National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Computing

CS 212: Object Oriented Programming

Class: BSCS-6ABC

Lab02: Flow Control [Loops, Decisions & Arrays]

Date: 21st February, 2017

Time: 2:00pm- 4:50pm

Instructor:

Dr. Mian M. Hamayun/Dr. Anis ur Rahman

Learning Objectives

The objectives of this lab are to introduce the concepts of relational operators, if conditions, loops and arrays. In particular, the students are assumed to have basic understanding of these concepts. However, more focus is put on implementing these constructs in the Java language.

Activity #1.

Your first activity will be to test the if statements, relational and equality operators in Java.
Test the following program:

```
// Compare integers using if statements, relational operators
// and equality operators.
import java.util.Scanner; // program uses class Scanner

public class Comparison {
        // main method begins execution of Java application
        public static void main( String[] args )
        {
                // create Scanner to obtain input from command line
                Scanner input = new Scanner( System.in );

                int number1; // first number to compare
                int number2; // second number to compare

                System.out.print( "Enter first integer: " ); // prompt
                number1 = input.nextInt(); // read first number from user

                System.out.print( "Enter second integer: " ); // prompt
                number2 = input.nextInt(); // read second number from user

                if ( number1 == number2 )
                        System.out.printf( "%d == %d\n", number1, number2 );

                if ( number1 != number2 )
                        System.out.printf( "%d != %d\n", number1, number2 );

                if ( number1 < number2 )
                        System.out.printf( "%d < %d\n", number1, number2 );

                if ( number1 > number2 )
                        System.out.printf( "%d > %d\n", number1, number2 );

                if ( number1 <= number2 )
                        System.out.printf( "%d <= %d\n", number1, number2 );

                if ( number1 >= number2 )
                        System.out.printf( "%d >= %d\n", number1, number2 );

        } // end method main
} // end class Comparison
```

Activity #2.
Do as directed in the comments at the start of Calculate class.

```
public class Calculate {
        /**
        * There is a mysterious error in this program. Identify this error.
        * Is this a syntax error? or a logical one.
        */

        public static void main(String[] args) {
                int sum;
                int x;
                x = 1;          // initialize x to 1 for counting
                sum = 0;        // initialize sum to 0 for totaling

                System.out.printf( "Going to calculate the sum" );

                while ( x <= 10 ) // while x is less than or equal to 10
                {
                        sum += x; // add x to sum
                } // end while

                System.out.printf( "The sum is: %d\n", sum );
        }
}
```

Activity #3.
What does the following program print? Predict the output, before you actually try this program.

```
public class Mystery
{
        public static void main( String[] args )
        {
                int y;
                int x = 1;
                int total = 0;

                while ( x <= 10 )
                {
                        y = x * x;
                        System.out.println( y );
                        total += y;
                        ++x;
                } // end while

                System.out.printf( "Total is %d\n", total );
        } // end main
} // end class Mystery
```

Activity #4.

Try the following program and pay special attention to how arrays are created/used in Java. What is the output of this program? How else can you use this program? Think!!

```
public class Cards {
        public static void main(String[] args) {
                String [] suit = { "Clubs", "Diamonds", "Hearts", "Spades"};
                String [] rank =
                        {
                                "2", "3", "4", "5", "6", "7", "8", "9", "10", "Jack", "Queen", "King", "Ace"
                        };

                String [] deck = new String [suit.length * rank.length];

                for (int i = 0; i < suit.length; i++)
                        for (int j = 0; j < rank.length; j++)
                                deck [rank.length * i + j] = rank [j] + " of " + suit[i];

                for (int k = 0; k < (suit.length * rank.length); k++)
                        System.out.println(deck[k]);
        }
}
```

Now add the following function to the Cards class and use it.

```
private static void modify_deck(String [] deck)
{
        int N = deck.length;
        for (int i = 0; i < N; i++)
        {
                int r = i + (int) (Math.random() * (N - i));
                String t = deck[i];
                deck[i] = deck[r];
                deck[r] = t;
        }
}
```

Describe the outcome of using this function?

Task #1:
Body Mass Index Calculator: The formula for calculating BMI is

$$\text{BMI} = \frac{Weight}{Height * Height}$$

Create a BMI calculator that reads the user's weight in kilograms and height in meters, then calculates and displays the user's body mass index. Also, display the following information, using control structures, so that the user can evaluate his/her BMI.

BMI VALUES

Underweight: less than 18.5
Normal: between 18.5 and 24.9
Overweight: between 25 and 29.9
Obese: 30 or greater

Task #2:
Drivers are concerned with the mileage their automobiles get. One driver has kept track of several trips by recording the kilometers driven and liters used for each tankful.

Develop a Java application that will input the kilometers driven and liters used (both as integers) for each trip. The program should calculate and display the kilometers per liter obtained for each trip and print the combined kilometers per liter obtained for all trips up to this point.

All averaging calculations should produce floating-point results. Use class Scanner and sentinel-controlled repetition to obtain the data from the user.

Task #3:
Write a program named Deal that takes a Command-line argument N and prints N poker hands (five cards each) from a shuffled deck, separated by blank lines.

You can consult / re-used the Cards class created in Activity 4 above. An example output from your program for N=3 could be:

```
10 of Hearts, Queen of Diamonds, 3 of Clubs, 10 of Clubs, 6 of Hearts

3 of Spades, 6 of Diamonds, 9 of Hearts, 2 of Clubs, Ace of Hearts,

Ace of Spades, 7 of Diamonds, 4 of Hearts, 3 of Diamonds, 8 of Hearts,
```

Task #4:
Minesweeper: Write a program that takes 3 command-line arguments `M`, `N` and `p` and produces an `M`-by-`N` Boolean array where each entry is occupied with probability `p`. `M` is the number of rows and `N` is the number of columns in the Boolean array.

In the minesweeper game, occupied cells represent bombs and empty cells represent safe cells. Print-out the array using an asterisk for bombs and a period for safe cells. Then, replace each safe square with the number of neighboring bombs (above, below, left, right or diagonal) and print-out the solution. For example, for `M = 3`, `N = 5` and `p = 0.20`, we can get the following results:

*	*	.	.	.
.	.	.	.	.
.	*	.	.	.

*	*	1	0	0
3	3	2	0	0
1	*	1	0	0

Try to write your code so that you have as few special cases as possible to deal with, by using an (`M+2`)-by-(`N+2`) Boolean array.

Hand in
Hand in the source code from this lab at the appropriate location on the LMS system. You should hand in a single compressed/archived file named Lab_2_<Your CMS_ID. Your_NAME >.zip (without angle brackets) that contains ONLY the following files.

1) All completed java source files representing the work accomplished for this lab: ActivityOne.java; ActivityTwo.java; ActivityThree.java; ActivityFour.java; Task1.java, Task2.java, Task3.java and Task4.java. The files should contain author in the comments at the top.
2) A plain text file named **README.TXT** that includes a) author information at the beginning, b) a brief explanation of the lab, and c) any comments, or suggestions.

To Receive Credit

1. By showing up on time for lab, working on the lab solution, and staying to the end of the class period, only then you can receive full credit for the lab assignment.
2. Comment your program heavily. Intelligent comments and a clean, readable formatting of your code account for 20% of your grade.
3. The lab time is not intended as free time for working on your programming/other assignments. Only if you have completely solved the lab assignment, including all challenges, and have had your work checked off for completeness by your TA/Lab Engineer should you begin the programming/other assignments.