Layla Oesper

This assignment is due by **10pm on Friday September 15** and is worth 16 points.

1 Goals

This goal of this assignment is to help you some practice writing your first Java programs. In particular, you will get more practice with retrieving and using input from a user, loops, printing output to the screen, and using some of the provided Java classes (`Scanner, StringBuilder, StringJoiner`). Getting comfortable with the basics of Java will set you up well for future assignments in this class.

2 Setup

There are no outside files that you will need for this assignment. However, this is a good time to get yourself setup for future assignments.

2.1 Choosing an Editor

One big part of that will be deciding what editor or program you will use when writing code. There are many good options to choose from. In CS 111 you likely used either Brackets or TextWrangler (both of which are installed on the lab machines). Either of those are good options to continue using this course. To see all the editors installed on the lab machines, go to `Applications/CarletonApps`. If you want to try something different, here are a few more options.

1. **Emacs** - This is an incredibly powerful editor that has been around forever (when I first started learning programming, this is what I was taught to use). Beware, there is some amount of start-up cost when you first start using Emacs, but here is a tour that might be helpful. Lastly, this is installed on the lab machines.
2. **Aquamacs** - This is a Mac specific version of Emacs that is also installed on the lab machines.
3. **Vim or Vi**- This is another incredibly powerful editor that has been around forever (I use a version of it all the time now). It has the advantage that essentially any UNIX system (Mac or Linux) always has it installed, and it is very lightweight and starts up quickly. You can start it in a terminal window in any Mac or Linux system by typing `vim` or `vi` at a terminal prompt.

A different type of option is using what's called an integrated development environment (IDE). These are programs that are designed to help people code - they often make it easier to organize large projects and have some features to make coding easier (those features vary based on the IDE, but may include a debugger to step through your code line by line, code completion to help you remember the names of methods, or compiling in the background and showing you the line an error occurred on). However, they can also add extra overhead in terms of having to learn how to use the IDE and learn how to use Java. For that reason, I'm not going to talk about IDEs in class. If you take more CS courses such as software design, you'll learn lots about IDEs. If you want to explore IDEs on your own, you're welcome to. There are a number of options out there, but a few that might be good to explore are BlueJay, Eclipse, IntelliJ and jGRASP.

2.2 Installing Java

Java is installed in the computer labs on campus (both in the CMC and elsewhere). But if you have your own machine and would like to use it for this class, you are welcome to install Java on your own computer as well using these instructions on the CS department website. Note that the only officially supported environment is the lab computers, but Mike Tie (`mtie@carleton.edu`, CMC 305) can help you troubleshoot if you run into problems installing Java on your home computer; I'm also happy to try to help you during office hours if you run into problems.

3 Your Assignment

This a solo assignment - you should not work with a partner. The reason for this is to make sure that everyone feels comfortable writing a short program on their own. All of your code files should include your name at the top (in comments). You can ask questions on Piazza or talk to me, the lab assistants, or the prefect if you're having trouble. The appendix of your book is also very helpful for Java syntax. Moodle has a link to the Javadocs for Java - these are incredibly useful as a reference! (I often refer back to Javadocs when writing code - that's what they're there for!)

For this assignment you'll be creating a small Welcome to Data Structures App. You should put all code for this assignment in a file called `Welcome.java`. All of the functionality of your App will be contained in the `main` method. In particular, your program should complete the following tasks in the specified order:

1. Print to the screen "Welcome to CS 201: Data Structures!"
2. Ask the user for their name and use the `Scanner` class to retrieve the user input.
3. Ask the user to enter an integer $(\ldots, -2, -1, 0, 1, 2, \ldots)$ and use the `Scanner` class to retrieve the user input.
4. Print to the screen "Welcome [name]" where [name] is the name the user typed in.
5. Print to the screen "Your name backwards is [backwards]" where [backwards] is the name the user typed in displayed backwards. You may find the `StringBuilder` class from the reading useful here.
6. If the the number n entered by the user was positive, print off a triangle of numbers like the following example where $n = 3$. (I suggest using the `StringJoiner` class from the reading.)

