CS112 – Spring 2024
Lab02
Instructor: Paul Haskell

TEXTBOOK READINGS:

- This lecture: sections 1.4-1.6, 2.1-2.3
- Next lecture: 2.4-2.5, 5.1-5.4
- Going forward, reading assignments will be posted only on the **CourseInfo README** page!

INTRODUCTION:

In this lab you will start working with Java! First you will download and install software to let you edit Java source code files. Then you will download and install a particular version of the Java Development Kit.

After that, you will copy, modify, and run some Java source code files I have provided. After your modifications and fix-ups, you will upload your Java source code files to your GitHub repository.

LAB02 Part 1

FIRST STEP – INSTALL SOFTWARE FOR EDITING Java SOURCE CODE

If you already have a text editor that you like, feel free to use it. If not, please install one of these. (On your own time, you can install several, try them, and see which you prefer.)

- Sublime – https://www.sublimetext.com/download ($80 but can "evaluate" forever for free)
- Notepad++ - (free)
  - https://notepad-plus-plus.org/ (for Windows)
  - Install PlayOnMac (https://playonmac.com) and then use that to install Notepad++
- TextMate – Mac only, gets good reviews
  - https://macromates.com
- VSCodium – https://sourceforge.net/projects/vscodium.mirror/ (free version of Microsoft's paid VSCode software)

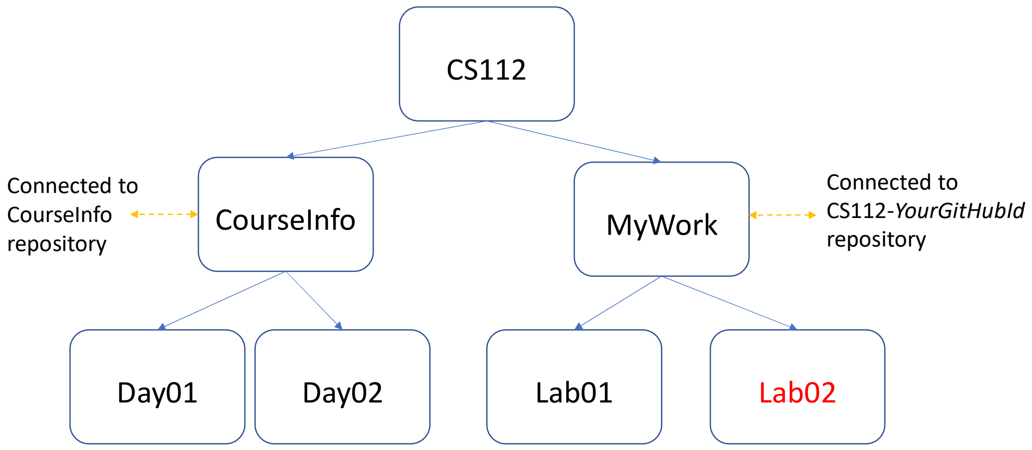
NEXT STEP – INSTALL JAVA DEVELOPMENT KIT (JDK)

- Browse to https://www.oracle.com/java/technologies/downloads/#java17
- Select the proper JDK version 17 download the "installer" for your computer (Linux, MacOS, or Windows) and run the installer.
- After the install, in a terminal window, type: `java --version`
- You should get a message saying that the java version is something like "17.0.3.1". If so, then type: `javac --version`
- Hopefully you again get a few lines that include a version number.
- If you do not, but rather get a message saying that the "java" command is not found and/or the "javac" command is not found, then you must add the proper directory to your $PATH variable. (This variable tells your computer where to find programs to run.)

- On Windows:
  - Open 'Settings' and then search for 'Advanced System Settings' and then click the 'Environment Variables' button.
  - In the 'System variables' section, click on 'Path' and then 'Edit'
  - Wherever your Java installer installed Java (probably C:\Program Files\Java\jdk….\bin), add that directory path to your Path variable.
  - Click 'OK' however often is necessary, and then <u>create a new terminal window</u> and try again
- On Mac:
  - Find where Java was installed (possibly /usr/lib/jvm/jdk-…/bin)
  - Edit the appropriate setup file in your home directory (probably **.bashrc**), find where your PATH variable is set, add the Java bin directory to your PATH, and save the file.
  - From your terminal window, type: `source .bashrc`
  - Try to run `java` and `javac` again. You should get output with the version number.

