Homework 0

1 Introduction

This assignment gets you started with the basic tools you will need to complete all of your homework projects. This project will

- ensure that you have correctly installed the Java SDK (Software Development Kit),
- give you practice using a text editor to write Java programs,
- give you practice compiling and running Java programs,
- give you practice identifying and locating an error,
- show you a bit of command line fun, and
- introduce you to Checkstyle.

2 Problem Description

You are a CS 1331 student who needs to install the Java SDK, configure it for command line use, and learn how to use a programmer's text editor to create and edit Java source code.

3 Solution Description

3.1 Setting Up Your Computer

1. Download and install the Java SDK on your computer using our installation instructions or Oracle's installation instructions
2. Download and install a programmer's text editor. You may end up trying out several over the course of the semester before you settle on one. See our guide to text editors.
3. Create a directory for your CS 1331 coursework somewhere on your hard disk. We suggest `cs1331`. Note: avoid putting spaces in file and directory names, since doing so complicates the use of some command line tools.
4. Create a `hw0` subdirectory of your CS 1331 coursework directory for your HW0 solution.

   On Unix/BASH you can create both of these directories at once with

   ```
   $ mkdir -p cs1331/hw0
   ```

   Note: the `$` is the command prompt (would be something like `C:\>` on Windows), the text after it is what you enter..

5. On the command line, go to the `hw0` directory you just created and enter these commands (including the change directory command):

```
$ cd cs1331/hw0
$ javac -version 2> hw0-output.txt
$ java -version 2>> hw0-output.txt
```

Please note what is happening here: > redirects the output of a program. `2>` redirects `stderr`, which is used for diagnostics (such as version strings). The first line creates the `hw0-output.txt` file, and the second line (with the extra >) adds to the file. Here is a nice discussion of the file descriptors `stdin`, `stdout` and `stderr`.

What this means is that > (or `2>`) will overwrite the file, so if you go back to repeat the first step, you'll need to repeat all the other steps as well.

3.2 Your First Java Program

1. Open your text editor and create a file in your newly created `hw0` directory named `NimblyBimbly.java` and enter the following Java program:

```
public class NimblyBimbly {

   public static void main(String[] args) {
      for (int i = 0; i < 9; i++) {
         System.out.print("\u004D\u0065\u006F\u0077 ");
      }
      System.out.println("...");
      System.out.println("\u004D\u0065\u006F\u0077\u0021");
   }
}
```

2. On the command line, go to the directory containing your newly created `NimblyBimbly.java` file and enter `javac NimblyBimbly.java`. Do a directory listing; you should see a file called `NimblyBimbly.class` that contains the compiled bytecode of your `NimblyBimbly` program. These commands should look like this:

```
$ javac NimblyBimbly.java
$ ls # the Windows equivalent of ls is dir
NimblyBimbly.class NimblyBimbly.java hw0-output.txt
```

3. Now enter `java NimblyBimbly` to run the program and see its output on the command line.

4. Add the output of your program to `hw0-output.txt` by running `java NimblyBimbly >> hw0-output.txt`.

3.3 Oh no. An Error

1. We have provided a file (`Schools.java`) for you that contains some sort of error. You can find this file in the `provided/` folder.

2. Navigate to the `provided` directory in your terminal.

3. Attempt to compile and run `Schools.java`.

4. At some point during this process you will encounter an error. Read this error carefully and determine where in `Schools.java` the error originated. That is, identify which line caused the error. It's alright if you don't understand what the error means or how to fix it, but do your best to reason through it and think about it. You will encounter lots of errors as you work through homework for this class, so it's important that you learn early how to decipher the error messages.

5. Finally, report your findings. Open up the `hw0-output.txt` file from before in your text editor. At the bottom, add two lines.

   (i) The first line should say when the error occurred - that is, during compilation or during running.

   (ii) On the second line, first put the *number* of the line causing the error. Then, copy the line itself.

   For example, if you compiled the program succesfully, the error happened when you ran it, and line 2 caused the error, you would add

```
runtime
2. public static void main(String[] args) {
```

   to the file.

4 Checkstyle

You must run checkstyle on your submission. The checkstyle cap for this assignment is **0** points.
Review the Style Guide and download the Checkstyle jar. Run Checkstyle on your code like so:

```
$ java -jar checkstyle-6.2.2.jar *.java
Audit done. Errors (potential points off):
0
```

The message above means there were no Checkstyle errors. If you had any errors, they would show up above this message, and the number at the end would be the points we would take off.

The Java source files we provide contain no Checkstyle errors. For this assignment, there will be a maximum of **0** points lost due to Checkstyle errors (1 point per error). In future homeworks we will be increasing this cap, so get into the habit of fixing these style errors early!

5 Turn-in Procedure

Non-compiling or missing submissions will receive a zero. NO exceptions

Submit `hw0-output.txt` to T-Square. Do not submit any compiled bytecode (`.class` files) or the Checkstyle jar file. When you're ready, double-check that you have submitted and not just saved a draft.

Please remember to run your code through Checkstyle!

5.1 Verify the Success of Your Submission to T-Square

Practice safe submission! Verify that your HW files were truly submitted correctly, the upload was successful, and that the files compile and run. It is solely your responsibility to turn in your homework and practice this safe submission safeguard.

1. After uploading the files to T-Square you should receive an email from T-Square listing the names of the files that were uploaded and received. If you do not get the confirmation email almost immediately, something is wrong with your HW submission and/or your email. Even receiving the email does not guarantee that you turned in exactly what you intended.
2. After submitting the files to T-Square, return to the Assignment menu option and this homework. It should show the submitted files.
3. Download copies of your submitted files from the T-Square Assignment page placing them in a new folder.
4. Recompile and test those exact files.
5. This helps guard against a few things.
   (a) It helps insure that you turn in the correct files.
   (b) It helps you realize if you omit a file or files. [1] (If you do discover that you omitted a file, submit all of your files again, not just the missing one.)
   (c) Helps find last minute causes of files not compiling and/or running.

[1] Missing files will not be given any credit, and non-compiling homework solutions will receive few to zero points. Also recall that late homework will not be accepted regardless of excuse. Treat the due date with respect. The real due date is midnight. Do not wait until the last minute!