Homework 0

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 JDK (Java 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

IMPORTANT: We do not use integrated development environments (IDEs) in this class. You must use the command line and a text editor to edit, compile, and debug your Java code.

Problem Description

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

Solution Description

Setting Up Your Computer

1. Download and install the JDK on your computer using our installation instructions (http://cs1331.gatech.edu/install-java.html) or Oracle's installation instructions (http://docs.oracle.com/javase/8/docs/technotes/guides/install/install_overview.html)
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 (http://cs1331.gatech.edu/text-editors.html).
3. Create a directory for your CS 1331 coursework somewhere on your hard disk; we suggest `cs1331`.
    - You can do this on the command line by navigating to the directory you want to contain is the cs1331 folder (using the `cd` command).
    - To create the folder use the command `mkdir cs1331`.
    - Enter the new folder by using the command `cd cs1331`.
    - Note: avoid putting spaces in file and directory names, since doing so complicates the use of some command line tools.
4. Create a subdirectory of your `cs1331` directory named `hw0`.
5. Download Schools.java (Schools.java) to your `hw0` directory.
6. On the command line, make sure you are in the `hw0` folder. Enter these commands:

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

Please note what is happening here:

`>` redirects the standard 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 more text to the file. Here is a nice discussion (http://www.jstorimer.com/blogs/workingwithcode/7766119-when-to-use-stderr-instead-of-stdout) 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.

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 using the command `ls` on Mac and Linux or `dir` on Windows; you should see a file called `NimblyBimbly.class` that contains the compiled bytecode of your `NimblyBimbly` program. These commands should look like this:

```
Mac / Linux:

$ javac NimblyBimbly.java
$ ls
NimblyBimbly.class NimblyBimbly.java hw0-output.txt build.gradle index.md ...

Windows:

C:\...\cs1331\hw0> javac NimblyBimbly.java
C:\...\cs1331\hw0> dir
NimblyBimbly.class NimblyBimbly.java hw0-output.txt build.gradle index.md ...
```

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
```

Oh no. An Error

1. We have provided a file (`Schools.java`) for you that contains some sort of error.
2. Attempt to compile and run `Schools.java` (using `javac` and `java`).

3. 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.
4. Finally, report your findings. Open up the `hw0-output.txt` file from before in your text editor. At the bottom, add two lines.
    1. The first line should say when the error occurred - that is, during compilation or during running.
    2. 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 successfully, 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.

Turn-in Procedure

Submit your `hw0-output.txt` file on T-Square as an attachment. When you're ready, double-check that you have submitted and not just saved a draft.

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 your program runs with no syntax or runtime errors. It is solely your responsibility to turn in your homework and practice this safe submission safeguard.

- 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.
- After submitting the files to T-Square, return to the Assignment menu option and this homework. It should show the submitted files.
- Download copies of your submitted files from the T-Square Assignment page placing them in a new folder.
- Re-run and test the files you downloaded from T-Square to make sure it's what you expect.
- This procedure helps guard against a few things.
    - It helps insure that you turn in the correct files.
    - It helps you realize if you omit a file or files. Missing files will not be given any credit, and non-compiling/non-running 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. Do not wait until the last minute! (If you do discover that you omitted a file, submit all of your files again, not just the missing one.)
    - Helps find syntax errors or runtime errors that you may have added after you last tested your code.