   1
   2,2
   3,3,3
7. Otherwise, if the the number n entered by the user was negative, print off a triangle of numbers like the following example where $n = -4$. (I suggest using the `StringJoiner` class from the reading.)

   -1
   -2,-2
   -3,-3,-3
   -4,-4,-4,-4
8. Otherwise, the number was 0, and just print "Cannot print a triangle of height 0."

For example, here are a few possible runs of the program and what the output might look like:

```
$ javac Welcome.java
$ java Welcome
Welcome to CS 201: Data Structures!
What is your name: Layla
Enter an integer: 5

Welcome Layla
Your name backwards is alyaL
1
2,2
3,3,3
4,4,4,4
5,5,5,5,5
```

```
$ javac Welcome.java
$ java Welcome
Welcome to CS 201: Data Structures!
What is your name: Sabinian II
Enter an integer: 0

Welcome Sabinian II
Your name backwards is II nainibaS
Cannot print a triangle of height 0
```

Last Required Challenge: Modify your code so that if a user does not enter a valid integer, the program prints out the following warning "You must enter an integer" and then quits.

Optional Challenge: Modify your code so that instead of quitting when a user does not enter an integer when prompted, the program just keeps re-prompting them to enter an integer until valid input is obtained.

3.1 Final Hints

- Make sure to fully test your code. Try out positive and negative numbers as well as integers and real numbers. The documentation for the Scanner class may be very helpful to look at as you try to figure out how to do all of this.
- Look back at examples from class or your book if you get stuck. It's totally okay to experiment and try out methods to see what will happen - that is all part of the learning process.
- Make sure to save your work regularly as you go along!
- If you get stuck, don't hesitate to ask for help from a lab assistant, prefect or me!

4 Submission and Grading

You'll submit `Welcome.java` to Moodle as a zipped file. We'll be using zip for all submissions this term, so it's a good habit to get into now. In particular, put your `Welcome.java` file into a directory named `[your_last_name]HW2`, zip this directory, upload it to Moodle. For example, my directory would be named `OesperHW2` and the resulting file would be `OesperHW2.zip`.

4.1 Hints for zipping a file or directory

- **From the command line:** You can zip files from the command line using the command: `zip -r [Name for zip file] [Name or directory of file to add to zip]`.

  For example, `zip -r OesperHW2.zip OesperHW2` will create the zip file `OesperHW2.zip` that contains the directory `OesperHW2` and all files contained inside that directory. The `-r` means "recursive" and is what tells `zip` to include all files in the directory in the zip file.

- **On a Mac:** Select the folder you want to zip. Then right click (command and click, or two finger click on many trackpads). You'll see a little menu pop up. Choose the option that starts with "Compress". For instance, if I had selected `OesperHW2`, it would say "Compress `OesperHW2`". This will produce a new zip file `OesperHW2.zip`.

- **On a PC:** Just as for a mac, selection the files/folders and right click. Select "Send to" in the menu, and then click "Compressed (zipped) folder".

4.2 Grading

This assignment is worth 16 points. Below is a partial list of the things that we'll look for when evaluating your work.

- Does the program compile? It is essential that you check whether your submitted work compiles. If you submit a program that doesn't compile, you will automatically receive at least a 25% deduction for the assignment, even if the problem was relatively minor.
- Does the program run correctly on test input?
- Does the program use `Scanner, StringJoiner, StringBuilder` correctly?
- Does the program handle positive/negative numbers and non-integer values correctly?
- Are comments used throughout the program to describe logic decisions?
- Is the program named `Welcome.java`? Please make sure to always use any names specified in an assignment.
- Does the program use Javadoc style comments at the top that include your name as the author of the code?
- Does the program use correct indentation?
- Does the program follow the Style guidelines outlined on the Moodle page?

Start early, ask lots of questions, and have fun! Layla, the lab assistants, and the prefect are all here to help you succeed - don't hesitate to ask for help if you're struggling!