CSCI 1301 – Programming Principles I
Georgia Southern University
Department of Computer Science
Fall 2024

Lab 2
Point Value: 20 points
Due: Friday August 23, 2024, end of lab

Objectives

1. Create an appropriate program for outputting simple messages to the console.
2. Experiment with variables, concatenation (with output), user input, and mathematical operations.
3. Archive Java source code files and other documents into a zip file.

Description

Complete the following steps as described and upload requested materials to Folio in the appropriate dropbox in a single zip file. Submit Java source code files to Gradescope.

- Students should work in groups of two for this lab. When submitting your zip file, fill out the comment header and indicate both group member names.

- Your submitted zip file should be named Lab#LastName1LastName2.zip. For instance, Lab2JonesSmith. Any requested screenshots and/or answers to questions should be included in a separate Word document.

- All programs should have a comment header. That comment header should include information similar to:

```
/**
 * File: PAssign0.java
 * Class: CSCI 1301
 * Author: Christopher Williams
 * Created on: Jun 6, 2016
 * Last Modified: Aug 16, 2018
 * Description:  Display three messages to the console
 */
```

- Before writing any code, for each problem, discuss with your partner how you plan to approach solving the given problem. Once you both agree on a way to attempt the problem, only then should code be written.

- Pay attention to names and any other requested material. Failure to include material will result in the loss of points as described below.

Lab Problems
For all problems, use inline comments to describe major actions. Ask for assistance if needed.

1. Variables and String Concatenation
   BACKGROUND/EXAMPLE: Variables can be used to store values for later use or manipulation by using the name they are given. Recall that an int data type holds an integer (whole number) value while a double stores floating-point/fractional values. In Java, we must declare variable types before using them. For instance, the value 2 could be stored in a variable named dataValue and the value 5.9 could be stored in a different variable name anotherValue.

   ```
   int dataValue = 2;
   double anotherValue = 5.9;
   ```

   Additionally, String concatenation allows us to use String literals interspersed with variables so that the value is printed in-line with the text. This is done using the + symbol, double quoted String literals, and variable names. The following code:

   ```
   System.out.println("abc " + dataValue + " def " + anotherValue);
   ```

   would print the following to the console (given the value set above):

   ```
   abc 2 def 5.9
   ```

   TASK: Start by reading this entire problem, then discussing a plan for solving this problem with your partner (it may help to write your plan down). You should not type anything until you have a set plan.

   Once the plan is agreed upon, create a new Java class named Lab02Prob01 which should be in a file named Lab02Prob01.java. Inside of that file's main method, create a comment skeleton outlining the needed steps. Once the skeleton is complete, then begin adding Java code to complete the following tasks.

   Create an integer variable named birthYear and store the value of 1997 in it using the above code as an example. Also, create a variable named currentAge and populate it by calculating the age using the current year and the birthyear variable.

   To the console, print a statement that states "`You were born in 1997 and are 27 years old.`" using the values in the birthYear and currentAge variables.

   The program's output should look like:
   ```
   You were born in 1997 and are 27 years old.
   ```

   After passing the Gradescope tests, show a working version of this program to the instructor or TA before moving on.

2. Mathematical Expressions

BACKGROUND/EXAMPLE: Recall that the ComputeExpression.java program allowed for the evaluation of math operations inside of a print statement. Statements can be grouped with parentheses within a print statement like shown:

```
System.out.println("abc, " + ((10 - dataValue) + 15) + ", def");
```

would print the following to the console (given the value used in Problem 1, ((10 - 2) + 15)):

```
abc, 23, def
```

TASK: Copy/paste your Lab02Prob01.java program into a new Java file/class named Lab02Prob02 which should be in a file named Lab02Prob02.java. Add code so that it prints some statistics and descriptive text to the console about the birthyear and currAge from Problem 1:

a. Start by reading the entire problem and make a concrete plan with your partner on how to approach a solution.
b. Create a comment skeleton outlining the needed steps.
c. Calculate and print current age based on the current year (do not worry about which month it is)
d. Calculate and print how old the person will be in 15 years
e. Calculate and print the age of someone twice their age
f. Calculate and print the age of someone half their age. HINT: divide by 2.0 (not just 2) to get a fractional value in your calculation
g. Discuss if interleaving calculations and output make sense, or if the computations should be "bundled" separately from the output.

The program's output should look like:

```
You were born in 1997 and are 27 years old.
In 15 years, you will be 42 years old.
Someone twice your age is 54 years old.
Someone half your age is 13.5 years old.
```

After passing the Gradescope tests, show a working version of this program to the instructor or TA before moving on.

3. User Input

BACKGROUND/EXAMPLE: User input in Java is done through Scanner objects which must be created before use. For instance, the following code will create a Scanner object, read in a value from the user (after they hit Enter), and store the user-entered value in a variable named userInput. Before the class header, we must also include an import java.util.Scanner; directive. A Scanner object can be reused as needed after creation; you do not need a new one to receive more user input.

```java
import java.util.Scanner;
public class ScannerExample {
    public static void main(String[] args) {
            // Create a scanner object
            Scanner input = new Scanner(System.in);

            // Prompt the user for a number
            System.out.println("Enter a whole number: ");

            // Store the number they enter in userEnteredValue
            int userEnteredValue = input.nextInt();
    }
}
```

TASK: Create a **new** class called Lab02Prob03 which should be in a file named Lab02Prob03.java. Copy and paste your existing code from Problem 2. Modify the code so that the birthYear is retrieved from user input and not set directly. All other statistics can and should stay the same. Test your new program with a few different years to make sure your code works correctly. Once you are confident in your program, submit it to Gradescope.

A sample run from Gradescope will look similar to:

```
Enter your birth year: 2005
```

The program's output should look like:

```
You were born in 2005 and are 19 years old.
In 15 years, you will be 34 years old.
Someone twice your age is 38 years old.
Someone half your age is 9.5 years old.
```

After passing the Gradescope tests, show a working version of this program to the instructor or TA before moving on.

4. Demonstrate all working programs to the instructor to receive credit for each problem. Turn in each problem inside the zip file detailed below.

5. Name the zip file Lab2LastName1LastName2.zip (e.g. Lab2JonesSmith) and submit that zip file to Folio. Both students must submit the exact same zip file for either to receive a grade.

Grade Breakdown

Demonstrate running application to instructor:	
Problem 1	5 points
Problem 2	5 points
Problem 3	5 points
Zipped source files (in Folio):	5 points

Total Possible: 20 points

Last modified: August 18, 2024