King Fahd University of Petroleum & Minerals

College of Computer Science and Engineering

Information and Computer Science Department

First Semester 091 (2009/2010)

ICS 201 – Introduction to Computing II

Major Exam 1

Wednesday, 4th November, 2009

Time: 120 minutes

Name: _____

ID#: _____

Please circle your section number below:

Section	03	05	04	06
Instructor	Tarek	Sami	Sukairi	Sukairi
Day and Time	SMW 9 - 9:50	SMW 8 -8:50	SMW 8 - 8:50	SMW 13:10 – 14:00

Question #	Maximum Mark	Obtained Mark
1	25	
2	25	
3	30	
4	20	
Total	**100**	

Question 1 [25 marks]

Consider the following class: (This class represents an object for a rented car)

```
class RentACar {
    private String name;
    private double baseCharge, chargePerKm, totalCharge;

    public RentACar(String name, double baseCharge, double chargePerKm)
    {
           this.name = name;
           this.baseCharge = baseCharge;
           this.chargePerKm = chargePerKm;
           totalCharge = baseCharge;
    }

    public void calculateCharge(double distance) {
           totalCharge += chargePerKm * distance;
    }

    public String toString() {
           return name + " Total Charge: " + totalCharge;
    }
}
```

(a) [19 marks] Write a subclass of the above class called "**MyCar**" with the following features:

- Include an instance variable **isCommercial** that indicates whether the car is being used for commercial purposes (e.g. as a taxi) or not.
- Include appropriate constructor(s).
- Override the **calculateCharge** method to calculate the charges based on:
    - If the vehicle is being used for commercial purposes, the calculated charges (as calculated by the **calculateCharge()** method) increase by 50%
    - If the vehicle is *not* being used for commercial purposes, there is a discount of 10% in the calculated charges.
- Override the **toString()** method to indicate the details of the vehicle followed by its commercial use status.

(b) [6 marks] Write a main method to test your class **MyCar**. Calculate the **totalCharge** for each of your vehicles for a distance of 100 km. Your output should look as follows:

```
Toyota Total Charge: 100.0 Commercial Vehicle: false
Suzuki Total Charge: 80.0 Commercial Vehicle: true
```

[blank page]

Question 2 [25 marks]

(a) [6 marks] The following java program will output

```
public class A extends Object {
      public void m( ){
            System.out.println("m in class A");
      }
}

public class B extends A {
      public void m( ){
            System.out.println("m in class B");
      }
}

public class Test {
      public static void main (String args[]){
            Object var = new B();
            ((B)var).m();
            ((A)var).m();
      }
}
```

Output

(b) [6 marks] The following java program will output

```
class A {
      int A1 = 70;
      String A2 = "AA";
      public A (int A1) {
            this("AB");
            System.out.println("A1 = " + A1 + ", A2 = " +
A2);
      }
      public A(String A2) {
            this.A1 = 80;
            this.A2 = A2;
      }
}

class B extends A {
      public B() {
            super(90);
      }
}

public class C {
      public static void main(String[] args) {
            A What = new B();
      }
}
```

Circle the correct answer:

a. **A1 = 70, A2 = AA**

b. **A1 = 80, A2 = AB**

c. **A1 = 90, A2 = AA**

d. **A1 = 90, A2 = AB**

(d) [6 marks] Given the definition of the following classes, Print the output produced by the main method in the Test class.

```
class Base {
      public Base() {
            this("Base(String s)");
            System.out.println("Base()");
      }
      public Base(String s) {
            System.out.println(s);
      }
}

class Child extends Base {
      public Child(){
            this(4775);
            System.out.println("BChild");
      }
      public Child(int value) {
            super("Exam 1");
            System.out.println("BChild(int value)");
            System.out.println(value);
      }
}

class GrandCh extends Child {
      public GrandCh(String a) {
            System.out.println(a);
      }
}

class Test {
      public static void main(String args[]) {
            new GrandCh ("GrandCh Created");
      }
}
```

