CSE 002: Fundamentals of Programming

Laboratory Session 1: Hello World

Objectives. This lab session is intended to give you a first experience in writing software. You will traverse and create the unix directory file system, set up your account for submitting homework 1, and write a 'hello world' program. While the directions below are provided in detail, their purpose is for you to understand exactly what each command does, so that you can use it in the homework. The training wheels come off in homework 1.

Phase 0. Log into your sun lab account.
If you are in the Sandbox Lab (PL 112) then turn on the computer. It will boot up to a the java desktop log in screen. If you are in the Sun Lab, then the computers should already be at this screen. Type in your Lehigh user name and your Sun Lab password. This information was provided to you in an email with subject "Your CSE/ECE Workstation Network account is available."

Phase 1. Create a directory for the CSE002 class and for submitting homework 1.
First, change the current directory to your home directory with the command "cd ~;". Then create a directory for your work in this class: "mkdir cse002.134;". The ".134" part denotes the current semester, and helps the automatic homework collection process. Finally, create a submission directory for homework 1: "mkdir cse002.134/Hw1".

You have thus created two directories – one for your work in the class, and one inside that for submitting homework 1. For subsequent homework assignments, you will create other directories – Hw2 and so on. Note that capitalization must be exactly as shown here. Your account is now ready to submit homework 1.

Phase 2. Write a hello world program.
Now, open Dr. Java. Start by saving the file as "HelloWorld.java". The save command is in the file menu at the upper left. Next, as we always do, start by adding comments that describe what this program does. A comment is any line that starts with two slashes.

Phase 3. Modify the "HelloWorld.java" program to create a box around the message "Hello World".

```
////////////////////////////////////////////////////////////////////////////////
//  How to compile this program:
//     javac HelloWorld.java
//  How to run this program:
//     java HelloWorld//
//
//  Prints "Hello, World". This is your first program.
//
//  These lines are "Comments" - they are lines in a program that DO NOTHING, but
//  are used by programmers to explain what some piece of code does.  Remember, code
//  is CODE, so it is not easy to read.  Good programming practices always involve
//  adding comments to explain what code does, to make it easier for other people to
//  read your code: That makes the difference between code worth writing, and code
//  that no one else can use or read.
////////////////////////////////////////////////////////////////////////////////
```

Remember, comments help you and others read the code that you are writing. **The standard for writing comments is that a person that does not know how to read java can still read the comments to get a clear idea of how the program works.** You will be evaluated and graded on how well your software is commented, and it will be a significant portion of your grade. While a “hello world” program is as simple as it gets, and while it is going to feel obvious to you after you finish, imagine how little the code would make sense to someone with no computer knowledge. That person might be your boss. Write your comments for that person.

Phase 2a. Now add the class structure:

```
// This line defines a class.  We will not use classes in CSE002
// but in Java we must always have at least one.  So we define this here.
public class HelloWorld
{

}
// This curly brace ends the class, which we defined above.  Note that it also matches
// a curly brace next to the class definition.  All curly braces must match a definition.
```

Note that the class structure has two curly braces, because it is a container for code. They must match, or the computer will get confused. Note also that the class is named “HelloWorld”. It is not an accident that you are putting it in a file called “HelloWorld.java”. In fact if the beginning of the file name (e.g. before the “.java” part) is not the same as the class name, the software will not compile. Try it – what error does the compiler throw?

Phase 2b. Now add the main method INSIDE the class structure.

```
// This line defines a main method.  The computer ALWAYS starts reading
// commands from the main method.  The main method is a special kind of
// method that must exist in all programs, so we use it here.  We will go
// into detail about methods in general later in the class.
public static void main(String[] args) {

     // Now we end the main method.  This tells the computer that the main method
     // is finished, so the task is over.  Other methods finish in the same way.
     // Notice also that this curly brace is matched to the curly brace next to
     // to the method declaration.  All curly braces must match or the computer
     // will get confused, and likely give you a compiler error.
}
```

Since you are adding the main method INSIDE another code container, make sure that everything you add is indented by one tab, just like how the comments inside the main method (above) are indented by one tab. This is just one of many ways that you can make your code easier to read for someone who does not understand how software works. This is a habit that you should adopt now - making things easier to read costs you 30 seconds when you write it, but it saves you hours of grief and debugging time in the future. It is time wisely spent.

