King Fahd University of Petroleum and Minerals
College of Computer Sciences and Engineering
Information and Computer Science Department
First Semester 2010/2011 (101)
ICS 102 - Introduction to Computing I

Final Exam
Saturday, January 29, 2011
Time: 150 minutes

Name: _____

ID#: _____ **C:** _____

Please circle your section number below:

Section	02	04	06	08
Instructor	Almuhammadi	Almuhammadi	Zhioua	Ghouti
Day and Time	SM 08:00-08:50	SM 10:00- 10:50	SM 13:10 - 14:00	SM 09:00-09:50

Note:

1. This is a closed book, closed notes exam.
2. Usage of calculators, laptops and cell phones is prohibited during the exam.
3. Please **switch off** your cell phones NOW.

Question #	Out Of	Score	Remarks
1	18		
2	30		
3	32		
4	20		
Total	**100**		

~Good Luck ~

Q1. [18 marks] Consider the following class definition:

```
public class Rectangle {
        private double length;
        private double width;
```

For each of the following methods in this class, write only the headers of the methods as described below (Do not provide the body of the methods).

Hint: The methods can be public, private, static, non-static, void … etc.

a) *getArea* - returns the area of this rectangle.

```
public double getArea() {
  return length*width;
}
```

b) *hasSameAreaAs* - takes another rectangle and checks whether it has the same area as this rectangle or not.

```
public boolean hasSameAreaAs(Rectangle ref) {
   if(length*width == ref.getArea())  //    if(getArea() == ref.getArea())
        return true;
  else
       return false;
}
```

c) *isSquare* - checks whether this rectangle is square or not.

```
public boolean isSquare() {
   if(length == width)
        return true;
  else
       return false;
}
```

d) *isSquare* - checks whether a given rectangle is square or not.

e) *adjust* - swaps the length and the width of this rectangle if the value of **length** is less than **width**.

```
public void adjust() {
    double tmp;
   if(length < width) {
      tmp = length;
      length = width;
      width = tmp;
   }
}
```

f) *mergeArea* - takes two rectangles and creates one rectangle with an area equals to the total area of the two given rectangles.

```
public Rectangle mergeArea(Rectangle ref1, Rectangle ref2) {
   Rectangle ref3 = new Rectangle(ref1.getLength() + ref2.getLength(),ref1.getWidth() +
ref2.getWidth());
        return ref3;
}
```

Q2. [30 marks] Consider the following definition of the class `Circle`

```
public class Circle {
  private double radius;   // the radius of the circle
  private double xCenter, yCenter;   // the coordinates of the center point
```

(a) Provide the following constructors and methods:

1- A *full-argument* constructor that creates a circle object and initializes its fields to a given **radius**, **xCenter** and **yCenter** (in this order).

```
public Circle(double radius, double xCenter, double yCenter) {
    this.radius = radius;
    this.xCenter = xCenter;
    this.yCenter = yCenter;
}
```

2- A *no-argument* (*default*) constructor that creates a circle object of radius = 1 and centered at the point of origion (0,0).

```
public Circle() {
  this(1.0, 0, 0);
// OR
     this.radius = 1.0;
     this.xCenter = 0.0;
     this.yCenter = 0.0;
}
```

3- A *copy constructor* for this class.

```
public Circle(Circle ref) {
   this.radius = ref.getRadius();
   this.xCenter = ref.getxCenter();
   this.yCenter = ref.getyCenter();
}
```

4- Accessor methods for **radius**, **xCenter** and **yCenter**.

```
public double getRadius {
  return this.radius;
}
public double getxCenter {
  return this.xCenter;
}
public double getyCenter {
  return this.yCenter;
}
```

5- Mutator methods for **radius**, **xCenter** and **yCenter**.

```
public void setRadius(double radius) {
   this.radius = radius;
}
public void setxCenter(double xCenter) {
   this.xCenter = xCenter;
}
```

```
public void setyCenter(double yCenter) {
   this.yCenter = yCenter;
}
```

6- A method **toString** that converts the information of this Circle object into a string as in the following example: **Circle of radius = 12.5 at (2.5, 3.0)**

```
public String toString() {
  return "Circle of radius = " + this.radius + " at (" + this.xCenter + ", " + this.yCenter + ")";
}
```

7- A method **equals** that returns **true** if this Circle object has the same radius and center point as a given Circle, and returns **false** otherwise.

```
public boolean equals(Circle ref) {
   if(this.radius == ref.getRadius() && this.xCenter == ref.getxCenter() && this.yCenter ==
ref.getyCenter())
        return true;
  else
       return false;
}
```

8- A method **getArea()** that returns the area of this Circle object.