Output

(e) [7 marks] What is the output of the following program?

```
class G {
      int gr = 60;
      public G () {}
      public G (int gr) { this.gr=gr; ShowMe(); }
      public void ShowMe() {
            System.out.println("In G: gr = "+gr);
      }
}

class F extends G {
      int fa = 30;
      public F () {
            ShowMe();
      }
      public F (int gr) {
            super(gr);
            ShowMe();
      }
      public void ShowMe() {
            System.out.println("In F: gr ="+gr+"  fa = "+fa);
      }
}

public class Major1 {
      public static void main(String[] args) {
            G g1 = new G(50);
            G f1 = new F(40);
      }
}
```

Output

Question 3 [30 marks]

(a) [5 marks] A stack is a data structure that holds a set of elements and follows the idea of LIFO: Last In First Out, that is : the last element added is the first element removed. To add a new element in the stack, the method **`push`** is used. To get and then remove an element from the stack, the method **`pop`** is used. Define an interface **`Stack`** with only the two methods **`push`** and **`pop`**. The signatures of the two methods are:

- **`void push(Object obj)`**
- **`Object pop()`**

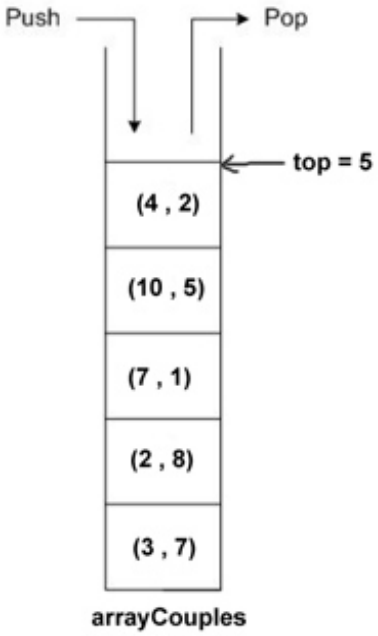
(b) [25 marks] The goal of this question is to implement a stack of integer couples (See the next figure).

You will have to define a class **`CoupleStack`** that implements the interface **`Stack`** and also to define a class **`Couple`** as an inner class in **`CoupleStack`**. Couple has only two instance variables of type **`int`** : **`left`** and **`right`**. Don't implement any accessor nor mutator for **`Couple`** class.

The class **`CoupleStack`** has two instance variables :

- **`arrayCouples`** which is an array of 10 **`Couple`** objects.
- **`top`** which is the number of couples in the stack.

Initially the stack is empty and as long as you push couples in the stack, the **`top`** variable increases. If you pop elements, the **`top`** decreases. Once you reach the maximum number of couples in the stack (10 couples), you should not push any other couple before popping at least one couple.



Give the definition of class **`CoupleStack`** (without accessors/mutators) as well as the inner class **`Couple`** and implement the methods: **`push`** (to add a **`Couple`** to the stack) and **`pop`** (to get and remove a **`Couple`** from the stack).

[blank page]

Question 4 [20 marks]

Write a program in java that does the following:

1. It prompts the user for an integer

2. It calculates the factorial of that integer in **another method (`public double factorial(int n)`).** Observe that the factorial of an integer *n* is given by:

factorial(*n*) = *n* (*n* – 1)(*n* – 2)… 3. 2. 1

For example
factorial(4) = 4.3.2.1 = 24

By definition,
factorial(0) = 1.

3. The resulting factorial is printed in the **`main`** method:

A sample output session is:

```
Enter an integer: 5
The factorial of 5 is 120.
```

4. Several exceptions could be thrown in this process.

(a) if the user enters a non-integer (for example **3.5**)

(b) if the user enters a negative integer (factorials of negative integers are not defined)

(c) if the user enters an integer whose factorial is too large to fit in the range **`int`**. (Typically java's primitive type **`int`** cannot calculate factorial(14) or higher correctly).

5. Use java's built-in exceptions and/or user defined exceptions to catch an exception and print an appropriate message for each one of the above cases separately. Make sure the program repeatedly asks for user input until the user enters a valid input. The program then prints the factorial of the input.

A sample output session is:

```
Enter an integer: 3.5
Exception: You entered a non-integer

Enter an integer: -3
Exception: You entered a negative integer

Enter an integer: 20
Exception: factorial(20) is too large for int data type

Enter an integer: 7
The factorial of 7 is 5040
```

Process Completed.

[blank page]