```
class Shoe {
        public Shoe() {
                this("This is a shoe");
                System.out.println("Base Class");
        }
        public Shoe(String s)      {
                System.out.println(s);
}
}

class TennisShoe extends Shoe {
        public TennisShoe(){
                this("This is a Tennis Shoe");
                System.out.println("Derived Class");
        }
        public TennisShoe(String s) {
                super("Exam 1");
                System.out.println(s);
}
}

class WhiteTennisShoe extends TennisShoe {
        public WhiteTennisShoe(String s) {
                System.out.println(s);
}
}
```

What is the output of the following Test class?

```
class Test {
        public static void main(String args[]) {
                new WhiteTennisShoe ("A white tennis shoe is created");
        }
}
```

Question 2 [15 Points]

Consider the following **Sale** class:

```
public class Sale
{
   private String name; //nonempty string
   private double price; //nonnegative

   public Sale() {};
   public Sale(String theName, double thePrice)   {
      setName(theName);
      setPrice(thePrice);
   }

   public static void announcement( )    {
      System.out.println("This is the Sale class.");
   }

   public double getPrice( )    {
      return price;
   }

   public void setPrice(double newPrice)    {
      if (newPrice >= 0)
         price = newPrice;
      else
      {
         System.out.println("Error: Negative price.");
         System.exit(0);
      }
   }

   public String getName( )    {
      return name;
   }

   public void setName(String newName)    {
      if (newName != null && newName != "")
         name = newName;
      else {
         System.out.println("Error: Improper name value.");
         System.exit(0);
      }
   }

   public String toString( )    {
      return (name + " price = SAR " + price);
   }

 public double bill( )    {
      return price;
   }
}
```

We want to provide an implementation of a class **DiscountSale** which represents a sale having a discount that is represented as a percentage.

1. Provide an implementation of a constructor for the **DiscountSale** class that initializes the fields of that class.

2. Using the overriding concept, provide a method **bill()** that returns a DiscountSale bill, **toString()** that displays: the name, the discounted price and discount of a **DiscountSale** Object, and a method **announcement()** that returns the string "This is a DiscountSale class".

3. Assume that the default constructor of **DiscountSale** is defined. What is the output of the following code?

```
 Sale sl = new DiscountSale("map",5,0);
  DiscountSale ds = new DiscountSale();
  if (sl instanceof DiscountSale) {
      ds = (DiscountSale)sl;
      System.out.println("ds was changed to " + sl);
}
```

4. Suppose you add the following method to the class **Sale**:

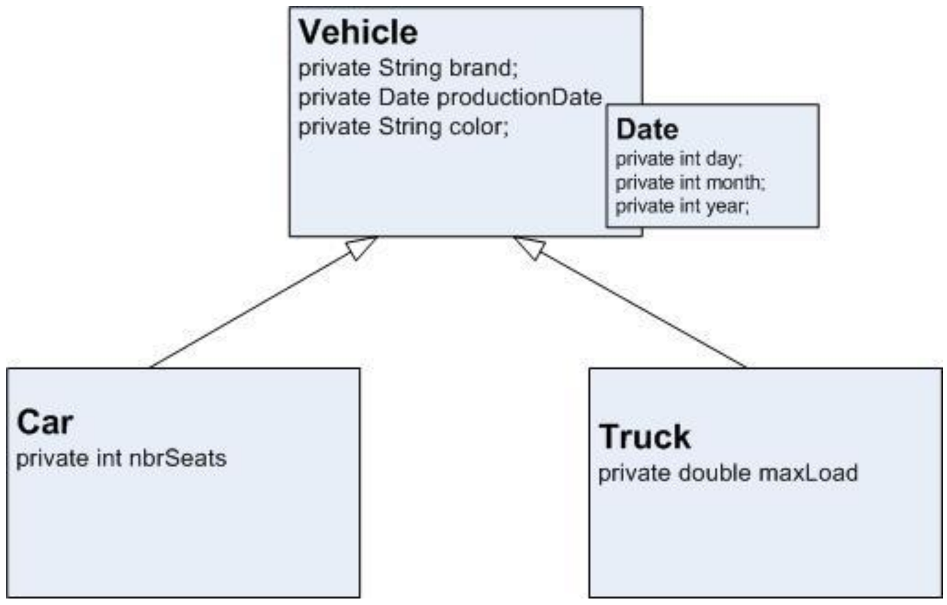
```
public void showAdvertisement() {
      announcement();
      System.out.println(toString());
}
```

Assume further that the method **showAdvertisement()** is not overridden in the class **DiscountSale**. What is the output of the following code?

```
Sale s = new Sale("floor mat",10.00);
DiscountSale discount = new DiscountSale("floor mat",11.00,10);
s.showAdvertisement();
discount.showAdvertisement();
```

Question 3 [25 Points]

The figure illustrates a Vehicle inheritance hierarchy consisting of 3 classes and 1 inner class. This illustration does not show all methods of the classes. It shows only the members that you need in this exercise. Don't worry about the remaining members.



Car and Truck classes are derived from Vehicle class. Date is a <u>public</u> inner class in Vehicle.

Part I

a) Write a constructor for Vehicle that takes three parameters: a String for brand, a Date for productionDate and a String for color:

b) Write a no-argument constructor for Vehicle that initializes the brand to "No brand", the Date to January 1st, 2000 and the color to "No color".
<u>Important</u>: the no-argument constructor should call the constructor you wrote in (a).
<u>Note</u>: assume that Date class has a constructor that takes three integer parameters.

c) Write a constructor for class Car that takes four parameters: String, Date, String and int :

d) Assuming that class Vehicle has accessor methods: getBrand(), getProdDate(), and getColor(), overwrite method equals in class Truck such that two Truck objects are equal means that they have the same brand, productionDate, color and maxLoad :

Part II

a) Compiling the source file Vehicle.java will generate how many files? List them:

b) Is the inner class Date inherited in Car?

c) Suppose you have a test class: TestClass with only a main method. Write a code to create an object of class Date initialized to January 1st, 2000:

```
Public class TestClass
{
  Public static void main(String [] args)
  {


   }
}
```