National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Computing

CS 212: Object Oriented Programming

Class: BSCS-6ABC

Lab01: Basic Programming in Java

Date: 14th February, 2017

Time: 2:00pm- 4:50pm

Instructor:

Dr. Mian M. Hamayun/Dr. Anis ur Rahman

Learning Objectives

After completing this lab you will be able to do variable declarations and initializations with some printing and math in Java.

Your first activity will be creating the "Hello World" program using Java. We'll try a few experiments using it to see what happens for some basic errors. As you carry out the steps in this lab, it is important to think about what you are seeing and to understand what's going on. The more attentive you are here the fewer headaches you'll have later!

Basic Lab Instructions!

- Talk to your classmates for help.
- You may want to bring your textbook to future labs to look up syntax and examples.
- Stuck? Confused? Have a question? Ask a TA/Lab Engineer for help, or look at the book or past lecture slides.
- Complete as many problems as you can within the allotted time. You don't need to keep working on these exercises after you leave the lab.
- Before you leave today, make sure to check in with one of the Lab Engineers/TAs in the lab to get credit for your work.

Activity #1.

Your first task will be creating the "Hello World" program using java. Create a new directory CS-212, and navigate into it. Using notepad or the editor of your choice (preferably NetBeans), create a file called ActivityOne.java and enter the following code:

```
// This program outputs the message "Hello World"
public class ActivityOne
{
	public static void main (String args [])
	{
		System.out.println ("Hello World");
	}
}
```

Save the file, and open command window to use for compiling and running your program. In the new window, navigate into your CS-212 directory and issue the following command to compile the program, if you are working on a Linux like environment (In case you are using NetBeans, just Build and Execute the project).

javac ActivityOne.java

If you get any compiler messages, then the learning process has already started! Make sure the name of the file and the name of the class are identical, that you're in the right place in the directory structure, and the code is exactly as shown above.

Once the program compiles, enter the command to run it.

java ActivityOne

You will see the output in the console window.

After getting the program to work, let's see what happens if we deliberately break it. We'll try several variations that violate java syntax to learn what happens. Watch carefully to see what message the compiler generates for each of these small errors—they are mistakes every programmer makes on a regular basis.

1) Start by changing the name of the class in the first line, saving the file, and then attempt to re-compile it with the javac/NetBeans Build command. Observe what happens.
2) Now restore the name in the class header and try removing one of the opening braces, then save and compile again. Put the brace back and remove the closing brace that matched it, and save and compile again. Make a note of the messages the compiler generates for each change.
3) Try a version with a lowercase "s" at the beginning of the output statement instead of the uppercase "S."
4) Finally, try a version where the only change is the removal of the semicolon at the end of the print statement. This is a mistake every programmer makes more than once—it's good to see how the compiler responds.

Activity #2.

Modify your ActivityOne file to produce the following console output. Note the blank lines; you should include those in your output.

Hello, world!
I am taking CS 212.
I hope it is a lot of fun!

My teacher’s name is Hamayun / Anis.

I hope he gives me a grade of 4.0!

Activity #3.

Programs should be indented properly to make them easier to read.

{ brace -> increase indent of following lines by one tab
} brace -> decrease indent of that line and following lines by one tab

The following program has poor indentation. Fix it.

```
  public class ActivityThree {
public static void main(String[] args) {
System.out.println("Properly indented programs");
    System.out.println("look ever so much better!");
 System.out.println("please fix me");
       System.out.println("and make me beautiful again");
}
  }
```

Activity #4.

How many lines of output are produced (including blank lines)?

```
public class ActivityFour {
   public static void main(String[] args) {
      System.out.println("Testing, testing,");
      System.out.println("one two three.");
      System.out.println();


      System.out.println("How much output");
      System.out.println();
      System.out.println("will there be?");
   }
}
```

Activity #5.

