**Lab Report 04**

**In this lab, we are being asked to award point to BN customers depending on how many books they buy each month, using a switch. The more books a customer has purchased, the more points should be awarded to them according to a table we are given.**

**Coding an algorithm to compute the points that should be awarded is pretty simple. If the customer has bought 0 books in a month, they get 0 points. If they bought 1 they get 5, 15 for 2, and so on. Input validation will also be needed so that a string or negative integer cannot be entered for the amount of books bought.**

**A screenshot of a computer screen

Description automatically generated**

**For these instances, the program runs as normal because all three values that were inputted are valid cases for the switch statement.**

****

****

****

**For these instances, however, the program stopped because a negative integer, decimal, and string were entered respectively.**

****

****

**A black background with white text

Description automatically generated**

**At first, I struggled to figure out how to award 60 points because I set a parameter inside the nextInt method, which did not work. Once I removed it, however, I was able to award 60 points as the default case if the other ones were not met.**

**This lab accustomed me to the switch statement, how it works, and how to effectively use it. I learned that it can oftentimes serve as a less if-else block in situations where numbers are involved or for randomly generating different outcomes.**

**Additional Questions:**

1. **Why is the keyword “break” used in a switch statement? What happens when “break” is omitted?** “Break” serves as indicator to the program that one case of the switch statement has ended and that it can move on to the next. If it is omitted, the program will continue taking any code that follows as being part of the same case, which can produce an error.
2. **What is short-circuit evaluation?** **Explain using an example.** An entire Boolean expression does not need to be evaluated to determine its value. if (value > 0 || value != 0) and value = 1, then the expression will be true because it is an OR expression whose first operand is true. On the other hand, if (value != 1 && value > 0) and value = 1, then the expression will be false because it is and AND expression whose first operand is false.
3. **Name the built-in methods that are used to test the equality of two strings?** equals and equalsIgnoreCase both are used to test this. equals is case sensitive while equalsIgnoreCase isn’t.
4. **Given the following code snippet.**

**int count = 25;**

**if (count < 0 && count >10)**

**{**

**System.out.println(“The Value of count is outside the range of 0-10”);**

**}**

**else if (count > 3 && count > 7)**

**{**

**System.out.println(“The value of count is within the range of 3-7”);**

**}**

**else**

**{**

**System.out.println(“Error”);**

**}**

* 1. **Eliminate the syntax error in the above code snippet and rewrite the code.**

int count = 25;

if (count < 0 || count > 10)

{

System.out.println(“The value of count is outside the range of 0-10”);

}

else if (count > 3 && count < 7)

{

System.out.println(“The value of count is within the range of 3-7”);

}

else

{

System.out.println(“Error”);

}

* 1. **What will the program print to the console?** “The value of count is outside the range of 0-10”