Name and, if possible, ID#:__________

AMERICAN UNIVERSITY OF ARMENIA

College of Science and Engineering

COMP120 Introduction to Object-Oriented Programming

FINAL EXAM

15/15

Date: Monday, May 18 2015
Starting time: 09:20
Duration: 1 hour 40 minutes
Attention: **ANY TYPE OF COMMUNICATION IS PROHIBITED**

Please write down your name at the top of all used pages

Problem 1

Consider below a ***public interface Valuable*** that includes the only method ***public double value(double x)***:

```
public interface Valuable {

      public double value(double x);
}
```

1.1 Implement a ***public class Function*** that encapsulates a member variable of type ***Valuable*** and computes its integral in the specified range from x_1 to x_2 using the approximation:

$$\int_{x1}^{x2} f(x)dx \approx \frac{x_2 - x_1}{6}\left(f(x_1) + 4f\left(\frac{x_1 + x_2}{2}\right) + f(x_2)\right)$$

```
public class Function {

      private Valuable f;
      private double dx;

      public Function(Valuable newValuable, double newDX) {
            //TO BE IMPLEMENTED
      }

      public double integral(double x1, double x2) {
            //TO BE IMPLEMENTED
      }
```

1.2 Implement an expression

$$\sqrt{x^2 + a} + \sqrt{x^2 + b}$$

as a ***public class Roots*** that implements the interface ***Valuable*** and encapsulates double parameters ***a*** and ***b***. The parameters are initialized by the two-argument constructor ***public Roots(double newA, double newB)***;

1.3 In a separate ***public static void main(String args[])*** write a code that inputs two double values, creates an object of type ***Roots*** and, using the class ***Function***, prints the value of its integral from $x_1 = 1.0$ to $x_1 = 2.0$:

```
public static void main(String args[]) {
      Scanner input = new Scanner(System.in);
      double a = input.nextDouble(), b = input.nextDouble();

      //TO BE COMPLETED

}
```

OOP-FT8 180515. m107

Use the backside, if needed

```
private Valuable f;
private double dx;

~~public double value(double x) {~~
    ~~return x*x;~~
~~}~~

public Function (Valuable newValuable, double newDX {
        f = newValuable;
        dx = newDX;

    }
public double integral(double x1, double x2) {
 double integral;
integral = ((x2-x1)/6) * (f.value(x1) + 4*f.value((x1+x2)/2) +
                                              +f.value(x2));

return integral;

}
```

1.2

```
public class Roots implements Valuable {
 private double a;
 private double b;

public Roots (double newA, double newB) {
            a = newA;
            b = newB;
  }
```

cont.

Lree {

```
public double value ( double x) {
    return (Math.sqrt ( (x*x)+a) + Math.sqrt ((x*x) + b)) ;

}
```

[1.3]

```
public static void main ( String args [] ) {
        Scanner input = new Scanner (System.in);
 double a = input.nextDouble (), b = input.nextDouble();
  Function func   = new Function (   ...      , 1);
   func. integral ( 1.0, 2.0);
   Roots root = new Roots ( 2.0, 4.0);
     root.value ( 2.5);

}
```

5

roblem 2

```
public class Bishop extends ChessPiece {
private Rectangle cap;
public Bishop ( int size ) {
                                          super (size);
  ~~cap= new Rectangle ( size/4, /3)~~
  cap = new Rectangle ((size * 2)/3 , size/3 );

 }
~~public void drawCap (Graphics g) { cap.width~~
~~    g.drawOval (size/2, size/6, Rectangle.width~~
~~g.draw( cap.x, cap.y, cap.width, car~~
public void drawCap( Graphics g) {
g.drawOval (cap.x, cap.y, cap.width, cap. ~~length~~ height);

}
}
```

5

Lee {

Name and, if possible, ID#:____

```
public class Life extends Animator {

      private boolean grid[][] = new boolean[100][100];
      private int cellSize = 4;

      public void init() {
            for (int row = 0; row < grid.length; row++)
                  for (int col = 0; col < grid[0].length; col++)
                        grid[row][col] = Math.random() < 0.5;
      }

      private int sum9(int row, int col) {
            int result = grid[row][col] ? -1 : 0;

            for (int i = Math.max(0, row - 1);
                        i < Math.min(grid.length - 1, row + 1); i++)
                  for (int j = Math.max(0, col - 1);
                              j < Math.min(grid[0].length - 1, col+ 1); j++)
                        result += grid[i][j] ? 1 : 0;

            return result;
      }

      public boolean tick() {
            //TO BE IMPLEMENTED
      }

      public void snapshot(Graphics g) {
            //TO BE IMPLEMENTED
      }
}
```

```
public booleun tick() {
for(int row=0; row<girol.length; row++) {
   for(int col=0; col<girol[0].length; col++) {
      if (gird[row][col]==true && (sum9(row,col)<2||sum9(row,col)>3)){
            gird[row][col] = false;

      }
 if (gird[row][col]==true && (sum9(row,col)==2||sum9(row,col)==3)) {

   }
if (gird[row][col]== false && sum9(row,col)==3) {
            girol[row][col]==true;
   }
```

Use the backside, if needed

```
   }
 } return true;
  }
```

OOP-~~FIT~~ 180515.M107

```
for( int row = 0; row < grid.length ; row ++) {
  for(int cel = 0; col < grid[0]. length ; col++) {.
       if (gird [row][col] == false) {
            g.drawRect ( row * size, col * size, cellSize, cellSize);


       } else {
            g. fillRect(row * size, col * size, cellSize, cellSize);


       }

     }
   }
}
```

5