Lab 9
Loops, Debugging

The following exercises are to be completed during lab class. If you do not have time to finish during lab, they must be completed before the beginning of the following lab session.

Set-Up

- Create a new project in your Eclipse workspace named: **Lab09**
- In the *src* folder, create a package named: **edu.ilstu**
- Import the following files from T:\it168\Labs\Lab09
  - *ControlStructures.java*
  - *Problem1.java*
  - *Problem2.java*
  - *Problem3.java*
  - *Problem4.java*
  - *Problem5.java*
  - *Problem6.java*
  - *Problem7.java*
  - *Change.java*
  - *ConvertLoop.java*

Part 1

Problems 1 - 7 refer to the pre-lab control structure problems. You will check your answers by running the code in Eclipse. Fill out the last column in the Lab 09 Pseudocode Test Data document to be turned in.

You will also use the debugger to watch how the computer executes each statement and see what values are stored in the variables as the program executes. This will help you learn how the debugger works as you will find it a useful tool when your program doesn't give the correct answers. Be sure to pay attention to whether a condition is true or false and where the execution of the code goes as a result of that condition.

Problem 1

Check your answer for pre-lab problem 1

- Open the class Problem1.java
- Run the program
- You will be presented with dialog boxes to enter values for variable1 and variable2.
    - Enter 2 for variable1 and 5 for variable2 as given in the problem.
- Compare the output with what you have written in your pre-lab. Did you get it right?

Create test data that will thoroughly check this code. Put these in the table on the Lab 9 Answer Sheet.

- Run the program for each of the test cases that you created.
- Did you get your expected result?

Use the debugger to step through the code.

- Set a breakpoint on line 52 in Problem1.java
- Debug the program (follow instructions on p. 4 of the Debugger handout).
- Fill in values for each variable when you get each dialog box.
    - Your first test case should be the one given in the original problem - 2 for variable1 and 5 for variable2.
- Execution will stop on line 52 where the breakpoint was set.
- The actual code for the control structure is in a method in another class called ControlStructures. In order to see the code, you need to do a "Step Into" (see p. 6 in the handout or hover over the debug controls to find the correct button).
- It should now stop on the first statement in the called method.
    - Look at the Variables window to see the values stored in each variable
- Click on the "Step Over" button.
- It is now stopped on the if statement.
- Highlight the expression between the parenthesis, then right-click and choose Inspect.
    - This will let you see how the computer evaluates the expression.
- Click "Step Over" again and you will see it go to the statement after the if which is what should execute since the expression is true.
    - Look at the Variables view to see what is currently in each variable.
    - Notice that the "true" isn't assigned to the variable *answer* until after the statement executes.
- Click "Step Over" again.
    - Note that all statements to execute when the expression is true are completed, so it goes to the next statement after the control structure.

    - Also note that the variable *answer* in the Variables view is bold. This indicates that the value in that variable changed when executing the last statement.
- Click "Step Over" one more time to return to the original call statement in Problem1.java
- Look at the Variables view and see that *answer* still has a value of *null*. Remember that the value won't change until after the statement is executed.
- Click on "Step Over" and see that the value in answer has now changed.
- Click on the "Resume" button to complete the program.
    - You should see the final output dialog box.

Repeat all of the steps given in Problem 1 for the rest of the pre-lab control structure problems. Use the tables on the Lab 9 Answer Sheet to create your test data before doing each problem.

Problem 2

Create test data that will thoroughly check this code. The breakpoint should be set on line 52 in Problem2.java

Problem 3

Create test data that will thoroughly check this code. The breakpoint should be set on line 52 in Problem3.java

Problem 4

Create test data that will thoroughly check this code. The breakpoint should be set on line 41 in Problem4.java

Problem 5

Create test data that will thoroughly check this code. The breakpoint should be set on line 40 in Problem5.java

Problem 6

Create test data that will thoroughly check this code. The breakpoint should be set on line 40 in Problem6.java

Problem 7

Create test data that will check this code. The breakpoint should be set on line 42 in Problem7.java

There is another class, the Change class, that is involved with Problem 7. Be sure that you step into all methods that are called.

Part 2

Open the provided file `ConvertLoop.java`

Convert the following code so that it uses nested while statements instead of for statements. Place the new code at the end of the code in `ConvertLoop.java`.

```
int s = 0;
int t = 1;

for (int i = 0; i < 5; i++)
{
      s = s + i;
      for (int j = i; j > 0; j--)
      {
            t = t + (j-1);
      }
      s = s + t;
      System.out.println("T is " + t);
}
System.out.println("S is " + s);
```

Part 3

Create a class called `OddIntegers` with a main method. Write the code that will compute the sum of the first n positive odd integers. For example, if n is 5 you should compute 1 + 3 + 5 + 7 + 9. Read the value for n from the user and display the result to the screen with an appropriate label.

Part 4

Create a class called `Triangle` with a main method. Write a program that asks the user to enter the size of a triangle (an integer from 1 to 50) and validate this input. Display the triangle by displaying lines of asterisks. The first line will have one asterisk, the next two, and so on, with each line having one more asterisk than the previous line, up to the number entered by the user. On the next line write one fewer asterisk and continue by decreasing the number of asterisks by 1 for each successive line until only one asterisk is displayed. You must use nested for loops; the outside loop controls the number of lines to write, and the inside loop controls the number of asterisks to display on a line. For example, if the user enters 3, the output would be

```
*
**
***
**
*
```

(Hint: You can use more than one set of nested for loops.)

To Be Submitted

The following files should be zipped together into a file called Lab09.zip and submitted to ReggieNet by the beginning of your next lab.

- Lab09PseudocodeTestData.docx
- ConvertLoop.java
- OddIntegers.java
- Triangle.java