

CMPE 160, Spring 2015

# Project-1

Cagatay Yildiz, Can Tunca, Haluk O. Bingol

March 12, 2015

**Due:** 30.03.2015, 09.00 AM.

**Submission:** Will be announced later.

## Contents

1	Description	1
2	Details of Given Code	2
3	Using GUI	2
4	Goals	3
5	Details of Given Code	4
6	Testing Your Code	4
7	Bonus	4
8	Hints for Implementation	4

## 1 Description

In this project, you have two different tasks:

- You are going to design and implement a **Java** library that will be used for drawing that is necessary for a primary school level geometry book.
- Using the GUI provided by us, your objects are going to move in a plane.

In the .zip folder, you can see a number of packages and classes. Do not get rid of these files although some of them are empty. These classes supposed to give you some insight about what your library should look like. More specifically, they exemplify the correct usage of interfaces and inheritance.

One thing to do in the project is to make this library richer. What kind of classes your library is going to have is a part of your design but you are expected to add at least 3 different objects that do not have 4 edges (triangle, hexagon, etc). You are welcomed to add as many features as you want. But do not forget: The main goal is not to draw some weird polygon with 25 edges but to learn how to use object oriented programming.

In the second part of this project, you are going to use your library. In a main class (see `bingol.test.Main`), you will create objects and see their movement on a plane. In the next PS hours, your assistants are going to show you a Java library, that you will be using for this purpose.

## 2 Details of Given Code

**Warning.** Do not change anything in the packages that starts with `cmpe160.s20151.project1`. Consider that they are given to you as a library of `.class` files or simply a `.jar` file, as in the case of `java.awt.*`. In addition, the owner is nice enough to share with you its implementation in `.java` files. Hence you cannot change the library. Note that when we do test your submissions, we will have your codes, that is, packages starts with `student.yourLastName.yourName.project1.*`, and include our `cmpe160.s20151.project1.*` library. Therefore any changes you make in our library will be lost. So there is a possibility that your code will not work if you make changes in our library.

These packages are like the backbone of your design. You can `extend` or `implement` those but cannot change the source code itself. Check javadoc for documentation of this family of packages.

Your own package structure is `student.yourLastName.yourName.project1`. Feel free add sub-packages to your own package structure in which all packages start with `student.yourLastName.yourName.project1`. For instance, if I were you, I would add a class named `MyTriangleInterface` to the package with a name `student.yourLastName.yourName.project1.interfaces`. Later on you will be using the same structure for your Project2 which will be in `student.yourLastName.yourName.project2`.

A valid question would be what to do if the source code we gave is not enough for your purposes? For example, `Quadrilateral` class is implemented in `cmpe160.s20151.project1.implementations` but some functionality you will later need are missing. In this case, what you should do is to create a new class named `MyQuadrilateral`, which extends `cmpe160.s20151.project1.implementations.Quadrilateral` and to use `MyQuadrilateral` while creating instances. This way, you not only avoid changing the source code that we provided, but you also implement what is needed.

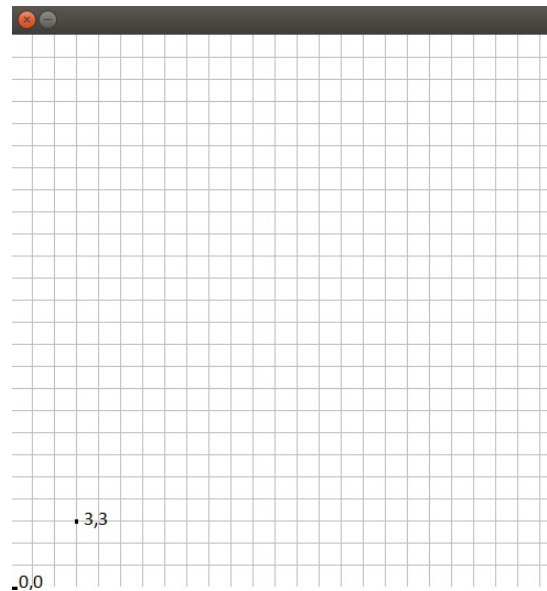
## 3 Using GUI

Using GUI is pretty easy. An example is in `cmpe160.s20151.project1.implementations.Main`, which is also given in Listing 1.

