Name and, if possible, ID

**AMERICAN UNIVERSITY OF ARMENIA**
*College of Science and Engineering*
**COMP120 Introduction to Object-Oriented Programming**

# FINAL EXAM

4/15

| | |
|---|---|
| **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 return value; double dx
      }

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

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

}
```

Public class Roots{
Private Valuable y;
Private double a, b
Public Roots (double newA, double newB);
a,b = newA, B
return
{

2

*Use the backside, if needed*



OOP FT 180515 L121

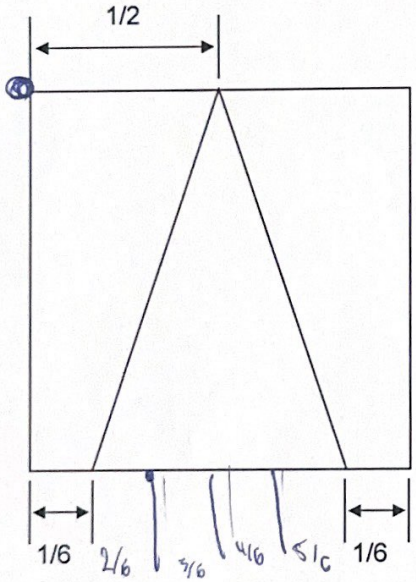
**Name and, if possible, ID#.**

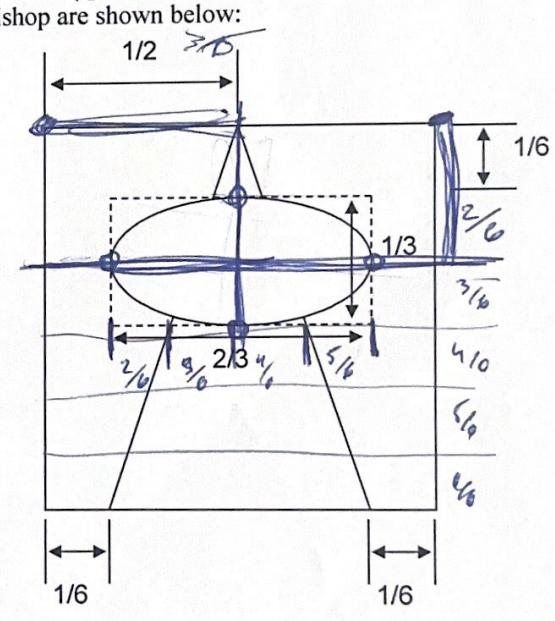
**Problem 2**

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:



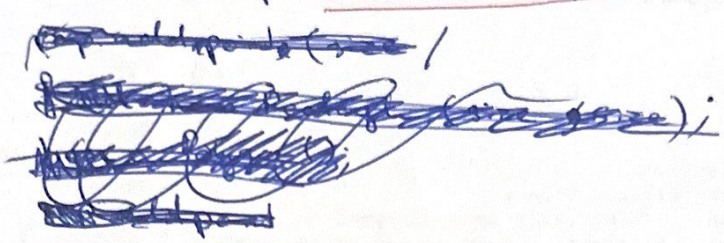

*Use the backside, if needed*



OOP-FT-180515-L121

rivate rectangle cap;

Public Bishop extend chesspiece (int size) {



cap = new rectangle (int 4/6 size, int 2/6 size)

}

Public void drawrectangle cap (graphics g) { cap=?

g.drawrectangle cap (field.1/6 size, field 1/6 size, field.width.4/6 size, field.height.2/6 size)

{

2

}