Name and, if possible, ID#: _____

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

FINAL EXAM

10/15

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 max in the specified range from x_1 to x_2 by looking at:

$f(x_1), f(x_1+dx), f(x_1+2dx), \ldots, f(x_1+k*dx)$, where $k = 1, 2, \ldots$ and $x_1+k*dx < x_2$

```
public class Function {

      private Valuable f;
      private double dx;

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

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

1.2 Implement an expression

$$a * sin(x) + b * cos(x)$$

as a ***public class Harmonic*** that implements the interface ***Valuable*** and encapsulates double parameters ***a*** and ***b***. The parameters are initialized by the two-argument constructor ***public Harmonic(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 ***Harmonic*** and, using the class ***Function***, prints its maximal value in the range from $x_1 = -1.5$ to $x_1 = 1.5$:

```
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.H095

Use the backside, if needed

- newDx) {

for (int i = 0 , i > 0 , i++) {

double Funct = X, +(i)(newDx)

I was not try to translate the function

1

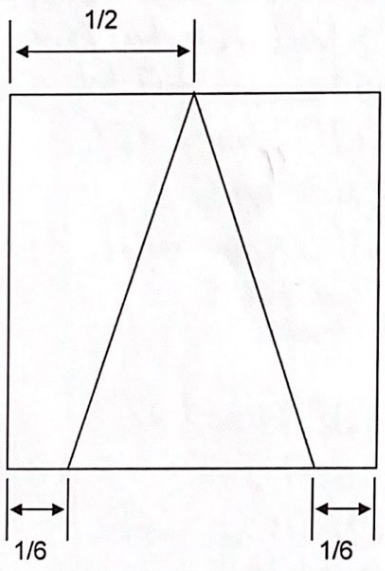
Name and, if possible, ID#:_____

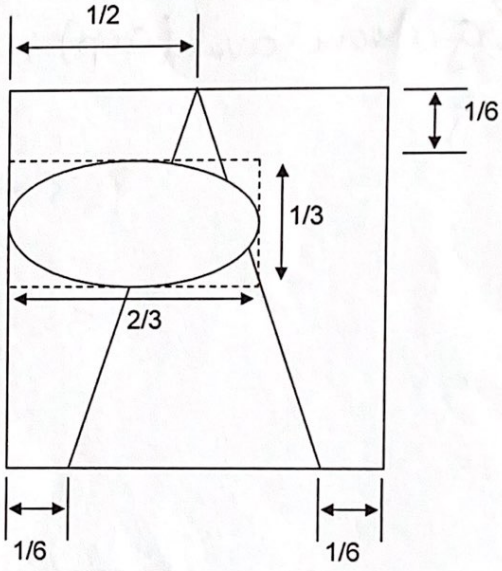
Problem 2

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 Knight 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 knight are shown below:





OOP.FT.180515.1095

Use the backside, if needed

Or

```
public class Knight extend class piece {

 private Rectangle cap;
 constructor - ?

public void drawcap(Graphics g){
   g.chesspiece(size)  ⟹ I don't It Its true, I wanted to restrain it again
   Or.
   g.drawOval(0, size*.1/6 , size*1/3, size*2/3);
```

or

```
   cap = new oval(0, size*1/6 , size*1/3, size*2/3);
```

not correct

```
   public void drawcap(Graphics g){
   g.drawoval(cap);
   }
```

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 row=0, row< grid.length, row++){
for( int col=0, col< grid.lenght, col++)
   int free = sum9(i,j);
    IF (grid[i][j]==true){
        IF( free<2 || free>3){
return  grid[i][j]=false;
      } else{
          return grid[i][j]=true;
                }
IF (grid[i][j]==false){
     IF( free==3){
return grid[i][j]=true;
      } else,{ grid[i][j]=false;
          return
      }
}
```

OOP.FT.180515.H095

Use the backside, if needed

```
public void snapsho(Graphics g){
    For (int row =0, row< grid.length, row++) {
        for (int col=0 , col< grid.length , col++)
```

I drawed all sq

```
            [redacted] =
            g.drawRect(col*4 , row*4 , 4 , 4);
```

and filled the true ones

```
            IF (Grid[i][j] == true){
                g. setColor(Color.BLACK);
            For g.FillRect(col*4 , row*4 , 4, 4);
            }
```