Laboratory Assignment № 1

Dr. Duncan, CSC 1350, Louisiana State University 09/10/2019

Working with Variables

Learning Objectives

1. More on Using the 'print' and 'println' Methods
2. Working with Variables
3. Basic Arithmetic Operations and Expressions

During the laboratory exercises in this course, you will use the Netbeans Integrated Development Environment (IDE) to write and test your program. You will also archive your source code in a zip file and submit your work via a digital dropbox for grading. In today's lab, you will experiment with the use of the *print* and *println* methods of the *PrintStream* class. You will also experiment with basic variable declaration and assignment statements. You will also compute the average of two integers to illustrate integer division and double division. Be sure to follow all the standards given on the programming style guidelines and discussed during the lectures.

The *OutputDemo* Program

Start the Netbeans IDE. Create a project using the following menu options: *File* | *New Project*. If they are not already selected, select *Java* in the *Categories* pane and *Java Application* in the *Projects* pane. Click the *Next* button at the bottom of the frame. In the *Project Name* field, type *OutputDemo*, the name of the Java class containing your program. Leave the defaults for all other fields and click the *Finish* button. Netbeans will automatically generate some starter code for your program. Modify the starter code as described below:

Listing 1: OutputDemo.java

```
package outputdemo;

/**
 * This program is written to experiment with the use of the <br>
 * 'print' and 'println' methods, variable declaration, concatenation,<br>
 * dividing two integers and dividing doubles or mixed operands<br>.
 * CSC 1350 Lab # 1<br>
 * @author Type your name here
 * @since Type the date this program was written
 */
```

```java
public class OutputDemo
{
   public static void main(String[] args)
   {

   }
}
```

---

1. Remove all Netbeans auto-generated comments from the starter code and add Javadoc header comments as shown above. Also, remove any auto-generated comments from the body of the main method. After these modifications, your program should be as shown above. Run the program. Observe that it generates no output since the main method does not have any statements.

2. Add the following lines of code to the main method:

   ```java
   System.out.println("first name: John");  //need '(', ')', and double quotes
   System.out.println(" last name: Tyler");  //the starting blank aligns ':'
   ```

   String literals must be in double quotes. A string literal is displayed exactly as enclosed in the double quotes. The argument to the *println* method must be enclosed in a pair of parenthesis. Run the program.

3. Try enclosing the string literal in single quotes and observe that this leads to syntax errors and the program will not compile. Replace the single quotes with double quotes and run the program again.

4. The // is used for commenting. Anything following // is not executed. Delete the comments from the program and run the program again. Observe that the output remains the same.

5. Replace the line

   ```
       System.out.println("first name: John");

       with

       System.out.print("first name: ");
       System.out.println("John");
   ```

   Observe that the program produces the same output. What if *print* is replaced with *println* will the output be the same? What is the difference between the *print* and *println* methods?

6. Replace

   ```
   System.out.print("first name: ");
   System.out.println("John");

   with

   System.out.print("first name:");
   System.out.println(" John");
   ```

   Run the program. Is the output the same? Revert to the original code since beginning a string literal with whitespace characters is not a good idea.

7. Recall that a variable is a named storage location. We will now experiment with declaring variables and assigning values to them. Add this line at the bottom of your main method:

   ```
   int first = 3;
   ```

   This statement allocates memory to store an integer, assigns the value 3 to that location and the name that can be used to access that memory location is *first*. Every variable has a name, value and type. The statement above declares the variable and initializes it using the integer literal 3. Run the program and the output remains the same since the program does not display the variable.

8. Modify the program by replacing

   ```
   int first = 3;

   with

   int first;
   first = 3;
   ```

   Run the program. The program does the same thing. However, the new version has a declaration statement, which allocates memory for an integer, and an assignment statement that assigns 3 to the memory that has been allocated. The previous version of the code is better. A general rule of thumb is when we know the value of a variable, we should consolidate the declaration and assignment statements. Revert to the previous version, where the declaration and initialization are consolidated.

9. Add this statement at the bottom of the main method:

   ```
   int second = 8;
   ```

   Run the program and observe the output.

10. As a rule of thumb, when we have multiple variables of the same type, rather than declare and initialize them in separate statements, we can consolidate the declaration statements by using commas to delimit the variables. Replace

    ```
    int first = 3;
    int second = 8;

    with

    int first = 3, second = 8;
    ```

11. Add the line *System.out.println(first);* at the bottom of the main method and run the program. Observe that when a variable name is used as an argument to a method, the value of the variable is printed and not its name. Now, replace this line with *System.out.println("first");* and run the program again. What is the output this time? Why?