Listing 1: Sample code

```
public static void Test(String[] args) {
    GUI gui = new GUI();
    AnimationCanvas canvas = new AnimationCanvas();
    gui.addCanvas(canvas);
    /* YOUR CODE: START */
    canvas.addObject(new Square(new Point(1, 13), new Point(4, 10)));
    /* YOUR CODE: END */
    canvas.start();
}
```

As you run this program, the following canvas will appear:



Here, each unit square is 1-by-1 and this canvas is 25-by-25.

In the second part of the project, you are going to override `draw` and `move` methods so that your objects are going to move in this 2D grid. Here is a list of thing to do:

- Each object is going to start their movements at random position and move in random directions.
- As an object hits to an edge, it is going to reflect. So, your objects are basically billiard-balls and this grid is a billiard table with no holes on edges.
- Another result of hitting a wall is the creation of another object of the same type (remember `instanceof`). For instance, if a `MyQuadrilateral` object hits an edge, another `MyQuadrilateral` object will be created. If it is `MySquare`, a new `MySquare` will be created. The new object will start its movement from the exact location of the parent when it hits the wall and move in random directions.
- While creating new objects as a result of hits, you are expected to keep the number of objects below a threshold. This threshold is equal to the number of edges for each and every class: For example at the same time, there must be less than 5 objects with 4 edges, less than 7 objects with 6 edges, etc.

## 4 Goals

In this project we do not want you to implement an API of size megabytes. The main goal is to learn, given you are a problem, how to make an appropriate design (in terms of functionality and usage of object oriented programming) as well as improving your coding skills. More specifically, we are interested in the following:

**Design.** Usage of appropriate objects, fields, methods, `public` and `private` keywords etc.

**Object Hierarchy.** Usage of `extends` and `implements` keywords, etc.

**Documentation.** Usage of Javadoc

**Implementation.** A proper implementation of your design.

**Testing.** Write down some piece of code like the one in `cmpe160.s20151.project1.test` and document what it does. Use package `student.yourlastname.yourname.project1.test` only for your test routines.

## 5 Details of Given Code

Do not change anything in the packages of `cmpe160.s20151.project1`. Check javadoc for documentation of this family of packages.

Your own package structure is `student.yourLastName.yourName.project1`. Feel free add sub packages, as in the case of `cmpe160.s20151.project1.test`, to your own package structure. Note that you have to change to your name. Later on you will be using the same structure for your Project2 which will be in `student.yourLastName.yourName.project2`.

## 6 Testing Your Code

Every class you write should have the following method for its testing purposes. What we want in test class is not each and every functionality of your code but some main functionalities:

```
public static void Test(String[] args) {  
// TODO Your test should be here  
}
```

There should be one more class that tests the entire project, that is, not the individual classes but their interaction that makes the project itself. So basically, you are going to draw these objects to the canvas and show how they move. See `cmpe160.s20151.project1.test`.

## 7 Bonus

Additional features, fully tested, properly documented.

- Changing colors of objects as two objects collide. Here, collision occurs if one corner of an object is within the other one. You may use any color you want.
- Implementing a blackhole. Without changing the code that we provide, can you create some kind of a blackhole in the grid? If an object falls into the blackhole, it will get lost and immediately appears at a random location and start moving at random directions. The size of the blackhole should not be any greater than 2-by-2. (**Hint:** You may want to create `MyCanvas` object that extends `Canvas`).
- You are welcomed to implement some additional features. If you think they are so cool, write a separate test class and document its functionality.

## 8 Hints for Implementation

Your library should support the following functionalities, noting that this list is **not** exhaustive:

- Use your own package. It starts with `student.yourLastName.yourName.project1`. Of course you need to use your own name.
- Initialization of many kinds of objects.
- Changing features of objects.
- Printing some information about objects.
- Calculating areas, circumferences and some other features of geometric shapes.

Some cool ideas would be:

- Moving objects,
- Rotating objects,
- Avoiding collisions between objects,
- Fixing all collisions between objects,
- Hiding or removing objects.