Lab 7: Pseudocode

This lab will give you practice writing your own pseudocode. Pseudocode is especially beneficial when writing complex algorithms. You will learn how pseudocode can make writing real code that much easier. If you need a reminder on how to write good pseudocode, feel free to look back at the reading.

Goal: Practice writing pseudocode and understand how pseudocode translates to real code.

Drawing Turtles

Remember the turtle demo from lecture? You are going to create your own! Your turtle can do the following things:

- move forward a number of steps (each step is one pixel long)
- turn left 90 degrees
- turn right 90 degrees
- put its pen down, which starts drawing the turtle's path
- lift its pen up, which stops drawing the turtle's path

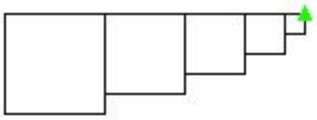
Start by coming up with pseudocode together as partners.

Pseudocode

You will be writing pseudocode for an algorithm that directs the turtle to draw a chain of squares decreasing in size by 10 pixels.

The width of the turtle's pen is one pixel. The turtle starts off facing upwards with the pen up.

Example: The chain should look like this for a starting length of 50:



- Fill in the following pseudocode method:

```
method drawSquares(turtle, sideLen){
     // write pseudocode here!
}
```

Note: `sideLen` is the starting side length of the squares.

Follow-up Note: The side length of a square should never be less than 0, right?

Checkpoint 1: Show your pseudocode to a TA before moving on. Now choose one partner to be the navigator and one to be the driver. Remember you will switch roles at the next checkpoint.

Now that you have finished writing your pseudocode, put your skills to the test and go code your algorithm. If you have written your pseudocode well, it should be a trivial step to translate it line by line into Java. But first...

Running the App

- Run `cs0150_install lab7`. This will install stencil code in `/home/<yourlogin>/course/cs0150/lab7/`.
- Open up eclipse, right-click on `App.java`, and choose “Run as… >> Java Application”
- What’s this?! You should notice that your program will not run…
- Continue reading to see how to fix this.

Oh no! It appears that Doofenshmirtz has locked the Turtle Drawer, which is preventing you from being able to execute your code. If you inspect the code in `MyTurtleDrawer.java` you’ll notice that the program can be unlocked by successfully passing in a password (that is dependent on your username) to the `checkPassword(String username, String password)` method. Luckily Terry the Turtle, another agent in the O.W.C.A, is on the case! Due to the cunning of Doofenshmirtz, our human eye is unable to see the inner workings of this method. So this looks like a job for… print statements! Help Terry unlock the password so he can draw his turtle wife and turtle children and finally be reunited with his long lost family.

Operation Turtle

`checkPassword(String username, String password)`, a method defined in `AbstractTurtleDrawer.java`, works by using your login to generate a password. You will have to fill in your login on line 14 of `MyTurtleDrawer.java`. It then stores this in the variable `correctPassword` and checks whether your inputted password equals `correctPassword`.

To unlock `MyTurtleDrawer` you have to figure out what this password is! You *could* try to figure out how `checkPassword(...)` generates the password, but *this is pretty hard*.

Maybe you can just figure out what password checkPassword is expecting. Use the Eclipse debugger or print statements to find the value of `correctPassword`!

Once you get the message “Yes! You got the right password!” you’ve completed Operation Turtle and can move on to reuniting Terry with his family..

Checkpoint 2: When you have finished this, switch roles with your partner: the navigator should now be the driver, and vice versa.

From Pseudocode to Actual Code

Now let’s take the “pseudo” out of pseudocode! You will be implementing your pseudocode in `MyTurtleDrawer` to make your turtle minions draw a chain of squares.

- Translate your pseudocode into real code in the `drawSquares(Turtle turtle, int sideLen)` method in the `MyTurtleDrawer.java` file. Look at the comments in the file to see what methods of Turtle you can call.

  Note: You do not need to modify the other files, `App.java` and `AbstractTurtleDrawer.java`.
  Note: The Turtle draws from a random point on the screen.

- Run your program. If your turtle does not draw a chain of squares as expected, go back and hand simulate your pseudocode again! What is going wrong?

Checkpoint 3: Show your program to a TA to get checked off for today’s lab!

Congratulations on finishing the lab. If you finish early, feel free to play around with the drawing turtles and make some cool designs (maybe try a spiral or a filled in square); this is a great way to practice writing loops!