Lab 01: Sideways Robot Face

Assigned: Thursday, ~~January 31~~ February 7, 2019
Due: Wednesday, ~~February 6~~ February 13 at 11:30pm

1 Objective

In this assignment you will learn how to use the AutoLab submission tool and practice writing Java programs made up of multiple methods.

2 Working From Home

You may find that you did not have enough time to complete this assignment during our scheduled session. The easiest way to continue working on is just to come back to the computer lab when it is open – the schedule is posted outside the door.

But if you would like to be able to work on your projects on your personal computer, you will need to install Eclipse on your personal computer. Fortunately, Eclipse is a free program (in every sense of the word), and can run on pretty much any kind of computer. You can download it from `https://www.eclipse.org/downloads/`. The link that you want to follow is `Download 64 bit` under `Get Eclipse IDE 2018-12` (or a later date, if a new version is released between when I write this document and you read it).

The second task is to get the file that you have been working on in the lab onto your personal computer, then (if desired) get the updated version back from your personal computer to the lab. There are many ways you could choose to do this:

- Just attach your `RobotFaceDrawer.java` file to an email to yourself, then download it from your email when you are going to work on it again. If you choose this route, you will need to create the workspace and project again on your own computer, then drop the `RobotFaceDrawer.java` file into the `src` file of that project. If there is already a version there, you may need to tell Eclipse to refresh before it will notice that there is a new version. This is by far the easiest option to set up.
- Install CyberDuck from `https://cyberduck.io/`. This program will allow you to copy files across the Internet between your computer and the lab computers. You will need to know the hostname of the computer that you want to connect to, and you can choose one of them from the list at `http://www.millersville.edu/computerscience/computer-labs/linux-lab.php`. You should be able to copy the entire workspace back and forth.
- Install MountainDuck from `https://mountainduck.io/`. This is built on top of the CyberDuck program mentioned earlier, but should make it possible to treat the files on the lab computers like another hard drive on your local computer, meaning that you won't have to copy files back and forth but can work on the lab versions directly as if they were on your computer. I haven't had a chance to try this, but it seems like the nicest option to me.

- Use DropBox or some other storage service to keep your files in the cloud.

I cannot provide technical support for any of these options, but the lab monitor in Caputo 131 should be able to.

3 Setup

Like in the previous assignment, you will need to start Eclipse and choose the workspace that you created last time. (It should now be the default.) When Eclipse has started and you have closed the Welcome screen (if necessary), you should see that your `Lab00` project is still there. Create a second project, named `Lab01`, in the same way that we did last week. As you did then, do not create a `module-info` file.

Unlike Lab 00, you will not submit your work through D2L. Instead, we will be using another website named AutoLab. Start by visiting `autolab.millersville.edu` with the Chromium web browser. Because this is the first time that you are using AutoLab, you will need to click the `Forgot Your Password?` link and enter your Millersville email address. An email will be sent to you with instructions for creating a password that you will use to login to AutoLab for all future assignments.

Once you are logged in, you should see our course (`161-s19`). Click on it, and then on the link for this assignment (`Lab 01: Sideways Robot Face`). Leave this window open, because you will come back here to submit your work.

Inside your `Lab01` project in Eclipse, create a new class named `RobotFaceDrawer`.

4 Assignment

Complete each of the steps below:

1. (10 points) Add a method named `drawTriangleUp` to the class. Whenever the method is called, it should draw the following picture:

```
     *
    / \
   /   \
  /     \
 /       \
/         \
```

   It should not print any spaces at the end of lines, and the bottom of the triangle should start at the beginning of the line. Go to the next line after printing the last part of the triangle.

   It would be wise to test each method after you are finished writing it. To do so, create a `main` method and (temporarily) have it call the method that you want to test. Then run the program.

In fact, while it is a very good idea to run your program after every method that you write to look at the output, you can do more than that. Let's submit your file to Autolab. Back in your web browser, you should see a `Submit File` button. This will open a window where you can find your file. Remember to look for `RobotFaceDrawer.java` in the `src` directory, in the `RobotFaceDrawer` project directory, in the workspace that you created.

After you have uploaded, you should be sent to a page that shows a table, where each row will be one of your submissions, and each column will contain your score for one of the questions (called "problems" by Autolab) on this assignment. Initially, you will just see dashes for every problem. Wait about 15 seconds, and then refresh the page. Now some of the dashes (the ones for questions the autograder can handle) will be replaced by numbers. Some problems will still have dashes, because those problems cannot be automatically graded.

You can click on any of the numbers to see a report about what was correct or incorrect in your program. Some of the lines at the beginning and end of this report will not make sense to you, which is OK. Look for the sections for each problem. If you made a mistake in the `drawTriangleUp` method, you will probably see a section of the report like this:

```
################################################################################
# Problem 01 drawTriangleUp: 0/10

output(Q01OutcomeTest): java.lang.AssertionError: The drawTriangleUp method did
not draw the correct number of lines. expected:<6> but was:<5>
```

This is saying that I am getting 0 points for problem 01, because while it was able to run the method that I wrote, the method did not produce the correct output. You can ignore the beginning of the error message and focus on the end, which says that the method should have printed 6 lines but instead only printed 5.

