Lab 08: Dynamic Control Flow
CSE-020-01, Fall 2019 **Points: 20**

Overview

We will first work on some misconceptions when using if-statements. There are three types of if-statements that should be used depending on the situation. Refer to lab 6 for the description on the differences between the three structures. We will also introduce new operators that are shorthand for accumulating values into one variable.

Code Segments

We need to do three things in this sequence of code. First, read a number from the user. Add it to a running total that keeps track of the total of all the numbers entered. Then we have a running count that we decrease each time we successfully read a number from the input. The basic way of writing these three things is in three distinct java statements.

```
int num = input.nextInt();
total = total + num;
runningCount = runnningCount - 1;
```

In some situations, we do not need to keep track of the number read as the only thing we are interested in is the total. In this case, we do not need to store the number in a temporary variable. We can directly add it to the total using the following segment:

```
total = total + input.nextInt();
runningCount = runnningCount - 1;
```

This type of operation is called accumulating. We can shorthand this type of logic using **+=**. This operator will take a variable with the current value and store the new value into the variable after adding a number to the original value. In other words, read and write operations are using the same variable.

```
total += input.nextInt();
runningCount = runnningCount - 1;
```

Frequently in code segments, we will need to change the value of the variable by adding or subtracting 1. Shorthand for these operations are applying **++** and **--** to the counting variable. There are two ways of using these operators called prefix and postfix.

```
total += input.nextInt();
runningCount--;
```

The usage shown above is the postfix operation and the one shown below is the prefix operation. When they are used in isolation, there is no difference between the two cases.

```
total += input.nextInt();
--runningCount;
```

When the operators are used in larger expressions or when the "current" value has to be calculated then the differences become apparent. Prefix means the operation adds one before the value is checked. Postfix means the value is checked and then the operation adds one. The following code segment shows the effect of the unary operator applied prefix or postfix.

```
int i = 3;
i++;
System.out.println(i);    // Outputs "4"
++i;
System.out.println(i);    // Outputs "5"
System.out.println(++i);  // Outputs "6"
System.out.println(i++);  // Outputs "6"

System.out.println(i);    // Outputs "7"
```

Getting Started

After starting Eclipse, create a new project called **Lab20_8**. Import `Error7.java` and `FiveAverage.java` files from the Lab 08 assignment page into the project and load them. `FiveAverage` makes use of the new operators we discussed and introduced in this lab. It dynamically figures out how many inputs to ask from the user and calculates the average.

Part 1: Fix – Error7.java

The program asks the user to enter the 0th, 1st, 2nd, 3rd and 4th numbers for a total of 5 numbers. Then it calculates the average of all the five numbers regardless of the value. Your job is to ensure that the program then correctly checks each number and only totals up the positive numbers. Finally, the code should print out the average of only the positive input numbers.

Part 2: (Assessment) Logic Check

Create a Word document or text file named `Part2` that contains answers to the following:

1. Enter the following values as input for each of the five numbers. Check the output to make sure your `Error7` is calculating correctly. For each of the input, show the output of the program as shown below:

```
*** This program will find the average of 5 numbers ***
Please enter the 0th number: 1
Please enter the 1st number: 2
Please enter the 2nd number: 3
Please enter the 3rd number: 4
Please enter the 4th number: 5
The average of 5 numbers is 3
Average of the positive numbers is 3
```

- 1, 2, 3, 4, 5
- 5, 10, 15, 20, -25
- 5, -5, 0, 0, 0
- 5, -5, 10, -10, 0
- -1, -2, -3, -4, -5

2. What is the output of the following code sequence? (code included at the end of `FiveAverage.java`):

```
int i = 10;
i--;
System.out.println(i);
--i;
System.out.println(i);
System.out.println(++i);
System.out.println(i++);
System.out.println(i);
System.out.println(--i);
System.out.println(i--);
System.out.println(i);
if (i++ == 8)
	System.out.println("Eight");
if (++i == 9)
	System.out.println("Nine");
System.out.println("Final value " + i);
```

3. Show the change in the code above so that it also prints out `Nine`.

Part 3: Create – TenAverage.java

Copy and extend the structure of `FiveAverage.java` into a new program called `TenAverage.java` that will do the following:

- Asks the user to input a number between 0 and 10. This will be the count (number of values to average).

- Gets numbers to average from the user from 0 to count.
- Prints out the average of the numbers entered.

What to hand in

When you are done with this lab assignment, submit all your work through CatCourses.

Before you submit, make sure you have done the following:

- Attached the file named `Part2` containing answers to the assessment questions.
- Attached the fixed `Error7.java` and the newly created `TenAverage.java` files.
- Filled in your collaborator's name (if any) in the "Comments…" text-box at the submission page.

Also, remember to demonstrate your code to the TA or instructor before the end of the grace period.