General Instructions:

1. Follow the instructions and especially the naming standards.
2. Put the source files in a folder named {**YOURID}_{FIRSTNAME}_{LASTNAME}_AS2** and compress it**. (**Use **.zip** format**)**
3. **Note:** replace your info, like **123456_ALI_ALILI_AS2.zip** be careful with the order(id, first name, last name).
4. You will lose points for not following the naming standard.
5. Be as clear and neat as possible when you write codes. Use naming conventions and indentations properly.
6. **Neither plagiarism nor any type of cheating will be tolerated!**

Assignment 2

We have already practiced defining Point and Circle classes and using them.

We will use the similar Point class here.

Point.java

- Define the class **Point**. What should be the access modifier of this class? Why?
- Define **x** and **y** fields both of type float.
- Define a constructor which will take
    a. two parameters, **x** and **y** coordinates resp. and assign the fields
    b. no parameters (remember default constructors?) and assign the fields to **0.0**.
    c. single parameter (remember copy constructor?)
- Define accessor (getter) methods for each of the fields(instance attributes) of the class(type).
- Test all above in the Main class.

- Define a method **void translate(float dX, float dY);** which will translate the point.
- Define a method **double distance(Point p);** which will find the distance in between **this** point and **p** point.
- Define a method **static double distance(Point p1, Point p2);** which will find the distance in between **p1** point and **p2** point. What is the difference between the last two methods?

- Define a method **int inWhichQuadrant();** which will return one of [1,2,3,4], the quadrant that **this** point belongs to or 0(ZERO) if it lies on the coordinate axis. *Ex: new Point(3,5).inWhichQuadrant() returns 1 whereas new Point(3,-5).inWhichQuadrant() returns 4.*
- Define a method **static int inWhichQuadrant(Point p);** which will return one of [1,2,3,4], the quadrant that point **p** belongs to. What is the difference between the last two methods?
- Define a method **String toString();** which will return the String format. *Ex: (0.0, 1.0)*
- Define a method **boolean equals(Point p);** which will return if true the given point **p** is equal (the same x and y coordinates) to **this** point, false otherwise..
- Test all above in the Main class.

Now since we already have a fully functional Point class, let us create another using Point.

Rectangle.java

- Define the class **Rectangle**. What should be the access modifier of this class? Why?
- Define **topLeft** and **bottomRight** fields both of type **Point**.
    a. *Note: Any rectangle with sides parallel to x and y axis can be represented by its top left and bottom right vertices. Assume that we are working with these rectangles only*
- Define a constructor which will take
    a. two parameters of type Point. topLeft and bottomRight vertices resp. and assign the fields.
    b. four parameters of type float. First two represent the coordinates of the topLeft vertex and last two represent coordinates of the bottomRight vertex.
    c. single parameter (remember copy constructor?)
    d. *Can you explain what kind of association is there in between Point and Rectangle classes?*
- Provide getter methods for both fields.
- Define a method **void translate(float dX, float dY);** which will translate the rectangle.
    a. *Note: Consider using the translate method of Point class to delegate the job.*
- Define a method **double area();** which will calculate and return the area represented by this instance.
- Define a method **double perimeter();** which will calculate and return the perimeter represented by this instance.
- Define a method **int inWhichQuadrant();** which will return one of [1,2,3,4], if all the vertices of this rectangle is in the same quadrant and **0** otherwise.
- Define a method **String toString();** which will return the String format. *Ex: [(0.0, 5.0),(7.0, 2.0)]*

- Define a method **boolean equals(Rectangle r);** which will return true if the given rectangle **r** is equal (the same vertices) to **this** rectangle, false otherwise.
- Test all above in the Main class.

Let us add another class using Point.

Triangle.java

- Define the class **Triangle**. What should be the access modifier of this class? Why?
- Define **vertex1, vertex2, vertex3** fields all of type **Point**.
    a. *Note: Any triangle can be represented by its three vertices.*
- Define a constructor which will take
    a. three parameters of type Point. Assign fields accordingly.
    b. single parameter (remember copy constructor?)
    c. *Can you explain what kind of association is there in between Point and Triangle classes?*
- Provide getter methods for all fields.
- Define a method **void translate(float dX, float dY);** which will translate the triangle.
    a. *Note: Consider using the translate method of Point class to delegate the job.*
- Define a method **double area();** which will calculate and return the area represented by this instance.
- Define a method **double perimeter();** which will calculate and return the perimeter represented by this instance.
- Define a method **int inWhichQuarter();** which will return one of [1,2,3,4], if all the corners of this triangle is in the same quarter and **0** otherwise.
- Define a method **String toString();** which will return the String format. *Ex: [(0.0, 0.0),(0.0, 3.0),(4.0, 0.0)]*
- Define a method **boolean equals(Triangle t);** which will return true if the given triangle **t** is equal (the same vertices) to **this** triangle, false otherwise.
- Test all above in the Main class.

Once you tested all the classes separately, now add all of them into one package, **geometry** and add necessary statements in each class.

Create another package **main** which has only one class, **Main**. This class might look like the following:

```java
package main;
import geometry.Rectangle;
import geometry.Triangle;
import geometry.Point;

public class Main {
    public static void main(String[] args) {
        Rectangle r1 = new Rectangle(new Point(0, 5), new Point(6, 0));
        Rectangle r2 = new Rectangle(0, 3, 4, 0);
        Rectangle r3 = new Rectangle(r2);
        Rectangle r4 = new Rectangle(new Point(-3, 5), new Point(-1, 2));

        System.out.println(r1);
        System.out.println(r2);
        System.out.println(r3);

        System.out.println(r4.inWhichQuadrant());

        System.out.println(r1.equals(r2));
        System.out.println(r2.equals(r3));

        System.out.println(r2.area());
        System.out.println(r2.perimeter());

        Triangle t1 = new Triangle(new Point(3, 5), new Point(), new Point(1, 4));
        Triangle t2 = new Triangle(t1);
        Triangle t3 = new Triangle(new Point(3,4),new Point(1,2),new Point(1, 3));

        System.out.println(t1);
        System.out.println(t2);
        System.out.println(t3);

        System.out.println(t3.inWhichQuadrant());

        System.out.println(t1.equals(t2));
        System.out.println(t2.equals(t3));

        System.out.println(t1.area());
        System.out.println(t1.perimeter());
    }
}
```

And output might look like:

```
[(0.0,5.0),(6.0,0.0)]
[(0.0,3.0),(4.0,0.0)]
[(0.0,3.0),(4.0,0.0)]
2
false
true
12.0
14.0
[(3.0,5.0),(0.0,0.0),(1.0,4.0)]
[(3.0,5.0),(0.0,0.0),(1.0,4.0)]
[(3.0,4.0),(1.0,2.0),(1.0,3.0)]
1
true
false
3.4999999999999996
12.19012549796275
```