# CMSC 113 – COMPUTER SCIENCE 1
# Lab#1: Hello Bash, VS Code, & Java

## Objective

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

## Create some folders: CMSC113 and Lab1

1. **Start your computer**

2. **Log in**

3. **Create a CMSC113 Folder**
   On your own computer you will be storing your work under your home folder. For "`Deepak Kumar`", for example, the home folder can be found in: "`C:\Users\Deepak Kumar`".

   That is, on drive C: inside the "`Users`" folder, there is a folder called "`Deepak Kumar`". Go ahead and locate your home folder on your computer and navigate to it and create a new folder: **`CMSC113`**.

   This is where you can store all your class work. We recommend you further create subfolders inside CMSC113 for each of the tasks (e.g. Lab1, Lab2, etc. and Assignment1, Assignment2, etc.). For example, for this lab, create a folder: **`Lab1`** inside the CMSC113 folder.

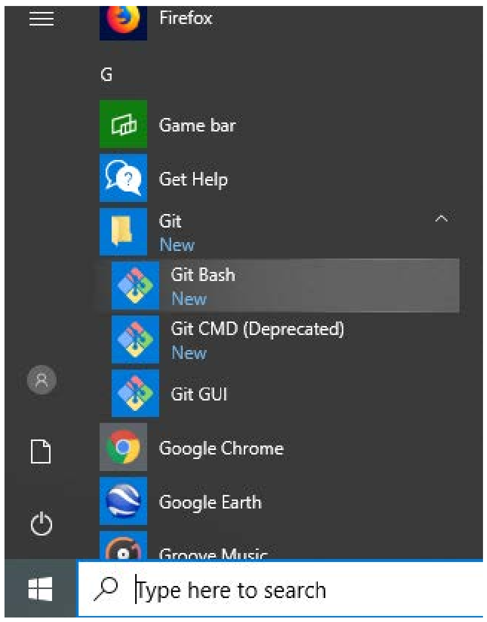
## Linux-style Command Line Interface (CLI) – Using the Bash Shell

This section has three parts- PARTA, PARTB, and PART C. Our goal today is to learn some basic Linux commands to navigate files and directories (PART A), learn how to create and edit text files (PART B), and finally, how to write, compile, and run Java programs. Please follow the handout in the order written and DO everything that is requested of you. Do not hesitate to ask the instructor in case you have any questions during the lab. This is highly encouraged! You may not be able to complete the entire lab in today's section. This is by design. Please, take some time this week, before your next class, to return to the lab to finish.

## Part A: Working with the Linux command line- Bash Shell [15-20 min]

While you are using a Windows computer, you will actually be working in an environment very similar to a Linux computer. Beyond this semester, if you take more computer science classes, you will primarily be using Linux computers. This class will prepare you to be ready for that experience.

In Linux, instead of using a mouse to navigate, most computer scientists use a command line interface (CLI) to interact with the computer. The interface is very similar to a Windows command shell or a MacOS command shell, if you have ever had an opportunity to use it. The command shell (or CLI) you will be using is called **Git Bash**[1]**.** You installed Git Bash in Lab#0. Let us learn how to use it.

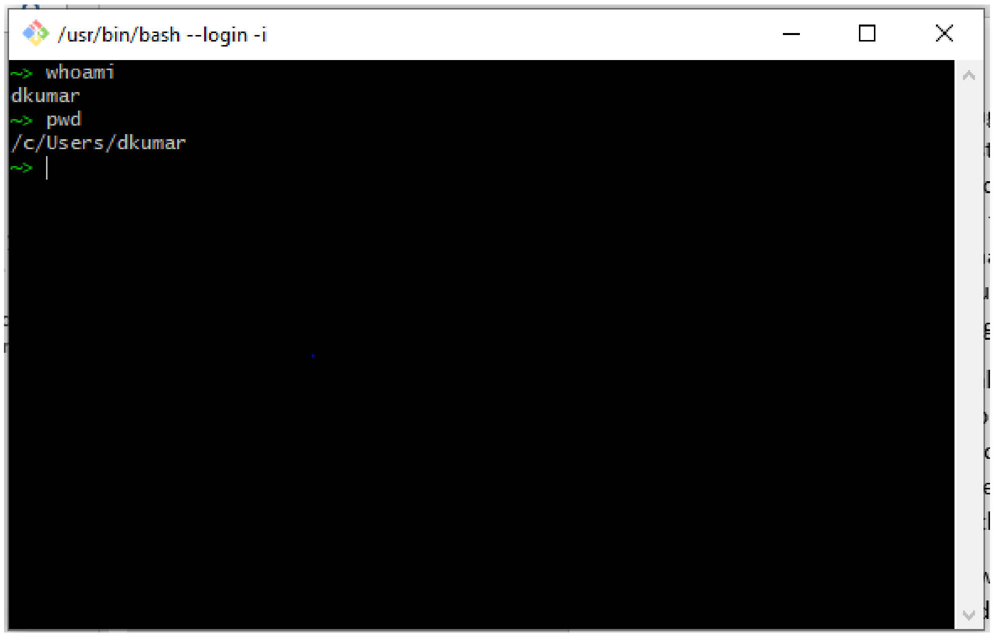


Find the Git Bash app in your Windows menu (see picture on right) and start the app.

