Lab 1 Setup: Setting Up Your Computer

We encourage you to complete as much of this setup as you can on your own before coming to lab. If you get stuck, come to office hours or lab to get help.

A. Installing a Text Editor (optional)

If you don't already have a favorite text editor, we recommend installing one.

The three most popular GUI text editors these days seem to be:

1. Sublime Text (free but nags you until you pay): https://www.sublimetext.com/
2. Atom (free): https://atom.io/
3. Visual Studio Code (free): https://code.visualstudio.com/

See this text editor review for a more thorough look at these and other text editors.

The choice isn't very important, as we will only be using a text editor a few times throughout the course. Most of the time we'll be using something else called an IDE.

You do not need to pick one of the three options above. You're welcome to use a different text editor entirely (built-in text editors, vim, emacs, etc).

B. Configure Your Computer

Depending on your operating system, there are a few things we need to do to set your computer up for 61B.

The precise steps to take depend on your operating system.

- Windows instructions [Video Walkthrough]
- macOS instructions
- Linux instructions

Move on to the next section only once you've completed the instructions above for your operating system. Advanced users on Windows may also use the new Bash for Windows feature, but we will not be providing official directions. Note that if you use Bash for Windows, you'll need to install Java twice (once inside Bash for Windows, and once inside Windows itself, following the directions above).

C. Learn to Use the Terminal

If you already know how to open and use a terminal, skip this section.

The terminal is an application that allows you to run all sorts of programs, as well as manipulate files in your own computer. It is a powerful but also dangerous tool, so please be careful with using some of these commands. On Unix-like operating systems, the Terminal application will provide you with everything that you need. On macOS, for example, you can use Spotlight to search for the Terminal application.

The lab computers run the Linux operating system. As such, you can use terminal commands to make changes to your directory and files. Here are some important ones that you may find useful in this course:

- `cd`: change your working directory

  ```
  cd hw
  ```

  This command will change your directory to `hw`.

- `pwd`: present working directory

  ```
  pwd
  ```

  This command will tell you the full absolute path for the current directory you are in if you are not sure where you are.

- `.`: means your current directory

  ```
  cd .
  ```

  This command will change your directory to the current directory (aka. do nothing).

- `..`: means one parent directory above your current directory

  ```
  cd ..
  ```

  This command will change your directory to its parent. If you are in /workspace/day1/, the command will place you in /workspace/.

- `ls`: list files/folders in directory

  ```
  ls
  ```

  This command will list all the files and folders in your current directory.

  ```
  ls -l
  ```

  This command will list all the files and folders in your current directory with timestamps and file permissions. This can help you double-check if your file updated correctly or change the read-write- execute permissions for your files.

- `mkdir`: make a directory

  ```
  mkdir dirname
  ```

  This command will make a directory within the current directory called `dirname`.

- `rm`: remove a file

  ```
  rm file1
  ```

  This command will remove file1 from the current directory. It will not work if `file1` does not exist.

  ```
  rm -r dir1
  ```

  This command will remove the `dir1` directory recursively. In other words, it will delete all the files and directories in `dir1` in addition to `dir1` itself. Be careful with this command!

- `cp`: copy a file

  ```
  cp lab1/original lab2/duplicate
  ```

  This command will copy the `original` file in the `lab1` directory and and create a `duplicate` copy in the `lab2` directory.

- `mv`: move or rename a file

  ```
  mv lab1/original lab2/original
  ```

  This command moves `original` from `lab1` to `lab2`. Unlike `cp`, mv does not leave original in the `lab1` directory.

  ```
  mv lab1/original lab1/newname
  ```

  This command does not move the file but rather renames it from `original` to `newname`.

There are some other useful tricks when navigating on a command line:

- Your shell can complete file names and directory names for you with *tab completion*. When you have an incomplete name (for something that already exists), try pressing the `tab` key for autocomplete or a list of possible names.
- If you want to retype the same instruction used recently, press the `up` key on your keyboard until you see the correct instruction. This saves typing time if you are doing repetitive instructions (like running Java programs on the command line while testing).

C. Test Run

Let's ensure that everything is working.

1. First open up your terminal. Check that git is a recognized command by typing the following command:

   ```
   git --version
   ```

   The version number for git should be printed. If you see "git: command not found", or similar, try opening a new terminal window, restarting your computer, or installing Git again.

2. Second, let's check that `javac` and `java` are working. Start by running the following commands at your terminal.

   ```
   mkdir ~/temp
   cd ~/temp
   ```

   1. Then, open your operating system's file explorer in this directory. You can do this from the command line:

      - Mac: `open .`
      - Windows: `explorer .`

- Ubuntu: `gnome-open .`
- Linux Mint: `xdg-open .` or `mate .`

2. In this newly opened directory, create a file `HelloWorld.java` with these contents:

```
public class HelloWorld {
    public static void main(String[] args)
        System.out.println("Hello world!");
    }
}
```

3. In your terminal, enter `ls` (list the files/folders in this directory). You should see `HelloWorld.java` listed.

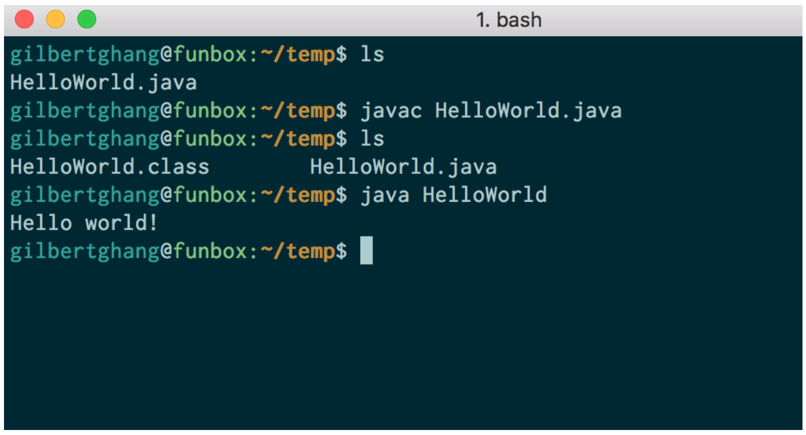
4. Run `javac HelloWorld.java`. If this produces any output, then something may be wrong with your setup. Try opening a new terminal window or restarting your computer. If that still doesn't work, see the Troubleshooting section under the directions for your operating system.

5. Type `ls`, you should see both `HelloWorld.java` and a freshly created `HelloWorld.class` (the `javac` command created this file).

6. Run `java HelloWorld`. It should print out “Hello world!” for you. If it didn't, something is wrong with your setup!

7. You're done! You can also delete the “temp” folder and its contents as you please.

The screenshot below shows what we're hoping for when we do steps 4-7. If you see something similar to this, your java setup is complete.



```
gilbertghang@funbox:~/temp$ ls
HelloWorld.java
gilbertghang@funbox:~/temp$ javac HelloWorld.java
gilbertghang@funbox:~/temp$ ls
HelloWorld.class        HelloWorld.java
gilbertghang@funbox:~/temp$ java HelloWorld
Hello world!
gilbertghang@funbox:~/temp$
```