CMSC 113 – COMPUTER SCIENCE 1
Lab#1: Hello Computer Science, Linux, Bash, Java !

Objective

Familiarize yourself with the Linux environment, the **bash** command line shell (basic commands), and write your first **Java** program and learn how to **compile** and **run** a program using **VS Code** editor and the Java compiler.

This lab has two parts (PART 1 and PART 2). At the end of each part, you are asked to fill out the **Lab Report** (last sheet in this handout). This charts your progress in this lab. Please remember to submit the Lab Report to your instructor before leaving the lab. Submitting the report will count as proof of attendance in the lab. You are not required to complete everything in the Lab Report. Please, feel free to ask for help from your instructor during the lab.

PART 1: Starting the computer, logging in, and creating a course folder

1. **First, make sure you start the computer in Linux**
   The computers in the CS labs are configured to run two operating systems: Windows & Linux. Today, you will be doing your lab work using Linux.

   First make sure the computer is started into the Linux operating system. If you see a purple Login screen then you are using the Linux system. If you see the Windows start screen, restart the computer you can click the RESTART button (if you see one), or press the POWER button on the computer.

   Keep a close eye on your screen as the computer starts up and you will get a white screen with a Windows Logo and a Linux Logo next to it. By default, the computer will start as a Linux system. [To start into Windows press the LEFT ARROW key on the keyboard to select the Windows icon. Then press RETURN.]

   Now that you have the Linux login screen (the purple screen), proceed to the next step.

2. **Log in**
   In the login screen enter your username and password provided to you by your instructor. If you do not have a log in username/password, please see your instructor.

3. **Working with the Linux command line**
   Before doing any Java programming, let's do a warmup on learning and working with the command line. First, log in to your Linux account.

   Next, you will open a CLI window (*aka* Terminal emulator). Click on Activities (top-left corner of the screen) and enter **`Terminal`** in the search window that pops up. From the options it then gives you, click on the icon that is labelled **`Terminal`**. A window will pop up in the middle of your screen and it will have a prompt that may look something like this:

   ```
   [xena@codewarrior ~]$
   ```

   The above is a command prompt, implying that the system is ready for your commands. The

command prompt is preconfigured to show your username (in this case, **xena**), the symbol **@**, followed by the name of the computer you are logged into (in this case, **codewarrior**). This is then followed by the symbol "**~**" (which stands for your current home directory), and finally ends with a "**$**".

You type commands at the prompt and when you hit the RETURN key, the command is executed or carried out. For example, enter the command "whoami":

```
[xena@codewarrior ~]$ whoami
xena
```

The whoami command reports back the username of the person currently logged in (in this case, xena). Next, let us learn some other basic commands.

What is my present directory (i.e. home directory): pwd

```
[xena@codewarrior ~]$ pwd
/home/xena
```

Directories are organized in a tree structure. Reading the result of the above command from left to right, "/" represents the root directory, home is a subdirectory of / that is the parent directory of all users on this computer, of which xena is one. After the first /, the rest of the /'s are used to separate subdirectories under them. The string /home/xena is also called a **directory path**. For users, the symbol "~" is a shorthand for their home directory /home/xena. More on this later.

We will make a new directory, called cs113 (/home/xena/cs113), so that you can store all your files related to this course in or under that directory. To make a new directory, use the command: mkdir

```
[xena@codewarrior ~]$ mkdir cs113
[xena@codewarrior ~]$
```

While there is no visible result, the prompt reappears, this creates a directory, cs113, in the home directory (xena). Since directories are organized in a tree structure cs113 is a subdirectory under your home directory. To examine the contents of a directory, the command ls is used (ls stands for show a listing of this directory):

```
[xena@codewarrior ~]$ ls
abc.txt     cs113 hello.java
letters     mail
```

It appears from above that xena's home directory contains five items: a text file-abc.txt, the cs113 directory (just created), a Java program-hello.java, etc. Thus, in Linux, files and directories coexist in all directories. One way to tell files apart from directories is to note the file extension(s). For example, ".txt" indicates a text file, ".java" is a Java program, etc. Later we will see how you can tell which is which.

You can navigate in and out of directories using the `cd` command (`cd` stands for change directory):

```
[xena@codewarrior ~]$ cd cs113
[xena@codewarrior cs113]$
```

Look at the new prompt, it clearly indicated that you are now in the `cs113` directory. Go ahead and issue the pwd command now:

```
[xena@codewarrior cs113]$ pwd
/home/xena/cs113
```

Also, use the `ls` command to examine its contents. It should be empty. You will just get the prompt back.