Bash Prompt

You will then get a CLI window (*aka* Terminal emulator or a bash shell) and it will have a prompt that looks like this:

~>



```
~> whoami
dkumar
~> pwd
/c/Users/dkumar
~> |
```

**Figure 1: The bash Command Line Interface (CLI)**

The above is a command prompt, implying that the system is ready for your commands. The command prompt is preconfigured to show the symbol "~" (which stands for your current home directory), and 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`" (as shown in Figure 1) and it will show the name of the user. [What you have to type is underlined from now on.]

[1] Ask you instructor to explain what "bash" means…

```
~> whoami
dkumar
```

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

What is my present folder (called a directory, in Linux parlance): pwd

```
~> pwd
/c/Users/dkumar
```

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

Earlier, you made a new directory, called CMSC113 in your account, where you will store all the files related to this course. You can also make a new directory, using the command: mkdir

```
~> mkdir Test
~>
```

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

```
~> ls
abc.txt    hello.java
letters    mail/ Test/
```

It appears from above that dkumar's home directory contains five items (yours will look different): a text file-abc.txt, a Java program-hello.java, a directory called Test (that we just created), etc. Thus, 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. In the listing anything that has a "/" after its name is a subdirectory.

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

```
~> cd Test
~/Test>
```

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

```
~/Test> pwd
~/Test
```

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 `Test` directory (as you would be if you are following along), you can go up into its parent directory (`/c/Users/dkumar`) by doing:

```
~/Test> cd ..
~>
```

Thus, "`..`" is shorthand for the parent directory. Try the `pwd` command to see what directory you are in (it should be /c/Users/dkumar). You can also enter the entire directory path to go to that directory:

```
~> cd /c/Users/dkumar/Test
~/Test> pwd
/c/Users/dkumar/Test
```

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

```
~/Test> cd
~>
```

Also, try the command: `cd ~`

What does it do? Next, try this: Remember the `CMSC113` directory you created earlier in your computer? You probably created under your home folder. Or, perhaps in your `Documents` folder. Let's go there. Go ahead and find out were.

If you created it in your home folder, enter the command: `cd CMSC113`

If you created it inside your `Documents` folder, enter: `cd Documents/CMSC113`

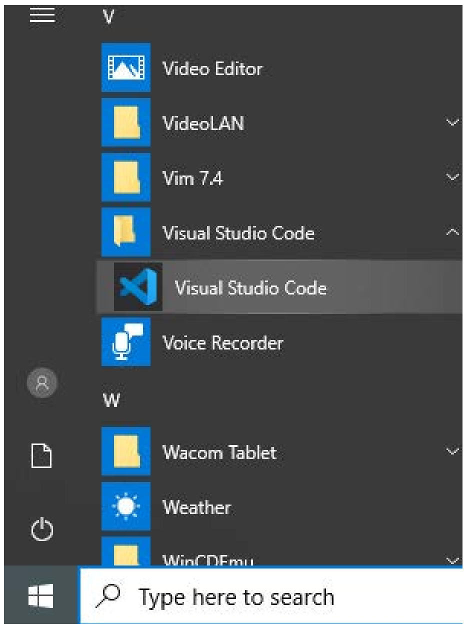
Do a `pwd` and `ls` to make sure you are in the correct place. If you also created a `Lab1` directory in CMSC113 go there. If not, create it now, using the `mkdir` command.

It takes a little time to get used to this mode of interaction. You should practice as much and as often as you can and you will soon become familiar with these commands and interacting with a CLI bash shell. Here is a summary of all the commands we introduced so far:

| **Command** | **Description** |
|---|---|
| **cd** | Change directory (e.g. cd /h/CMSC113/Lab1) |
| **ls** | List contents of current directory (subdirectories will be listed with "/" prefix) |
| **mkdir** | Create a new directory/folder in the current directory (e.g. mkdir Lab1) |
| **pwd** | Show the name of present working directory |
| **whoami** | Show name of the user that is logged in |

**PART B: Creating and editing files using Visual Studio Code [10 min]**

With the little comfort you now have navigating in and out of and creating directories it is time to learn how to create plain text and program code files. Let's first do plain text files.



1. **Start the VS Code app**
   Find the VS Code app that you installed as part of Lab#0 and start it. See picture on right.

