INTRODUCTION TO COMPUTER SCIENCE I
LAB 1
A First Program

1 Getting started

The following steps will get you started with this lab. You will be introduced to the programs and tools that you will need to complete the lab and project assignment. Feel free to ask questions about the details of getting rolling here.

1. **Login to your workstation:** Our lab is full of Windows desktop computers. These are run by our Information Technology department, and you must begin by logging into them using your college username and password.[1] These workstations won't do much themselves: they will primarily be used to connect to different systems on which the real work will happen.

2. **Login to a server:** The computer systems that we will use for our projects are `romulus.amherst.edu` or `remus.amherst.edu`, (henceforth, `remus/romulus`), which are UNIX (Linux) systems. To use these systems, you must login to them from your workstation using *Remote Desktop*, software that allows you to connect graphically to these servers. To do so, follow Remote Desktop Connection instructions that describe not only how to use this software on the Windows machines in Seeley Mudd 014, but also how to install (if needed) and use this software on your own computer, whether Windows, Mac, or Linux.

   Once you have logged into `remus/romulus`, you will see the graphical interface for that server, which looks very much like the desktop that your own computer presents. Right-click anywhere in the empty part of the desktop (not on an icon), and a menu will pop up. In that menu, select *Open in Terminal*. You will then see a *terminal window* within which there is a *shell*—a prompt at which you can type commands to the system. The shell is the place from which you will launch the program that allow you to edit your source code, to compile that source code, and to execute (run) your programs.

3. **Make a directory:** When you first login and open a terminal window, you will be working in your `Desktop` directory. However, we want to work in your *home directory*—the UNIX equivalent of your *My Documents* folder. So first, change into your home directory, like so:

   ```
   $ cd
   ```

   Now in your home directory (represented by the tilde character at the prompt), you should make a *subdirectory* (a folder) for your work for this lab by using the `mkdir` (***m**ake **dir**ectory*) command (shown below). Once you do that, you can `cd` (***c**hange **d**irectory*) into that new folder. Using those commands looks like the following, where the dollar sign (`$`) is the *prompt*—what the shell prints before stopping to wait for you to type a command:

[1] It is likely that you've already logged into a Windows workstation if you happen to be reading this document. Go with it.

```
$ mkdir lab-1
$ cd lab-1
```

4. **Get some starting source code:** Use the following command to obtain a sample Java source code file:[2]

   ```
   $ wget -nv --trust-server-names goo.gl/USnSwQ
   ```

   This command will generate some output to show you what it's doing. To ensure that you have copied the file into your `lab-1` subdirectory, use the `ls` (***list** directory*) command to list the files in the current directory, noting that the character following the dash (`-`) is a lowercase letter `L`, and **not** the numeral `1`. The `-l` part of the command indicates that you want to list the directory using the *long format*:

   ```
   $ ls -l
   ```

   You should see an output that looks something like this:

   ```
   total 4
   -rw-r--r-- 1 sfkaplan sfkaplan 187 Aug 9 22:18 Messages.java
   ```

You have now set yourself up to work with some Java source code. Move onto the next section.

2 Your assignment

Perform the following steps in order to see, use, and change your code:

1. **Examine the source code:** Run *Emacs*, a programming text editor, to examine the `Messages.java` file. In the following command, be sure to include the trailing ampersand (`&`), causing the text editor to run in the *background*—that is, to run while allowing you to enter more commands:

   ```
   $ emacs Messages.java &
   ```

   A new window will open, showing you a handful of lines of Java code (along with *code comments*—lines that begin with two forward slashes (`//`)—that provide some human documentation and commentary. This code shows a program named `Messages` that will start on the complex line beginning `public static void main (String[] args)`, carry out two commands to print messages, and then end. For now, examine the code, but leave it as-is.

   ASIDE: *Emacs* is a complex program that can do a great deal. To learn more about how to use it, you should read this documentation/tutorial on using Emacs. Also note that you are **welcome to use a different editor or an IDE**; so long as your submitted code runs correctly, I don't care what tools you used to write it.[3]

[2] If you are working on your own computer, and don't use the command-line like this on it, then simply copy-and-paste the link into your browser to download the code.

[3] *Eclipse* and *IntelliJ* are common. If you don't know what an *IDE* is, then don't worry about it, not even a little.

2. **Compile the code:** The source code must be translated it into a form that the computer can execute. Leaving your *Emacs* window open, click over to your terminal window again. In it, use the following command to *compile*—that is, translate into machine code—your source code:

   ```
   $ javac Messages.java
   ```

   In this case, **no news is good news**. That is, if the computer simply presents the shell prompt to you after you issue this command, then **the compilation succeeded**. The compiler—the `javac` program—will print messages into your terminal window only if it was unable to translate your program. Given that you've changed nothing from the original code, it should compile without error.

3. **Execute your program:** Once you have successfully compiled your program, it is time to run it and see what happens. Go to your shell window and issue this command:

   ```
   $ java Messages
   ```

   The messages that you saw in the code should be printed into your terminal window.

4. **Change something:** Alter your program slightly, and then test that your alteration did what you expected. Specifically:

   (a) **Edit:** Go back to your *Emacs* window. **Add your own print statement or two**, using the existing print statements as templates. Be sure to **save** your changed source code.

   (b) **Compile:** Issue the `javac` command to your shell, just like you did above. If there are error messages, you will need to return to *Emacs*, fix whatever is incorrect, save the changed code, and try to compile with `javac` again. Repeat until your code compiles with no error messages.

   (c) **Execute:** Issue the `java` command to your shell, just like you did above. Your program should run, this time printing the additional message that you added into the code.

Ta-da! You've just done some programming. You have also begun to use the tools on which your work will depend in this course: remote desktop, remus/romulus, the shell, the *Emacs* text editor, the `javac` compiler, and the `java` executor.

3 How to submit your work

Submit your modified `Messages.java` file. You may use either of the following two methods to use the CS submission system:

- **Web-based:** Visit the submission system web page. If you did your work on your own computer, open this link in your computer's browser; if you did your work on `remus/romulus`, then open this link in a browser opened within *remote desktop*.
- **Command-line based:** Use the `cssubmit` command at your shell prompt, as below, following the prompts it presents about which class and assignment you are submitting:[4]

```
$ cssubmit Messages.java
```

This assignment is due on Thursday, Sep-14, 11:59 pm.

[4]This command works only while connected to `remus/romulus`.