The `cd` command can be used to navigate to any directory. You will use it to navigate up and down a directory tree. For example, when you are in the `cs113` directory (as you would be if you are following along), you can go up into its parent directory (`/home/xena`) by doing:

```
[xena@codewarrior 246]$ cd ..
[xena@codewarrior ~]$
```

Try it one more time:

```
[xena@codewarrior ~]$ cd ..
[xena@codewarrior /home]$ pwd
/home
```

Thus, "`..`" is shorthand for the parent directory. You can also enter the entire directory path to go to that directory:

```
[xena@codewarrior ~]$ cd /home/xena/cs113
[xena@codewarrior cs113]$ pwd
/home/xena/cs113
```

No matter what directory you are in, you can always get to your home directory by just typing the `cd` command by itself:

```
[xena@codewarrior home]$ cd
[xena@codewarrior ~]$
```

Also, try the command: `cd ~`

What does it do? Next, try this: Navigate to go to the root directory (`/`). Check, using `pwd`, to make sure that you are there. Check its contents (using `ls`). Then, to go back to your `cs113` directory, enter the command: `cd ~/cs113`

Now that you are comfortable travelling in and out of directories, we can learn about copying

files from one directory to another. The simplest form of a copy command is:

```
cp item1 item2
```

This command creates a copy of file `item1` into a file named `item2`, both in the same directory. Alternately, you can also specify to copy a file into another directory:

```
cp item1 directory-path
```

This command creates a copy of `item1` into the directory specified. The resulting copy will also be named `item1`. See item#5 in the exercise below for an example.

Exercise 1: Do the following:

Navigate to the directory `~dkumar/CMSC113/LabPrograms/Lab1`

Check its contents, using `ls`.

You will notice a file named `README.txt`

In order to read the contents of the file you can use any of the following commands:

```
cat README.txt
more README.txt
less README.txt
```

These commands will each show the contents of the file specified. You will not notice any difference in the way these commands behave. We will examine these later.

Before starting the next exercise, please answer Question 1 in Lab 1 Report.

Exercise 2: Using the commands you have learned above, answer Question 2 on the Lab1 Report.

PART 2: Creating your First Java Program

As shown in class, creating and running Java programs requires three steps:

- Use an editor to write the program and save it in a file (extension **`.java`**) – **`VS Code`**
- Compile the Java program. Correct any syntax errors reported – **`javac-introcs`**
- Run the program – **`java-introcs`**

Let's see how we do this. First, we need to have a program we'd like to run:

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

1. **Use VS Code to create a program/source file.**

   Start VS Code by entering the command – **code**:

   ```
   [xena@codewarrior home]$ code
   ```

   The VS Code window will start. Enter the program above into a file called `HelloWorld.java`. The name of the file should be the same as the name of the class (always!). Make sure you have saved the program in the `cs113` directory. [Optional: You may want to create a directory, called **Lab1**, and do this there.]

2. **Compile the program.**

   To compile the program, use the following command in the command shell:

   ```
   [xena@codewarrior home]$ javac-introcs HelloWorld.java
   [xena@codewarrior home]$
   ```

   Depending on how correctly you typed your program, you may or may not have any syntax errors. In case there are no errors, the prompt will be returned as shown above. Otherwise, these will be reported following the command. You will then have to correct the errors in the Atom window, save the file, and then try to compile again.

   So, what is the purpose of compiling the program? Well, one is to detect and ensure that what you entered is a correct Java program. Second, to translate the Java program into Java byte code. This is essentially a version of your program translated into a more primitive language that a Java Virtual Machine (JVM) will be able to understand and run it. More on that in class.

   The byte code generated by the compiler is stored in a new file. In this case, since we defined the class `HelloWorld`, the file will be called `HelloWorld.class`. Go ahead and look at the contents of your `cs113` directory (using `ls`). You will see the file `HelloWorld.class` sitting there. We are now ready to run your program.

3. **Run the program**

   You run the program by invoking the JVM (which is called `java-introcs`). The JVM needs to know the name of the class that makes up your program (i.e `HelloWorld`). Here is the command:

   ```
   [xena@codewarrior home]$ java-introcs HelloWorld
   Hello, World!
   [xena@codewarrior home]$
   ```

   The program runs, you can see its output on the line after the java command. And a new prompt is returned. You can now run the program again, using the same command.

Exercise 3: Write a new program, called `JavaJoe`, that prints out the following lyrics:

```
I love coffee
I love tea
I love the Java Joe
And it loves me
```

Exercise 4: Write the program, `UseArgument` that is described on Page 7 (Program 1.1.2) of your text. It is shown below:

```
public class UseArgument {
   public static void main(String[] args) {
      System.out.print("Hi, ");
      System.out.println(args[0]);
      System.out.println(". How are you?");
   } // main()
} // class UseArgument
```

You will store it in a file, `UseArgument.java`. Compile it (using `javac-introcs`), and run it using the command:

```
java-introcs UseArgument <your name>
```

Once done, answer Question 3 in the Lab Report and submit the Lab Report to your instructor. You are done for the day. Congratulations!

Time to Digest

In PART 1, you learned some basics of using the Linux command line interface (CLI) through a terminal window. Review the following commands:

whoami
pwd
mkdir
ls
cd
cp
mv
cat/more/less

In PART 2 you learned how to create, compile, and run simple Java programs.

This is a good start. Before next lab, please review Section 1.1 of your text and try out all Exercises (1.1.1 through 1.1.6).