Note: The areas of a circle whose radius is *r* is computed by: **area = πr^2**

```
public double getArea() {
  return Math.PI * this.radius * this.radius;
// OR
  return Math.PI * this.getRadius() * this.getRadius();
// OR
  return Math.PI * Math.pow(this.radius, 2);
//
  return Math.PI * Math.pow(this.getRadius(), 2);
}
```

(b) Write a main method that uses all methods and constructors of the class Circle and does the following tasks:

1- Create Circle *a* of radius = 5 centered at (3, 4).
2- Create a default circle *b*.
3- Print the location of Circle *a*.
4- Print all the information of Circle *b* and its area.
5- Create a new Circle *c* and make it identical to Circle *a*.
6- Move circle *b* four unites upwards. (**Hint**: use the accessor and mutator methods of yCenter)
7- Move Circle *c* three unites to the left and reduces its radius by 4.
8- Compare the two circles *b* and *c*, and print a proper message accordingly.

```
public static void main(String[] args) {
   Circle a = new Circle(5, 3, 4);
   Circle b = new Circle();
    System.out.println(a);
    System.out.println(b);
    System.out.println(b.getArea());
   Circle c = new Circle(a);
  b.setyCenter(b.getyCenter()+4);
  c.setxCenter(c.getxCenter()-3);
  c.setRadius(c.getRadius()-4);
  if(b.equals(c))
     System.out.println(b + " and " + c " are equal!");
  else
    System.out.println(b + " and " + c " are NOT equal!");
}
```

Q3. [32 marks] Consider the following class definition

```
public class List {
  private static final SIZE = 100;
  private int [] array;
```

(a) Write a *no-argument constructor* which creates a List with an array of length = SIZE.

```
public List() {
   this.array = new int[SIZE];
}
```

(b) Provide an accessor method for the instance variable **array**.

```
public int[] getArray() {
  return this.array;
}
```

(c) Provide a method ***fill*** that fills the **array** with values from the user (using Scanner).

import java.util.*;

```
public void fill() {
   Scanner myKeyb = new Scanner(System.in);
   for(int i = 0; i < array.length; i++) {
      System.out.print("Please fill array[" + (i+1) + "] = ");
      this.array[i] = myKeyb.nextInt();
   }
}
```

(d) Provide a method ***contains*** that checks whether a given integer is in this List or not.

```
public boolean contains(int x) {
   for(int i = 0; i < array.length; i++) {
       if(this.array[i] == x)
         return true;
   }
    return false;
}
```

(e) Provide a method ***equals*** which checks whether this List is exactly identical to a given List (i.e. it has the same elements in the same order).

```
public boolean equals(List other) {

  for(int i = 0; i < array.length; i++) {
       if(this.array[i] != other.array[i])
         return false;
  }
   return true;
}
```

(f) Write a main method which does the following tasks:

1- Create a List object called x
2- Fill x with values from the user
3- Print “6 is there” if x contains the integer 6 and “6 is not there” otherwise
4- Create a List object called y and fill it with values from the user.
5- Print “Same list” if x equals to y, and “Different list” otherwise.

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









}
```

Q4. [20 marks] Consider the following class definition

```
public class MyClass {
  private static int x = 5;
  public int y;
  public MyClass() { }
  public MyClass(int n) { y = n; }
  public int getX() { return x; }
  public int getY() { return y; }
  public String toString(){ return "" + "(" + x + "," + y + ")";}
  public void adjustX () { x++;  }
  public void adjustY () { y++;  }
  public void adjustY (int n) { y = ++n;  }
}  // end of MyClass
```

What is the output of the following error-free main method?

```
  public static void main(String[] args) {
                                        //Write the output in the corresponding boxes below
        MyClass a = new MyClass();

        System.out.println(a.getX());

        System.out.println(a.getY());


        MyClass b = new MyClass (20);

        System.out.println("b =" + b);


        int n = 10;

        a.adjustY(n);

        b.y = a.y;

        b.adjustX();

        b.adjustY();

        System.out.println("n =" + n);

        System.out.println(a+"/n\t"+b);


   } // end of main
```