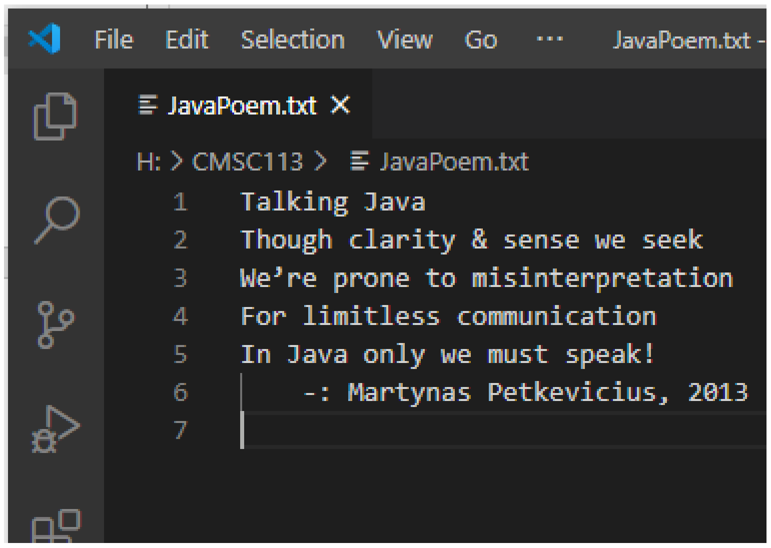
   The VS Code app will pop up a window.
2. **Enter some text**
   From the File menu, select "New File" and then enter the following text into it:

```
Talking Java

Though clarity & sense we seek
We're prone to misinterpretation
For limitless communication
In Java only we must speak!

     -: Martynas Petkevicius, 2013
```

3. **Save the file in CMSC113 directory**
   Next, we will save this as a text file, named "`JavaPoem.txt`" in the CMSC113 directory.

   Again, from the `File` menu, select the "`Save As…`" option. In the navigator window that pops up, navigate to your CMSC113 directory, and then enter the name of the file in the box at the bottom: `JavaPoem.txt` and hit the "Save" button.

To confirm that you have saved the file in the correct place, go into the Git Bash shell window, navigate to the CMSC113 directory and use the `ls` command. You should see the file listed:



```
Talking Java
Though clarity & sense we seek
We're prone to misinterpretation
For limitless communication
In Java only we must speak!
     -: Martynas Petkevicius, 2013

```

Figure 2: Creating a text file: JavaPoem.txt

```
~/CMSC113> ls
JavaPoem.txt          Lab1/
~/CMSC113>
```

You should see something similar to the above: the text file and also the Lab1 directory is listed.

In the shell, you can examine the contents of a text file using the `cat` command:

```
~/CMSC113> cat JavaPoem.txt
Talking Java

Though clarity & sense we seek
We're prone to misinterpretation
For limitless communication
In Java only we must speak!

     -: Martynas Petkevicius, 2013

~/CMSC113>
```

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

```
cat JavaPoem.txt
more JavaPoem.txt
less JavaPoem.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.

Now, 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`.

**Exercise:** Do the following:

Navigate to the directory `~/CMSC113/` (or, it may be `~/Documents/CMSC113`)

Check its contents, using `ls`.

Let's make a copy of the JavaPoem.txt file into the CMSC113 directory:

```
~/CMSC113> cp JavaPoem.txt JavaPoemCopy.txt
```

Then try the `ls` (and `cat/more/less`) command(s) to examine the contents of the directories and the contents of the copied file. We do not really need two copies of the file so we can remove one of them using the "`rm`" command (`rm` stands for remove):

```
~/CMSC113> rm JavaPoemCopy.txt
~/CMSC113>
```

Again, use the `ls` command to see if the file was removed.

Next, let's copy the file into `Lab1` directory:

```
~/CMSC113> cp JavaPoem.txt Lab1/
~/CMSC113>
```

Go to the `Lab1` directory (use the `cd` command) and examine its contents to see that the file was copied. Examine its contents using `cat/more/less`. Now that we have two copies of a file, we can delete (or remove) the one sitting in `CMSC113` directory. Go ahead and do it.

**PART C: Creating and running your First Java Program [10 min]**

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 an Editor to create a program/source file.**

   Using **VS Code**, as you did in PART B, 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 Lab1 directory of inside the `CMSC113` directory.

2. **Compile the program using the command javac-introcs**

   To compile the program, in Git Bash, first navigate to the `Lab1` directory. Then, use the following command:

   ```
   ~/CMSC113/Lab1> javac-introcs HelloWorld.java
   ```

```
~/CMSC113/Lab1>
```

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 VS Code 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 `Lab1` directory (using `ls`). You will see the file `HelloWorld.class` sitting there. We are now ready to run your program.

3. **Run the program using the command java-introcs**

   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:

```
~/CMSC113/Lab1> java-introcs HelloWorld
Hello, World!
~/CMSC113/Lab1>
```

   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 1:** 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 2:** 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`), and run it using the command:

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

**Time to Digest and Wrap up!**

In part A, you learned some basics of using the Linux command line interface (CLI) through a terminal window. Review the following LINUX/Bash commands:

```
cat
cd
cp
less
ls
mkdir
more
pwd
rm
whoami
```

Also, you briefly saw the two directory shorthand symbols: "~" (home directory) and "`..`" (parent directory).

In PART B you learned how to create basic text files using VS Code. And, as you saw in PART C, all program files are also text files. In PART C you learned how to create, compile, and run simple Java programs using the commands:

javac-introcs
java introcs

This is a good start.

**Homework before next class:**

Please, read Section 1.1 of your text and try out all Exercises (1.1.1 through 1.1.6).