Note that the main method also uses curly braces. They again must match or the computer will get confused as to what container holds the code. Try deleting one and pressing compile – what happens?

Phase 2c. Finally, add the print statement for the “Hello World”:

```
// This is our first and only command.  We call a special function
// called "println" that causes the code to write text out to the screen.
System.out.println("Hello, World");
//Note that this command ends with Semicolon.  ALL commands end with
// semicolon
```

Note that you are adding this inside the main method, which is a code container, so it must be indented further. That is okay – we have plenty of horizontal space in the code. To check your work, the entire program is written on the last page

Phase 2d. Now, it is time to compile your code:
Press the compile button on the upper right of the Dr. java screen. This will run the compiler. If you had any mistakes when typing this in, the compiler will show errors in translating the code. That's just fine – it happens to everyone. Just find the line and figure out what was typed in wrong. It could be that there was a misspelling. It could be that you forgot to pair one of the curly braces. It could be that the class name does not match the file name. If after a comprehensive search you cannot seem to figure out what is wrong, ask one of the class staff to help you, but remember – it is your responsibility to figure out what went wrong.

Compilation errors can sometimes be frustrating because the compiler does not always give a very clear message as to what the error is. It is actually very very difficult to make compiler software that can do that for all errors, so bear with the compiler. Finding bugs is an aspect of computer science that makes it different from almost all other disciplines, and you aren't expected be used to that feeling yet – you simply cannot know how long it will take you to find the bug until you actually find the bug. That feeling goes away as you get much more experienced with programming, but for now, plan ahead for it: Assume that debugging will be ridiculously hard, and start early in anticipation for it.

Phase 2d. Now, run your code:
If you worked out the compiler errors, you should now be pleased to find the computer printing out "Hello World" in the interactions pane in Dr. Java. Congrats, you're done! You finished your first program!

Phase 3. Now that you have a working program, it is time to modify this program to create a box around the message "Hello World". Make the modifications, save and compile your program again and run it. The output should exactly match the following, without any mismatched lines. Do not forget to add comments when you modify your program.

```
##################
#                #
#   Hello World  #
#                #
##################
```

Phase 4. Make sure that you have saved your work.

Save your work regularly and often. You work hard to write code, and it is always frustrating to lose it. Don't forget to save your work.

Phase 5. Things to try (OPTIONAL):
Try getting the code to do the following:

a) Print out something other than "Hello World"
b) Print out "Hello World" with two boxes around it

```
////////////////////////////////////////////////////////////////////////////////////
//  How to compile this program:                                                  //
//     javac HelloWorld.java                                                      //
//  How to run this program:                                                      //
//     java HelloWorld                                                            //
//                                                                                //
//  Prints "Hello, World". This is your first program.                            //
//                                                                                //
//  These lines are "Comments" - they are lines in a program that DO NOTHING, but //
//  are used by programmers to explain what some piece of code does.  Remember, code //
//  is CODE, so it is not easy to read.  Good programming practices always involve //
//  adding comments to explain what code does, to make it easier for other people to //
//  read your code: That makes the difference between code worth writing, and code //
//  that no one else can use or read.                                             //
////////////////////////////////////////////////////////////////////////////////////


// This line defines a class.  We will not use classes in CSE002
// but in Java we must always have at least one.  So we define this here.
public class HelloWorld {

     // This line defines a main method.  The computer ALWAYS starts reading
     // commands from the main method.  The main method is a special kind of
     // method that must exist in all programs, so we use it here.  We will go
     // into detail about methods in general later in the class.
     public static void main(String[] args) {

          // This is our first and only command.  We call a special function
          // called "println" that causes the code to write text out to the screen.
          System.out.println("Hello, World");
          //Note that this command ends with Semicolon.  ALL commands end with
          // semicolon

          // Now we end the main method.  This tells the computer that the main method
          // is finished, so the task is over.  Other methods finish in the same way.
          // Notice also that this curly brace is matched to the curly brace next to
          // to the method declaration.  All curly braces must match or the computer
          // will get confused, and likely give you a compiler error.
     }
```

```
}
// This curly brace ends the class, which we defined above.  Note that it also matches
// a curly brace next to the class definition.  All curly braces must match a definition.
```