Here is another example of an error message that you might see:

```
######################################################################
# Problem 01 drawTriangleUp: 0/10

output(Q01OutcomeTest): org.junit.ComparisonFailure: Line 3 printed by
drawTriangleUp does not exactly match specification. expected:<   /[   ]\>
but was:<   /[abc]\>
```

This time the problem is that on the third line of output I printed `abc` where I should have printed three spaces. The square brackets highlight where the output was different from what was expected.

There are many other kinds of error messages you might see depending on the different types of mistakes you might make. You will always only see one error message, even if there is more than one problem with the method. If you are having trouble interpreting a message, please ask for help!

If there were any errors in problem 01, you should fix them and re-submit your file. (Press the back button twice to have another opportunity to submit a file.) Eventually, you want to see a section of the report like this:

```
################################################################################
# Problem 01 drawTriangleUp: 10/10
```

Of course, there will still be errors with other problems, because you have not yet started working on them. Near the bottom you can see how close you are to getting the maximum possible autograded points:

```
################################################################################
# Total autograded points: 10/80
```

You should ignore any text below that part of the report. Re-submit your file as many times as you would like, ideally after each method that you finish, until you have earned the maximum possible points.

2. (10 points) Add a method named `drawTriangleDown` to the class. Whenever the method is called, it should draw the following picture:

```
\         /
 \       /
  \     /
   \   /
    \ /
     *
```

It would be wise to test this yourself and submit it to AutoLab to ensure that it works correctly before moving on to other steps.

3. (10 points) Add a method named drawDiamond to the class. Whenever the method is called, it should draw the following picture:

```
     *
    / \
   /   \
  /     \
 /       \
/         \
\         /
 \       /
  \     /
   \   /
    \ /
     *
```

The drawDiamond method should not contain any direct printing instructions. Instead, it should use the two methods you already wrote. To draw a diamond, first draw an up triangle, then draw a down triangle.

(AutoLab is not smart enough to determine whether or not you have implemented this method correctly – as long as the output is correct, it will give you full credit. But I will check how you wrote this method and others when grading.)

4. (10 points) Add a method named drawCheckedLineStartsFilled to the class. Whenever the method is called, it should print a # sign then a space, seven times, like this:

```
# # # # # # #
```

5. (10 points) Add a method named drawCheckedLineStartsEmpty to the class. Whenever the method is called, it should print a single space, then seven copies of a # sign followed by a space, like this:

```
 # # # # # # #
```

Find a way to implement this method by using your existing drawCheckedLineStartsFilled method.

6. (10 points) Add a method named `drawCheckedPattern` to the class. Whenever the method is called, it should draw the following picture:

```
# # # # # # #
 # # # # # # #
# # # # # # #
 # # # # # # #
# # # # # # #
 # # # # # # #
# # # # # # #
 # # # # # # #
# # # # # # #
```

   This method should not contain any direct printing instructions. Instead, it should use two of the methods that you already wrote (multiple times).

7. (20 points) Clear out any statements that you had put in the `main` method temporarily for testing. Then make the `main` method draw the complete picture that can be found on the next page. When looking at the output of your program, you will need to make the console window much bigger than it normally is.

   As much as possible, the `main` method should use the other methods that you wrote. Note that there is a blank line on both sides of the "nose".

```
     *
    / \
   /   \
  /     \
 /       \
/         \
     *
    / \
   /   \
  /     \
 /       \
/         \
\         /
 \       /
  \     /
   \   /
    \ /
     *

# # # # # # #
 # # # # # # #
# # # # # # #
 # # # # # # #
# # # # # # #
 # # # # # # #
# # # # # # #
 # # # # # # #
# # # # # # #

     *
    / \
   /   \
  /     \
 /       \
/         \
\         /
 \       /
  \     /
   \   /
    \ /
     *
\         /
 \       /
  \     /
   \   /
    \ /
     *
```

8. (10 points) [not autograded] Make sure that you have written Javadoc comments for the class and all methods, and that the comment for the class includes you as the author.

9. (10 points) [not autograded] Imagine that we wanted to change the robot's nose to be twice as tall, so that each checked line would have 14 `#` signs. How many lines of code would you need to change to accomplish this? If the answer is more than 1, you should reconsider how you implemented `drawCheckedLineStartsEmpty`, `drawCheckedPattern`, or `Main`. Suppose that we did not create any extra methods, but instead wrote all of our print statements in the `main` method. Then how many lines would you need to change?

   Add one or more single-line comments inside the `drawCheckedLineStartsFilled` method explaining the benefits that we are getting from having written that method.

5 Submitting Your Work

Make sure that you have submitted your final version to AutoLab, and that (unless you are satisfied with a lower score) you have 80/80 autograded points. After the due date I will grade the other 20 points, while also confirming that you implemented the methods correctly.

You do <u>not</u> need to submit your work through D2L – I will look at whatever your last AutoLab submission is.