CSE20 : Lab #1 – Basic Output

Welcome to CSE 20! This lab contains several parts. To ensure you get full credit, make sure you read this lab carefully and follow the instructions precisely. At this point, your TA will have explained how to logon to the lab computers, and how to start up a web browser.

Overview

The CatCourses system that you'll use in CSE 20 this fall lets us organize activities that you'll be doing inside and outside class. Notice the sidebar on the left; it organizes the activities that you'll do during class. All the assignments will be given inside CatCourses and you will turn in everything to us through it also.

(Reading) zyBooks : Chapter 1.3

Reading through the whole section on how to do print statements. Answer Activities 1.3.1, 1.3.2 and 1.3.5. You will include them in a text document and submit it in CatCourses.

Getting Oriented to the Eclipse Programming Environment

Starting Eclipse

Follow the instructions given by your TA to start up the Eclipse programming environment. Eclipse will first ask you where to maintain its "workspace" (information about the programs you create with it). If it's default choice is the `workspace` directory in your home directory, great. If not, click on the `Browse...` to navigate to your home directory and create a workspace directory there. After taking some time to get everything initialized, it will present you with a "Welcome" window. Close this window by clicking on its "X". You will now build a Java program. This is a two-step process: first you create an Eclipse *project*, then you create the Java program file.

1. Start by selecting "New:Project" from the File menu. Select the "Java Project" wizard. Name your project `day1` (with "Create project in workspace" and "Use project folder as root ..." selected).

2. Now select “New:Class” from the File menu. Provide the *class name* `FirstProgram` (notice there are no spaces in this name). Answer the question “Which method stubs would you like to create?” by selecting `public static void main (String [ ] args)` and unselecting the others. Then click “Finish”. Eclipse should provide you with a program framework.

Format of a Java program

The Java programs we will be writing early in CSE 20 each consist of *class definitions*. A definition provides the name of the class and its *variables* and *methods*. In Java, a class definition starts with the word “class” (possibly preceded by the word “public” or “private”), continues with the name of the class and the character “{” (a left brace or curly bracket), and ends with the character “}” (a right curly bracket). Inside the braces are definitions of methods and variables. A method definition similarly starts with naming information followed by a pair of braces that include the statements of the method. Braces appear inside methods as well, to group statements. Statements within such a group are separated with semicolons. Here’s an example of a “Hello world” program in Java (traditionally, the simplest program in a given language):

```
class SimpleGreeter {
    public static void main (String [ ] arguments) {
        System.out.print ("hello ");
        System.out.println ("world");
    }
}
```

At least one of the classes in a Java program must contain a method named `main`; a `main` method is what is triggered by the operating system to run the program. There are two formats for comments in a Java program. One, intended for short comments, starts with two consecutive slash characters and continues to the end of the line:

```
    // this is one kind of comment
```

The other comment format starts with the character sequence “/*” (slash followed by asterisk); it continues, perhaps over more than one line, and is terminated by the character sequence “*/” (asterisk, slash). Here’s an example:

```
    /*
      This is a
         multi-
         line comment */
```

Some commenting styles add single asterisks at the start of each line in the comment after the first, to improve readability:

```
    /*
     *  This is a
     *     multi-
     *     line comment
     */
```

Eclipse automatically provides the “boilerplate” for a new class: a couple of identifying comments, the header for a class definition, and the header for a `main` method.

Creating a Java program

Return to the Eclipse window and copy the code snippet below into the `main` method. All of these lines must be within the curly-brackets of main, so start the first line just after the line with the word “main” in it, similar to what was shown in the previous step. Don’t retype the lines; use the mouse to copy one line at a time from your browser window into the Eclipse window. (Select the text to copy with the left mouse button. Right click and select copy. Then click in the `FirstProgram.java` window at the position where the text should go, right click and select paste.)

```
System.out.print ("hello ");
System.out.println ("world");
System.out.print ("My name is Somebody!");
System.out.println ("Welcome to CSE20 at UC Merced");
```

As you copy each line, observe the automatic indenting and the highlighting of matching parentheses. A small red rectangle to the right of a line indicates a syntax (or grammar) error on that line. Once you assemble the complete program, you can test this by changing one of the occurrences of `println` to `prxntln`. A red icon to the left of a line indicates that Eclipse can correct the error; click on that icon, then double-click on the appropriate choice to fix the error.

Running a program

Once you have a syntactically correct program, you run it by selecting “Run As:Java Application” from the Run menu. When Eclipse asks you about saving your program file, click “OK”. The output from the program, which should be the message :

```
hello world
My name is Somebody!Welcome to CSE20 at UC Merced
```

It should appear in the Console window near the bottom of the Eclipse window. Rerunning the program can be done by selecting “Run Last Launched” from the “Run” menu. (This also has a keyboard equivalent.)

Try running the program several times. Notice how this program acts exactly the same each and every time.

Making changes and re-running a program

The Java programming language is nice in that as soon as we make a change to our program, the program is ready to run with the changes in place. Let's make a couple of changes to our FirstProgram.

First off, you probably noticed that the program displays exactly what is enclosed in double-quotes. Any text (letters, numbers and symbols) inside double quotes is called a "string literal" and will be used exactly as we type it. Let's make a couple of changes. I would like the first line to say "Hello, World!" instead of "hello world". See if you can figure out how to make those changes; feel free to ask the teaching assistant for help, but try it on your own first. Make sure your program runs once the new changes have been made.

Secondly, you also probably noticed that some of the lines of text have run together. That is because a string in double-quotes cannot have a "new line" or "enter" inside the quotes. Thus, we must use a different approach.

Look more closely at the 4 main lines of your program. Notice that some say "print" while others say "println"? The second of these, "println" is pronounced "print line" and its job is to display the literal text inside the quotes and then add a new line or "enter" at the ***end*** of the line. Go ahead and make a change to your program so that "My name is Somebody!" and "Welcome to CSE20 at UC Merced" appear on separate lines once the program is run.

Since your name is probably not **Somebody**, let's make that change now. Replace the word **Somebody** with your name. If your name is somebody then please leave it as is. Make sure your program runs and the output looks good before you turn in your final version of this first program.

Collaboration

Credit anyone you worked with in three different ways:

- Given help to
- Gotten help from
- Collaborated with and worked together

What to hand in

When you are done with this lab assignment, you are ready to submit your work. Make sure you have done the following ***before*** you press Submit:

- Answers to Participation Activities 1.3.1, 1.3.2 and 1.3.5 (in a text file)
- Attach your program file (the file is called FirstProgram.java). List of
- collaborators (just the names)