12. Replace

    ```
    System.out.println(first);

    with

    System.out.print("first = ");
    System.out.println(first);
    ```

    Run the program. What is the last line of output? Don't display a value if what it represents is not indicated.

13. It is better to use concatenation to achieve the same output. Recall that the + operator means concatenation when at least one of its operands is a string. Replace

    ```
    System.out.print("first = ");
    System.out.println(first);

    with

    System.out.println("first = "+first);
    ```

    Observe that the left operand of + is a string literal, "first = ", and is displayed exactly as enclosed in the double quotes while the second operand *first* is a variable and its value is displayed. What if the second operand of + is changed to a string literal by enclosing it in double quotes, how would the last line of output of the program be different?

14. Several expressions may be concatenated. Replace

```
System.out.println("first = "+first);

with

System.out.println("first = " + first + ", second = "+second);
```

What would the last line of output be after this change? Run the program.

15. Whenever two integers are divided, the fractional part of the answer is discarded since the quotient of two integers is an integer. For example, 3 / 4 is 0 and 5 / 4 is 1. When at least one of the operands is a *double*, then the quotient is a *double*. For example, 3.0 / 4, 3 / 4.0 and 3.0 / 4.0 are all equal to 0.75. Make the following modification to the program. Add this line at the bottom of the main method:

```
double average = (first + second) / 2;
System.out.println("Average = " + average);
```

Run the program. What average is printed and why? Replace the 2 with 2.0 and run the program again. What average is printed? Why?

16. Add this line to the bottom of the main.

```
System.out.println("first name: " + John);
```

Observe that the program gives a syntax error. *John* the second operand of + is neither a string literal nor a variable. Since *John* is not enclosed in double quotes, the Java compiler assumes that it is a variable name. However, we have not declared a variable called *John* so that code has a syntax error and will not compile.

17. Replace

```
System.out.println("first name: " + John);

with

String firstName = "John";
System.out.println("first name: " + firstName);
```

Run the program. Why does it work this time?

18. Add these lines of code to the bottom of the main method.

```
String name = "John Tyler";
System.out.println("name: " + name);
```

What would be the last line of output of the program? Run the program.

19. Finally, add these lines of code to the bottom of the main method.

    ```
    System.out.println("3 / 5 * 5 = " + (3 / 5 * 5));
    System.out.println("3 / 5 * 5.0 = " + (3 / 5 * 5.0));
    System.out.println("3 / 5.0 * 5 = " + (3 / 5.0 * 5));
    ```

    What would be the last three lines of output of the program? Run the program.

20. In naming variables, we follow the rules for naming an identifier: names of identifiers can only have $, underscore(_) and alphanumeric characters and cannot begin with a digit. We also follow the camelcase convention when naming variables: the first word, of a variable name is in lowercase letters and all other words, if any, begin with uppercase letters. So *firstname* is not a conventional variable name. The name *firstName* follows the proper convention for naming variables. Also, use descriptive names for variables. Use 'average' rather than 'a' and 'salary' rather than 's' for your variable names. Avoid unnecessary comments in your program.

    ```
    double price = 3.5;
    int quantity = 15;
    double amount = price * quantity;  //calculating the amount
    ```

    The comment "calculating the amount" is unnecessary since anyone looking at the expression *amount = price * quantity* knows that the amount is being calculated.

Compiling and Running The Program

To run your Java application in Netbeans, select *Run* | *Run Main Project* from the Netbeans menu or, alternatively, click the green ▷ button in the menu bar. Whenever your program runs, Netbeans automatically saves it and displays the output in the output frame at the bottom of the Netbeans window.

Archiving The Program

Using windows explorer, navigate your way to the "My Document" folder. Navigate your way through the following path: *NetBeansProjects* | *outputdemo* | *src* | *outputdemo*. You will see a file called *OutputDemo.java*, your source code. Right-click the source file and choose *Send to* | *Compressed (zipped) folder* from the pop-up menu. Drag the zip file to your desktop and rename the zip file *PAWSID_lab01.zip*, where *PAWSID* is the prefix of your LSU/Tiger email address - the characters left of the @ sign. Double-click the zip file to view its contents. Verify that it contains a copy of the OutputDemo.java file. Upload the zip file to the dropbox set up on Moodle for the submission of your program.