Unit 5: Lab - House

Unit & Day
5.6

Last Edited
February 24, 2025 1:03 PM

Documentation

Starter Code

https://harvardwestlake.sharepoint.com/:f:/s/Usmath/Elc7o440ckdIka2Npo2yjIYB_jncVdhLyCuZoD5gZiNj_w?e=HBaWfE

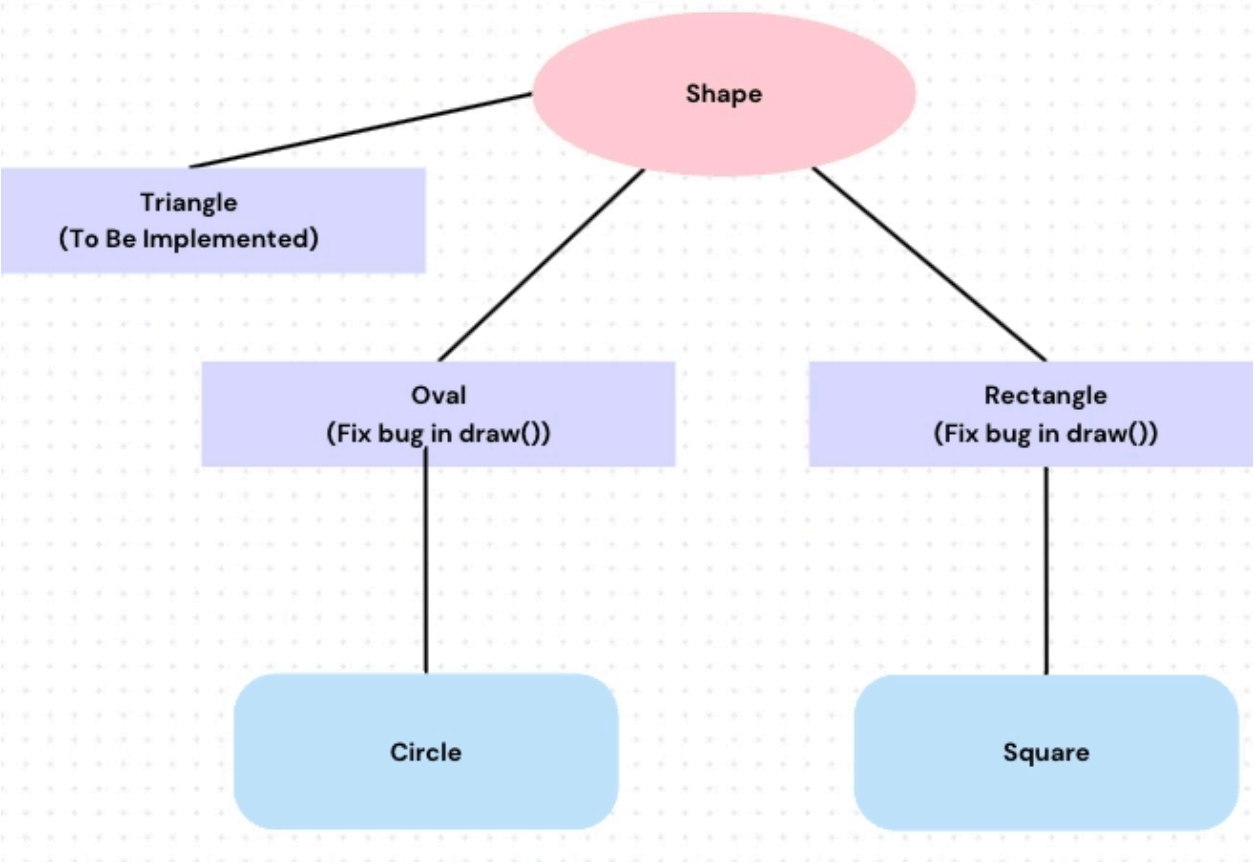
About Lab:

Starter Code: Download the **Starter Code** and place it in a folder named “Unit 5 House Lab”.

Tip: Read through each given class to understand how they work!

In this lab, you’ll receive a starter class hierarchy derived from the Shapes Lab.

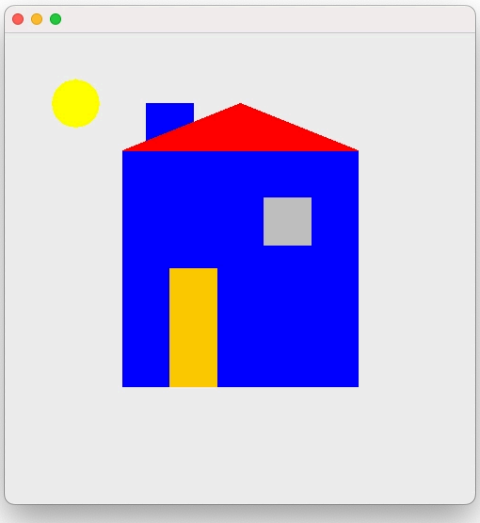
This class features the following class structure:



Goal:

The major goal of this lab is to draw a house!

