

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

FINAL EXAM

6/15

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 derivative at the specified point *x* using the approximation:

$$f'(x) \approx \frac{f(x+dx) - f(x-dx)}{(2 * dx)}$$

```
public class Function {  
    private Valuable f;  
    private double dx;  
  
    public Function(Valuable newValuable, double newDX) {  
        //TO BE IMPLEMENTED  
    }  
  
    public double derivative(double x) {  
        //TO BE IMPLEMENTED  
    }  
}
```

*dx = dx*

*f = newValuable;*

*double derivative = f(dx)*

*derivative = (f(x+dx) - f(x-dx)) / (2 \* dx)*

*return derivative;*

- 1.2 Implement an expression

$$\exp(-a * (x - c)^2)$$

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

- 1.3 In a separate *public static void main(String args[])* write a code that inputs two double values, creates an object of type *Gauss* and, using the class *Function*, prints the value of its derivative at the *x = 1.0* point:

```
public static void main(String args[]) {  
    Scanner input = new Scanner(System.in);  
    double a = input.nextDouble(), c = input.nextDouble();  
  
    //TO BE COMPLETED  
}
```

*Gauss g = new Gauss(a, c);*

*Function f = new Function(g, 1.0)*

*double derivative = f.derivative(1.0);*

*System.out.print("Derivative = ");*

*System.out.println(derivative);*

*}*

*See SS*

Use the backside, if needed

Page 1 of 4

OOP-PT-180515-L123

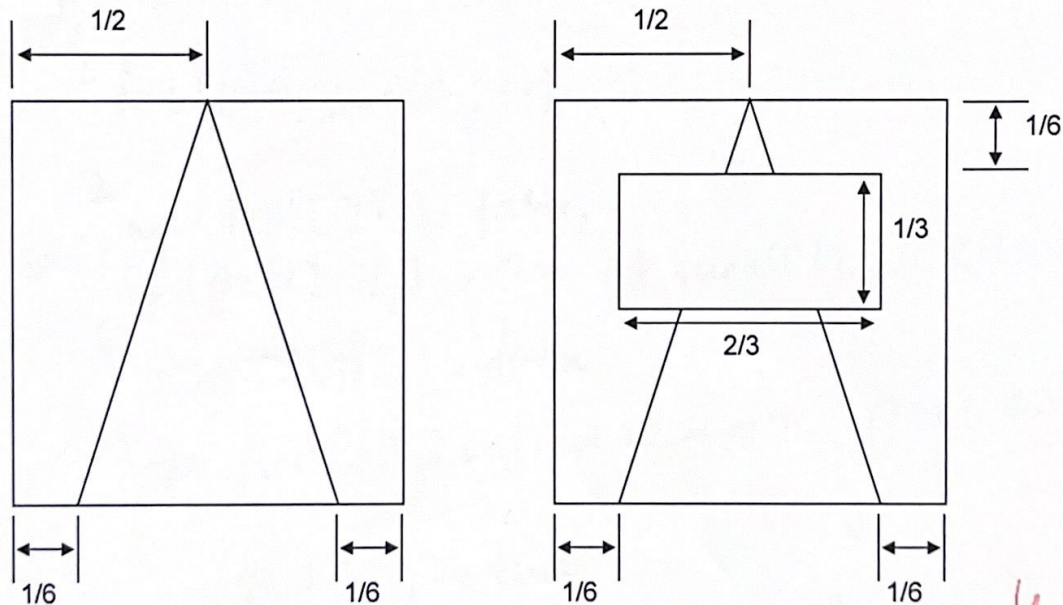


## Problem 2

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 shown:

```
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 Rook 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 rook are shown below:



Use the backside, if needed

Page 2 of 4

004-PF-130315-1/23



```

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
    }
}

```

0  
see SM, KG, GT,  
SS

```

int i, j;
for (i = 0; i < 100; i++)
{
    for (j = 0; j < 100; j++)
    {
        newGrid[i][j] = false;
        if (grid[i][j] == true && sum9(i, j) < 2 || sum9(i, j) > 3)
        {
            newGrid[i][j] = false;
        }
        if (grid[i][j] == true && sum9(i, j) == 2 || sum9(i, j) == 3)
        {
            newGrid[i][j] = true;
        }
        if (grid[i][j] == false && sum9(i, j) == 3)
        {
            newGrid[i][j] = true;
        }
    }
}
grid = newGrid;

```

Use the backside, if needed

Page 4 of 4

OOP-FI-180315-L125