CS202: PROGRAMMING PARADIGMS & PRAGMATICS

Semester II, 2020 – 2021

Lab 1: Introduction to Java

Aim: Introduce you to programming in Java using command-line interface in Linux. *This lab assumes that you are familiar and comfortable using the Linux operating system!!*

- **Let's get started!**
  - Create a directory structure to hold your work for this course and all the subsequent labs:
    - Suggestion: `CS202/Lab1`

- **First Java Program: Hello World!**
  - The first java program that you write will be in a file named `HelloWorld.java`
  - To create that file and start editing it, use the following command in the terminal (file names are case-sensitive): `gedit HelloWorld.java`
  - File will be created in the current directory, which should be `CS202/Lab1`
  - The traditional first program just says "Hello World!". To make that happen, type the following lines of Java code into the editor:

```
public class HelloWorld {

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

  - The name of the class is "HelloWorld" with upper case "H" and "W". The name of the class must match the first part of the file name, HelloWorld.java. After typing in the code, save your file and exit from gedit.
  - Back on the command line, compile the program by entering the command: `javac HelloWorld.java`
  - If you have made no mistakes while typing the program, this command will succeed with no error messages; you will simply get another command prompt. If there are errors in the program, they will be listed. In that case, you have to edit the file to fix the errors. Once the program has been compiled successfully, you will have a new file (compiled program) named `HelloWorld.class`. Use the `ls` command to check that it is there. Once you have `HelloWorld.class` , you can run the program by entering the command: `java HelloWorld`
  - Note that this command uses the class name `HelloWorld`, not the name of the file `HelloWorld.class.` The program should simply print 'Hello, World!'
  - Before proceeding, if you haven't already seen an error from the compiler, you should change the program to introduce an error, and see what happens when you compile it. (Try removing one of the semicolons, for example.) Note that the error messages from compiler will contain line numbers that tell you where the compiler found an error. Pay attention to the line numbers—they can help you find the errors. But the line numbers aren't foolproof; often the line that you have to change to fix the error is not the same as the line where the compiler noticed a problem.

- **A Program with Computation:**
  - For a second example, let's write the Fahrenheit-to-Celsius conversion program.
  - To begin, use the command in the terminal: `gedit Temperature.java &`
  - Then type the following program into the gedit window:

```
public class Temperature {
    public static void main(String[] args) {
        double degFahrenheit;
        double degCelsius;
        degFahrenheit = 98.6;
        degCelsius = (degFahrenheit - 32) / 1.8;
        System.out.print(degFahrenheit);
        System.out.print(" degrees Fahrenheit equals ");
        System.out.print(degCelsius);
        System.out.println(" degrees Celsius");
    }
}
```

  - Save your program, compile it, and run it, to make sure it works, using the commands in the terminal:

```
javac Temperature.java
java Temperature
```

- **A Simple String Example:**
  - This example is used to describe how the Java String object is created and used
  - Create a new file using the command in the terminal: `gedit JavaStringExample.java &`
  - Then type the following program into the gedit window:

```
public class JavaStringExample{

  public static void main(String args[]){

  /* String in java represents the character sequence. */

  /* creates new empty string */
  String str1 = new String("");

  /* creates new string object whose content would be Hello World */
  String str2 = new String("Hello world");

  /* creates new string object whose content would be Hello World */
  String str3 = "Hello Wolrd";

  /* IMPORTANT : Difference between above given two approaches is,
string object created using new operator will always return new string
ojbect. While the other may return the reference of already created
string ojbect with same content , if any. */

  System.out.println( str1.length());

  }

 }
```

- Save your program, compile it, and run it, to make sure it works, using the commands in the terminal:

```
javac JavaStringExample.java
java JavaStringExample
```

- OUTPUT of the above given Java String Example would be?

- **OOP Basics**
  - For this part, refer to `OOP Basics.pdf`
  - Note: This part has been adapted from : https://www.ntu.edu.sg/home/ehchua/programming/java/J3a_OOPBasics.html

- **Exercise 1:**
  - As an exercise in writing your first Java program, write a program that converts 45 inches into an even number of feet plus some number of inches left over. For this program, you must use variables of type `int`. If a variable *inches* of type `int` holds the total number of inches, then the expression:

    ```
    inches / 12
    ```

    computes the integral number of feet. When used with `int` values, the operator "/" computes the whole number of times that one number goes into another. The expression:

    ```
    inches % 12
    ```

    computes the number of inches left over above that number of feet. The operator "%" computes the remainder when dividing one integer by another. The value of inches % 12 is between 0 and 11.
  - Create and save this program in a file named `InchesToFeet45.java`
  - Output of the program should be: "45 inches = X feet, Y inches" (where X and Y are appropriate numbers)

- **Exercises 2:**
  - The programs that you have written so far are not very useful, since they do exactly the same thing every time you run them. More useful programs would get input from the user and would give outputs that depend on what the user enters into the program.
  - These exercises are also meant to give you a chance to figure out how to use the Java API.
  - To get user input, you can use the `Scanner` class defined in the package `java.util.` You need to ***import*** this class in your programs that require user input. Try browsing through the Java API documentation on this class to figure out how to use it!
    - **Offline**: `/usr/share/doc/openjdk-6-doc/jdk/api/javadoc/doclet/index.html`
    - **Online**: `http://docs.oracle.com/javase/7/docs/api/`
  - For your Exercise 2, you should make new versions of the `InchesToFeet45` program. The new versions will use input from the user.
  - For Exercise2, start by making a copy of the `InchesToFeet45.java` with a different name: `InchesToFeet.java.` In this case, your program should request the user to type in the total number of inches.

- **Submitting your work:**
    - All source files and class files as one tar-gzipped archive.
        - When unzipped, it should create a directory with your ID. Example: **2008CS1001** (NO OTHER FORMAT IS ACCEPTABLE!!! Case sensitive!!!)
        - ***Negative marks if the TA has to manually change this to run his/her scripts!!***
    - Source / class files should include the following: (Case-Sensitive file names!!)
        - HelloWorld.java / .class [2 Points]
        - Temperature.java /.class [2 Points]
        - JavaStringExample.java / .class [5 Points]
        - InchesToFeet45.java / .class [5 Points]
        - InchesToFeet.java / .class [2 Points]
        - Circle.java / .class [2 Points]
        - TestCircle.java / .class [2 Points]
    - ***Negative marks for any problems/errors in running your programs***
    - Submit/Upload it to Moodle