Lab 5: Writing a Simple Class

Introduction

In the text and in lecture, we saw some examples of using and writing simple classes in Java. We saw the basics of how instance variables, constructors, accessor methods, and mutator methods are declared and used. We also saw how to accomplish encapsulation with data hiding by declaring our instance variables to be `private` and by declaring our instance methods to be `public`. Although, this will not always be the case, it is a good initial rule-of-thumb to use for writing new classes. In this lab, you will complete a new, simple class and see it work by running it with a simple driver program.

MyRectangle Class

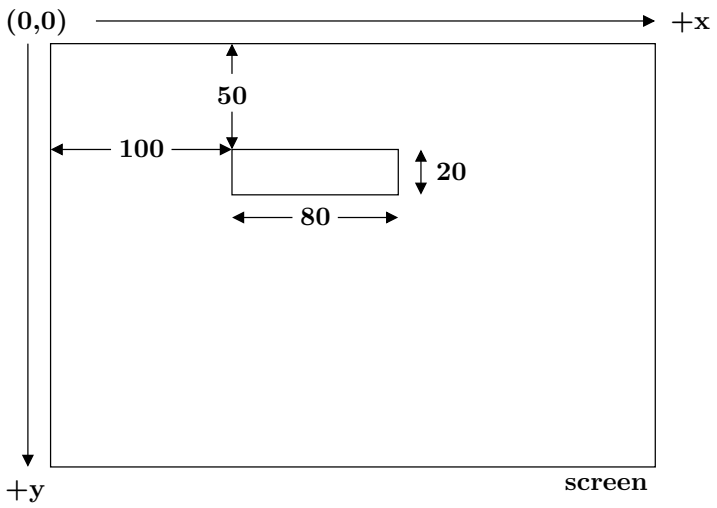
You will write a simple class that will represent a rectangle. Later on, we will see how to do this with graphics, but for now, we will just have to use our imaginations for the visual representations of the rectangles.

Instance Variables

Recall that an object is constructed using a class as its blueprint. So, think about an rectangle on your monitor screen. To actually draw a rectangle on your monitor screen, we need a number of data. Clearly, a rectangle requires the dimension of the sides. We will call these dimensions `width` (for the X-dimension) and `height` (for the Y-dimension). The types of these dimensions could be either floating point (`double`) or integer (`int`). Since pixels on the computer display are discrete, we will make these dimensions `int`. In addition to its size dimensions, a `MyRectangle` must have a position (location) on the screen. We will base this position on the upper left corner of the `MyRectangle`. The graphic coordinate space for computers start a (0, 0) and proceeds to the right for positive X and down for positive Y. Call the X position `startX` and call the Y position `startY`, and like the dimension variables, make these both `int`s. For example, consider the following instance variable values for a `MyRectangle`:

`startX`	100
`startY`	50
`width`	80
`height`	20

This would define the `MyRectangle` shown (not exactly to scale) in the figure below:



We do not want users of `MyRectangle` to have direct access to the instance variables, so will will declare them as `private`.

Constructors

Recall that constructor methods are used to create new objects of a given class. They are special methods in that they have no return type (not even `void`). For the `MyRectangle` class, you will have two constructors. One is a **default constructor** – this is used to create objects when no arguments are used. For `MyRectangle`, the default constructor will initialize all 4 of its instance variables to 0. The second constructor will have four parameters (in order): X position, Y position, new width, and new height, and it will simply initialize the instance variables from the parameters.

For example, the `MyRectangle` above could be created by the following:

```
MyRectangle rect = new MyRectangle(100, 50, 80, 20);
```

and a "default" `MyRectangle` could be created by the following:

```
MyRectangle rect = new MyRectangle();
```

Accessors

Accessor methods allow us to get information from objects or have them perform tasks that do not alter the object themselves. There is no special designation for accessor methods; rather we label them as accessors based on what we use them for. For `MyRectangle`, we will have the following accessors:

- `public int area()`: returns the area of the given `MyRectangle`
- `public String toString()`: returns the `MyRectangle`'s information in the form of a String. See sample output for details.
- `public boolean isInside(int x, int y)`: returns true if point `(x, y)` is inside the `MyRectangle`, and false otherwise.

Mutators

Mutators allow us to change the data within an object. As with accessors, there is no special designator for mutators; we label them based on what we use them for. For `MyRectangle`, we will have the following mutators:

- `public void setSize(int newWidth, int newHeight)`: changes `width` and `height` of this `MyRectangle` to the values passed in.
- `public void setPosition(int newX, int newY)`: changes X and Y positions of this `MyRectangle` to the values passed in.

Class Skeleton

A skeleton of the class above, named `MyRectangle.java`, is posted on the CourseWeb. Download this file onto your account or computer and fill in the details so that the class will work as intended.

Main Program

Main program in Object-Oriented Programming (OOP) are typically simple and used primarily to "set up" the objects, which in turn do the rest of the work of the execution. However, in this case, our main program will just demonstrate the functionality of our new class. It is provided on the CourseWeb under this lab (`Lab05.java`). Download this file and compile once you have completed the details of `MyRectangle.java`. Then run it to make sure it works as intended. Also read the comments carefully – some additional important / useful information is provided there. Sample output from the program to compare with yours is also on the CourseWeb (`lab05out.txt`).

Grading

For this lab, your TA will use the following grading criteria

1. (2 points) Constructors work correctly
2. (3 points) Methods `area()` and `toString()` work correctly (1.5 points each)
3. (2 points) The `isInside()` method works correctly.
4. (3 points) Mutator Methods work correctly (1.5 points each)

Due Date and Submission

Once you completed the program, you must demonstrate your program for your Lab TA. Once your TA already checked you, **DO NOT FORGET** to submit your `MyRectangle.java` file to the CourseWeb under this lab by the due date.

If you do not complete the lab this week, you may finish it and submit your code to the CourseWeb before the due date. However, you need to demonstrate it to your TA at the beginning of next week's lab.

No late submission will be accepted.