Name and, if possible, ID#:

AMERICAN UNIVERSITY OF ARMENIA

College of Science and Engineering

COMP120 Introduction to Object-Oriented Programming

FINAL EXAM

14/15

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 ~~f = newValuable(newDx);~~ f = newValuable;
      }                      ~~Valuable f = new Valueb~~le

      public double integral(double x1, double x2) {
            //TO BE IMPLEMENTED ~~double res = (x~~
      }
```

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.FT.180515.H102

```
{
   return ((X2-X1)/6)*( f.value(X1) + 4.0*f.value((X1+X2)/2.0)+
                                                        f.value(X2));
}

public class Roots implements Valuable {
   private double a, b;
      public Roots (double newA, double newB)
      {
            a = newA;
            b = newB;
      }
      public double value (double x)
      {
            return Math.sqrt(x*x+a) + Math.sqrt(x*x+b);
      }
   }
                                                                  5
// main
  double a = input.nextDouble();
  double b = input.nextDouble();                 Function f = new Function
  Roots tmp = new Roots(a,b);                                        (tmp);
  ~~Fx System.out.println( Roots~~        System.out.println(f.integral(1.0,2.0)
```

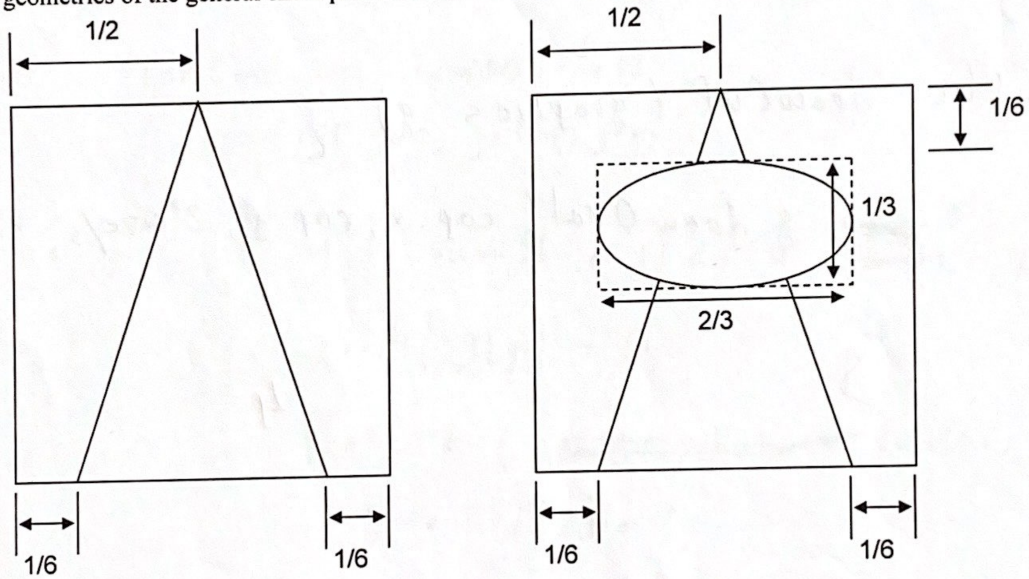
Name and, if possible, ID#:____________________________________

Problem 2

All 6 types of chess pieces can be drawn based on simple sketches consisting of a triangular base and rectangular cap. Consider below a ***public class ChessPiece*** that implements the triangular base only. Its geometry relative to the unit size of the square field is also sown:

```
public class ChessPiece {

      private Rectangle field;
      private Polygon base;

      public ChessPiece(int size) {
            field = new Rectangle(size, size);
            base = new Polygon(); //initially empty polygon
            base.addPoint(size / 6, size); //left vertex of the base
            base.addPoint(5 * size / 6, size); //right vertex of the base
            base.addPoint(size / 2, 0); //top vertex of the base
      }

      public void drawBase(Graphics g) {
            g.drawRect(field.x, field.y, field.width, field.height);
            g.drawPolygon(base);
      }

      public void drawCap(Graphics g) {
      }

      public void draw(Graphics g) {
            g.drawBase(g);
            g.drawCap(g);
      }
}
```

Extend a ***public class Bishop extends ChessPiece*** that encapsulates ***Rectangle cap*** member variable. Implement the constructor and override ***public void drawCap(Graphics g)***. The geometries of the general chess piece and the bishop are shown below:



OOP.FT.180515.H102

```
public Class Bishop extends ChessPiece {

  {     privente rectongte field;
        private polygon base; private oval cap;

public Bishop ( int size)

     {      super(size);
        field = new rectongle (size; size);

        base = new polygon ();
        base.addPoint (size/6; size);
        base. add Point (size*5 /6, size);
        base. addPoint (size/2,0);
        cap = new Oval ();
        cap.addPoint (size/6, size/6);

     }

   public drawCaP ( graphics g) {

        ~~cap~~. g.drawOval( cap.x, cap.y, 2*size/3; size/3);

        }
```

4

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 boolean tick() {

    for (int i=0; i<100; i++)
        for (int j=0; j<100; j++)
        { if ( sum9( i, j) <2 || ~~sum(9~~ sum9)(i,j) >3)
              grid[i][j] = 0;
          else if ( ~~sum9(i,j) ==2 ||~~ sum(9(i,j) ==3)
              grid [i][j] = 1;
        }
}
```

Use the backside, if needed

```
public void snapshot (Graphics g){

  int  i, j;

    for (i=0 ; i<100; i++)
      for( j=0; j<100; j++)

      { if (grid[i][j])

          {    g. fill Rect (4*j ; 4*i, 4, 4)

          }
              else
              {  g.drawRect (4*j ; 4*i, 4, 4)

          }

       }

   }
```

5