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          dx = new Dx; ~~=dx~~ f = new valuable;
      }

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

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)***; { a = new A ; b = newB ; }

Public double Value (double x) { Math.sqrt(x*x+a) + (Math.sqrt(x*x+b) ; }

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 $x_1 = 2.0$:

5

OOP.FT.180515.H087

```
public static void main(String args[]) {
      Scanner input = new Scanner(System.in);
      double a = input.nextDouble(), b = input.nextDouble();
      ~~Roots V(a , b)~~;
      //TO BE COMPLETED
      ~~function(V )~~;
      ~~d.integral (~~ ~~d.integral (1.0 , 2.0)~~;
}
      Valuable a[] = new Valuable[1]
```

Use the backside, if needed

```
a ~~Valuable~~ [0] = new Roots (a, b) ;
function z = new function(Valuable[0])
double T = ~~Valuable[0]~~ z. integral(1.0 , 2.0)
```

Name and, if possible, ID#:____

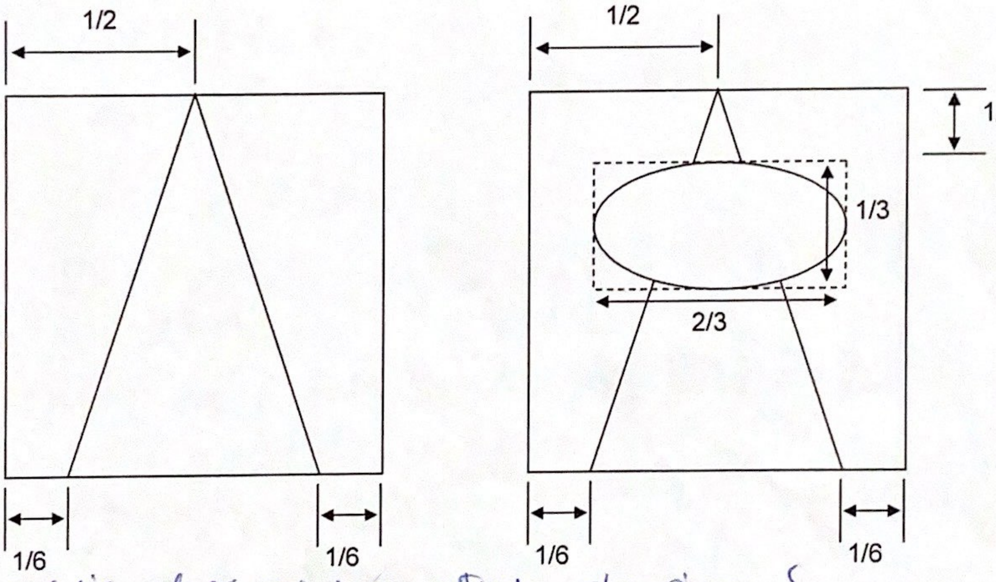
Problem 2

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);
       }
}
```

public Bishop(int size)
{ ~~cap = new Rectangle (size, size)~~
cap = new Retangle (Field.x+(size/6), (Field.y + (size/6)))

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:



5

public class Bishop extends ChessPiece {
privat Rectangle cap;
super(size); in constructor

OOP.FT.18.05.15.HQ87

Use the backside, if needed

public void drawCap (Graphics g){

~~2. fillov g.drawOval(size (size), size+(size/6), (2/3), (1/3)size)~~

2. draw oval(cap.x, cap.y, cap.width, cap.height)

2. drawOval(cap.x+~~(cap.x/6)~~, cap.y+(cap.x/6), (2/3)*cap.x, (1/3)*cap.x } ~~cap cap~~

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 void snapshot(Graphics g){
for(int i=0; i<100 ; i++)
 For(int j=0; j<100; j++) {
   if (grid[i][j]=="True")
    g.FillRect(j*4, i*4, 4, 4)
   if (grid[i][j]=="False")
    g.drawRect(i*4, j*4,4,4) }
                                 }
```

```
public boolean Tick() {
  For(int i=0 ; i< 100 ; i++)
   For( int j=0 ; j<100 ;j++ ) {
  if ( grid[i][j] =="True") {
  if ( sum9(i,j)>3 || sum9(i,j)< 2)
      grid[i][j]="false"
  else if ( sum9(i,j)==2 ||sum9(i,j)==3)
       grid[i][j]="True"
  ~~else~~if ( grid[i][j]== "False");}
        ~~if (grid[i][j]=~~                  {
   if( sum9(i,j) ==3)
        grid[i][j]="True"
                                    } }
```

5

OOP. FT. 180515. H087