NEXT STEP - SET UP A DIRECTORY ON YOUR PERSONAL COMPUTER FOR THIS LAB

- cd to your CS112 directory. cd to your **MyWork** directory.
- Create a subdirectory called **Lab02**.


CS112
Connected to CourseInfo repository
CourseInfo
MyWork
Connected to CS112-*YourGitHubId* repository
Day01
Day02
Lab01
Lab02

NEXT – Modify, compile, and run your first Java program

- First, you must get the latest up-to-date version of the "CourseInfo" repository. In a terminal window change directories to your **CS112/CourseInfo** directory. Then type `git pull`
- Copy the **helloworld.java** file from **CS112/CourseInfo/Day02** to your **CS112/MyWork/Lab02** directory. You can do this with the terminal window, or using Finder or File Explorer.
- Edit the file to put your GitHub userid where it belongs and save the file. Please <u>do not change anything else</u> in the file, because we will use an automatic tester to verify the result.
- Compile the file. From a terminal window, go to the **Lab02** directory and type:
  `javac helloworld.java`
- Run the resulting program: type: `java helloworld`

- Did you get the result you want? If so, upload your modified **helloworld.java** to your repository: in a terminal window, from the **Lab02** directory, type:
  `git add helloworld.java`
- Type: `git commit -m "Adding helloworld.java"`
- Type: `git push`

Part 2 – Correcting Broken Java Programs

Java has many similarities to Python, but also some key differences. In this lab, we will explore two areas where Java is different from Python: class-based syntax and typed variables. You will do this by taking some Java programs that are not legal Java and fixing their errors.

The programs listed below are in the ClassInfo repository in the **Day02** directory. Copy the programs to your **MyWork/Lab02** directory and try compiling and running them.[1] Look at the errors you get and fix them, so the output is correct. Then push the programs to your GitHub repository: in your **MyWork/Lab02** directory, type

```
git add *.java

git commit -m "Put your comments here"

git push
```

Program01

- Debug so the two text strings print successfully when the program is compiled and run

Program02

- Fix the code to include the tip. The tip only applies to the pizza price, not the tax!

Program03

- Debug so the program compiles. Fix the variable formulas so that each variable has the correct value. Do not simply print '365' but rather fix the existing calculation.

Program04

- Debug the program so it runs. Don't change any of the number values, just the variable types.

Program05

- Debug the variable usage so the correct cone volume is reported—be sure to check yourself! You can add (and later remove) additional `System.out.println()` commands to see what is going on. (I will verify your answer to about 1% accuracy.)

Program06

- Fix up the numerous bugs so the program compiles. For negative inputs, the `mySquareRoot()` method should return the negative of the square root of the absolute value of the input.

[1] You must run **javac ProgramName.java** and then **java ProgramName** .
Don't just run **java ProgramName.java** because you may get incorrect results.

Program07

- There are several fixes needed by this program. Do change the variable types if needed. Report the correct number of seconds per year and the correct approximation. The approximation should be pretty close to the correct value.

Program08

- Fix the program to print out the correct value for the total number of pizza choices on the menu. Don't forget about footnote #1 on the previous page!

Program09

- A few fixes are needed.
- This program shows something tricky about Java: we can initialize a char from an int constant but not an int variable.

Program10

- Debug this program so it runs correctly. What does it print?

CONCLUSION

The deadline for completion of Lab02 is Sunday January 28 at 11:59pm.

After this lab, you should be able to edit, compile, run, and upload Java programs. Please do experiment on your own to get more familiar. In the next few weeks, the programs you write will quickly get more complicated.

Task	Score, points
Created properly named subdirectory	1
helloworld.java compiles successfully	2
helloworld.java runs and generates first output line correctly	2
helloworld.java generates second output line correctly	2
helloworld.java outputs correct GitHub UserID	2
For each of Program01 – Program10	2 points if it compiles without significant changes to what it does. 2 points if it generates correct output