The following program contains 11 errors! What are they?

```
public class ActivityFive
   public static main(String args) {
      System.out.println(Hello world);
      system.out.Pritnln("Do you like this program"?);
      System.out.println()

      System.println("I wrote it myself.";
   {
}
```

Once you think you've found the errors, create/compile/run a corrected version of this program.

Activity #6.

Discover what error messages the compiler produces when you make each of the following mistakes. How many unique error messages are you able to cause the compiler to produce?

1) Naming your file incorrectly, then compiling.
2) Forgetting a keyword such as void or class
3) Forgetting a quotation mark "
4) Forgetting a parenthesis (or)
5) Forgetting a dot .
6) Using too many or too few braces { or }

Notice that the error messages don't always make it obvious what is wrong. But they usually tell you the right line number to fix.

Activity #7.

Write a complete Java program that produces the following output (note the blank line):

A "quoted" String is
'much' better if you learn
the rules of "escape sequences."

Also, "" represents an empty String.
Don't forget: use \" instead of " !
" is not the same as "

Activity #8.

Write a complete program that produces the following output:

Activity #9.

Change the following program to use compound assignments:

```
class ActivityNine {

    public static void main (String[] args){

        int result = 1 + 2; // result is now 3
        System.out.println(result);

        result = result - 1; // result is now 2
        System.out.println(result);

        result = result * 2; // result is now 4
        System.out.println(result);

        result = result / 2; // result is now 2
        System.out.println(result);

        result = result + 8; // result is now 10
        result = result % 7; // result is now 3
        System.out.println(result);
    }
}
```

Task #1:

Writing a program that stores a student's year (Freshman, Sophomore, Junior, or Senior), the number of courses the student is taking, and his or her GPA on a 4.0 scale. Declare variables with the appropriate names and types to hold this information. Print the stored information at the end.

Task #2:

Write an application that inputs one number consisting of five digits from the user, separates the number into its individual digits and prints the digits separated from one another by three spaces each. For example, if the user types in the number 42339, the program should print:

4 2 3 3 9

Assume that the user enters the correct number of digits. What happens when you execute the program and type a number with more than five digits? What happens when you execute the program and type a number with fewer than five digits?

Task #3:

Suppose you have a real number variable x. Write a Java expression that computes the following value y while using the * operator only four times:

$$y = 12.3x^4 - 9.1x^3 + 19.3x^2 - 4.6x + 34.2$$

Task #4:

In physics, a common useful equation for finding the position s of a body in linear motion at a given time t, based on its initial position s0, initial velocity v0, and rate of acceleration a, is the following:

$$s = s_0 + v_0 t + \frac{1}{2} a t^2$$

Write code to declare variables for s_0, v_0, a, and t, and then write the code to compute s on the basis of these values.

Hand in

Hand in the source code from this lab at the appropriate location on the LMS system. You should hand in a single compressed/archived file named Lab_1_<Your CMS_ID. Your_NAME >.zip (without angle brackets) that contains ONLY the following files.

1) All completed java source files representing the work accomplished for this lab: ActivityOne.java; ActivityTwo.java; ActivityThree.java; ActivityFour.java; ActivityFive.java; ActivitySix.java; ActivitySeven.java; ActivityEight.java; ActivityNine.java; Task1.java, Task2.java, Task3.java, Task4.java. The files should contain author in the comments at the top.
2) A plain text file named **README.TXT** that includes a) author information at the beginning, b) a brief explanation of the lab, and c) any comments, or suggestions.

To Receive Credit

1. By showing up on time for lab, working on the lab solution, and staying to the end of the class period, only then you can receive full credit for the lab assignment.
2. Comment your program heavily. Intelligent comments and a clean, readable formatting of your code account for 20% of your grade.
3. The lab time is not intended as free time for working on your programming/other assignments. Only if you have completely solved the lab assignment, including all challenges, and have had your work checked off for completeness by your TA/Lab Engineer should you begin the programming/other assignments.