All of the classes except for the `Triangle` class are already implemented.



⚠️ **Warning:** There is a bug in the draw methods for `Oval` and `Rectangle` that doesn't break your code. It's recommended to fix this bug *after* implementing `Triangle`.

Shape Class:

xCoord and yCoord Update

In the previous lab, the program required a single (x, y) coordinate point to create an `Oval` or a `Rectangle` object. We stored these x and y coordinates in separate variables.

Now, to draw a triangle, we need three (x, y) coordinate points. This raises a question: how can we modify the `Shape` class to accommodate triangles while maintaining compatibility with `Oval` and `Rectangle` classes?

We could add four more variables to store the additional coordinate points. While this would work, it's not ideal— `Rectangle` and `Oval` classes would initialize a third pair of unused coordinates set to $(0, 0)$.

Also, consider what would happen if we needed to implement more complex polygons in the future. Would we really want to create 100 x-coordinate variables and 100 y-coordinate variables for the `Hectogon` class?

Instead, `Shape` implements a more elegant solution by using an array of x-coordinates and an array of y-coordinates:

```
protected int xCoord[]; protected int yCoord[];
```

The array indices correspond to the coordinate points, matching the mathematical notation of (x_1, y_1), (x_2, y_2), and so on. Check out the Rectangle Class section below to see how both `Rectangle` and `Oval` classes adapted to this change!

Color and Stroke Weight Update

Building on the previous lab's stretch challenge, the Shape class now includes three additional instance variables:

```
protected Color fillColor; protected Color strokeColor; protected int strokeWidt
h;
```

These instance variables are not initialized in any constructor. Instead, users need to set these values after creating an object using setter methods (see `MyPanel` for examples).

You may modify this approach if you prefer setting colors directly in the constructor.

For your convenience, setter methods for these properties are included in the `Shape` class.

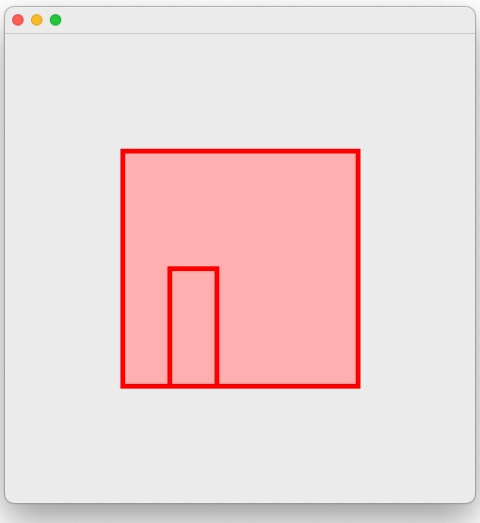
The overloaded methods that accept three integers represent RGB (Red, Green, Blue) color values.

```
// accepts a color object (i.e. Color.red) public void setFillColor(Color fillCol
or); // accepts an RGB color (i.e. 255, 200, 255) public void setFillColor(int r,
int g, int b); // accepts a color object (i.e. Color.red) public void setStrokeCo
lor(Color strokeColor); // accepts an RGB color (i.e. 255, 200, 255) public void
setStrokeColor(int r, int g, int b); // accepts an integer public void setStrokeW
idth(int strokeWidth);
```

✅ **Tip:** Google has a convenient color picker! Use the RGB values in the setters!

> Google
>
> Search the world's information, including webpages, images, videos and more. Google has many special features to help you find exactly what you're looking for.
>
> https://g.co/kgs/weXARFV

MyFrame and MyPanel Class:



These two classes maintain their core functionality from before. They extend Java's `JFrame` and `JPanel` classes, respectively.

You'll modify `MyPanel` to draw your house. Initially, the house will be basic (see image on the left).

⚠️ **Warning:** There is a bug where the `setFillColor()` function isn't working properly.

It's recommended to fix this bug *after* implementing `Triangle`.

Rectangle Class:

Recall that in the `Shape` class, we modified our `xCoord` and `yCoord` instance variables to store an *array* of coordinates.

How does this affect the `Rectangle` class?

Previously, we initialized a `Rectangle` object using this syntax:

```
Rectangle r = new Rectangle(2, 4, 30, 30);
```

The first two parameters represent the coordinate point, while the last two specify the dimensions.

With the changes to the `Shape` class, we would need to initialize it like this:

```
int[] xCoord = {2}; int[] yCoord = {4}; Rectangle r = new Rectangle(xCoord, yCoor
d, 30, 30);
```

This seems *annoying*.

Does the end-user *really* need to create a single-element array every time they want to construct a `Rectangle`? ***NO!***

Look at the constructor of the `Rectangle` class. The signature still takes an individual `xCoord` and `yCoord` value, but it creates a `new int[]` array ***within*** the constructor.

This approach has an additional benefit: it prevents users from creating `Rectangle` objects with arrays larger than one element.

By moving this responsibility from the user to the class itself, we've made everything more convenient!

Overridden `setCoordinates()` update

The `setCoordinates()` method in `Shape` is overridden to make changing coordinate points easier. While users can still set coordinates using arrays, this won't affect our current needs.

Square:

This class extends `Rectangle`. Read through the code to understand its structure—no implementation is required.

Oval:

Like the `Rectangle` class, this fully implemented class handles coordinate points in an array. Review the code to understand how each method works.

Unlike the previous lab which only allowed circles, this class lets you create any `Oval` object.

Circle:

This class extends `Oval`, do read the code to understand its structure. No implementation required.

Triangle (Implementation Required):

This class extends `Shape`. The only things that require implementation are the Constructors and the `draw()` method. The starter code provides you a text file, you must convert this to a java file before you begin.

This class does not have any instance variables.

- **`Triangle(int[] xCoord, int[] yCoord)`**

  A 2-param constructor. This constructor initializes the two arrays found in `Shape`.

- **`draw()`**

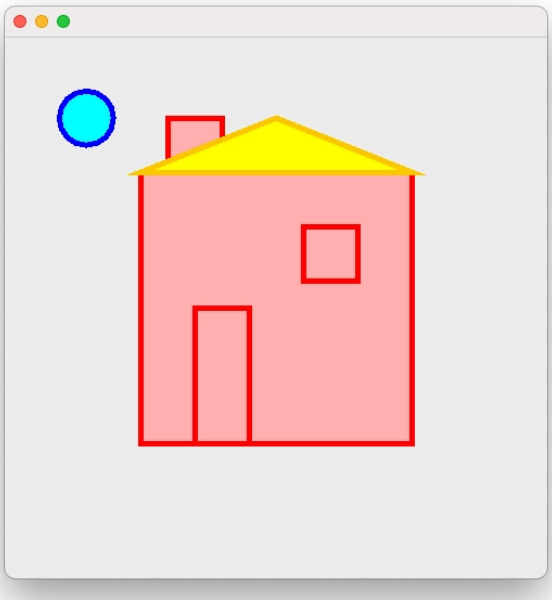
  This method draws a triangle using the array of x and y coordinates.

  Do read through the draw methods found in other classes to show you how to set it up properly.

  Furthermore, to draw or fill a triangle you have to use one of the following methods respectively:

```
obj.drawPolygon(xCoordArray, yCoordArray, numberOfPoints); obj.fillPolygon(xC
oordArray, yCoordArray, numberOfPoints);
```

Tester:



Once you've finished with the `Triangle` class, uncomment line 31 in the `MyPanel` class to draw a roof. Your goal now is to draw a house that looks like this!

All you have to add is a chimney, a window, and a sun!

Read below to understand how coordinates work in `JFrame`.

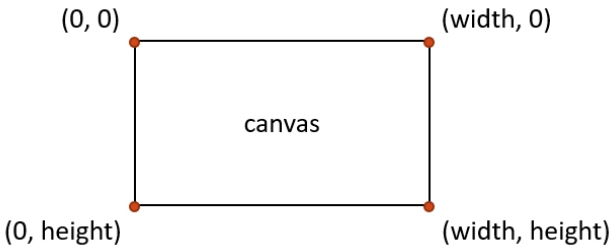
Unfortunately, it seems like all similar shapes stay the same color despite attempting to change their color. For example, all rectangles are printing out as red. Also all circles are printing out as blue!

This bug must be fixed.

To help draw your shapes read the following:

In computer science, we calculate coordinates a little different than we do in math class. Consider that our canvas is 500x500 pixels long.

The **top-left** corner is $(0, 0)$. Moving the x-coordinate to the right increases the x-coordinate as expected. However, moving the y-coordinate down increases the y-coordinate. So the **bottom-right** corner of our canvas is $(500, 500)$.



💪 **Stretch Challenge: Submit your code before continuing!**

It would make drawing much easier if we could initialized the color upon instantiation instead of setting them after the fact.

Add two more constructors in each class that can accept up to 3 more parameters for the `fillColor`, `strokeColor`, and `strokeWidth`.

```
Rectangle(int xCoord, int yCoord, Color fillColor); Rectangle(int xCoor
d, int yCoord, Color fillColor, Color strokeColor, int strokeWidth);
```

If the colors are set to `null`, the draw methods will execute but nothing will be displayed due to the absence of color. Therefore, `fillColor` and `strokeColor` can be set to `null` for a colorless result, and `strokeWidth` can be set to `0` to remove the outline.

Initializing a few colors could act as a color palette for your drawing!

By initializing a few `Color` objects in `MyPanel` using RGB values, you can simplify the coloring process:

```
Color plum = new Color(255, 87, 51); Color redOrange = new Color(248, 7
2, 28); Rectangle r = new Rectangle(20, 20, 30, 100, plum); Circle c =
new Circle(100, 100, 40, plum, redOrange, 3);
```

When completed, create a masterpiece!

Observe the masterpiece below: