Lab02: Strucured Programming with Java

Introduction to :

1. **String**
2. **Control Structures** (*if, if else, switch, for, while and do while*)
3. **Arrays**

1. String

Write a Java program that enters a 10-digit string as a typical U.S. telephone number. Extract the 3-digit area code, the 3-digit "exchange," and the remaining 4-digit number as separate strings, prints them, and then prints the complete telephone number in the usual form atting. A sample run might look like this:

```
Enter 10-digit telephone number: 1234567890
You entered 1234567890
The area code is 123
The exchange is 456
The number is 7890
The complete telephone number is (123)456-7890
```

```
import java.util.Scanner;

public class TelephoneNumbers {

     public static void main(String[] args){
     Scanner keyboard = new Scanner(System.in);
     String telephone, areaCode,exchange,number;
     System.out.print("Enter 10-digit telephone number: ");
     telephone = keyboard.nextLine();
     System.out.println("You entered "+ telephone);
     areaCode = telephone.substring(0,3);
     System.out.println("The area code is "+ areaCode);
     exchange = telephone.substring(3,6);
     System.out.println("The exchange is "+ exchange);
     number = telephone.substring(6);
     System.out.println("The number is " + number);
     System.out.println("The complete telephone number is
     ("+areaCode+")"+exchange+"-"+ number);
    }
}
```

2. Control Structures

Write a Java program that generates a random year between 1800 and 2000 and then reports whether it is a leap year. A *leap year* is an integer greater than 1584 that is either divisible by 400 or is divisible by 4 but not 100. To generate an integer in the range 1800 to 2000, use

```
Random rand = new Random();

int year = rand.nextInt(200) + 1800;
```

The `rand.nextInt()` method returns the ineger between 0 until 199 (excluding 200). The transformation `1800` converts a number in the range the range `1800 ≤ year < 2000`.

```
import java.util.Random;
public class TestLeapYear {
    public static void main(String[] args){
     Random rand = new Random();
     int year = rand.nextInt(200) + 1800;
     System.out.println("The year is "+ year);
     if(year%400 == 0 || year%100 != 0 && year%4 == 0)
          System.out.print("That is a leap year..");
     else
          System.out.print("That is not a leap year..");
    }
}
```

3. Arrays

An example of simple array declared and initilized

```
public class TestArray {
    public static void main(String[] args){

     int array[] = { 32, 27, 64, 18, 95, 14, 90, 70, 60, 37 };
     System.out.println("Index       Value" );
     for ( int counter = 0; counter < array.length; counter++ )
     System.out.println( counter+"       "+ array[counter] );

    }
}
```

2D Array:

A java program to calculate the sum of each row, colum, left diagonal and right diagonal of a two-dimensional (2D) array of size MxM. for example:

ROWS	[0]	1	2	3	4	5
	[1]	6	7	8	9	10
	[2]	11	12	13	14	15
	[3]	16	17	18	19	20
	[4]	21	22	23	24	25
		[0]	[1]	[2]	[3]	[4]
		COLUMNS				

```
Sum of row 1: 15
Sum of row 2: 40
Sum of row 3: 65
Sum of row 4: 90
Sum of row 5: 115

Sum of Col 1: 55
Sum of Col 2: 60
Sum of Col 3: 65
Sum of Col 4: 70
Sum of Col 5: 75

Sum of Left Diagonal: 65
Sum of Right Diagonal: 65
```

```
public class MatrixOperations {
final static int ROWS =5;
final static int COLUMNS =5;

      public static void main(String [] args){
      int [][]m = new int[ROWS][COLUMNS];
      int rowSum, colSum, leftDiagonalSum, rightDiagonalSum;
      Random rand = new Random();
      //Assigning Random values to array elements
      for(int i=0; i<ROWS; i++)
          for (int j=0; j<COLUMNS; j++)
              m[i][j]=rand.nextInt(25)+1;
      //Calculating the sum of each row
      for(int i=0; i<ROWS; i++){
          rowSum=0;
          for (int j=0; j<COLUMNS; j++){
               rowSum+=m[i][j];
          }
          System.out.println("Sum of Row "+(i+1)+" : "+rowSum);
       }
      //Calculating the sum of each column
```

```
        for(int i=0; i<ROWS; i++){
            colSum=0;
            for (int j=0; j<COLUMNS; j++){
                 colSum+=m[j][i];
            }
            System.out.println("Sum of Col "+(i+1)+" : "+colSum);
         }
        //Calculating the sum of left diagonal
        leftDiagonalSum=0;
        for(int i=0; i<ROWS; i++){
            leftDiagonalSum+=m[i][i];
            }
            System.out.println("Sum of left diagonal  : "+leftDiagonalSum);

        //Calculating the sum of right diagonal
            rightDiagonalSum=0;
            for(int i=0; i<ROWS; i++){
            rightDiagonalSum+=m[i][(ROWS-1)-i];
            }
            System.out.println("Sum of left diagonal  : "+rightDiagonalSum);
        }
}
```

Exercises

Exercise 1 *(EmailParser.java)*

Write a Java application that needs to ask the user for her or his email address in the format firstname.lastname@bahria.edu.pk OR firstname.lastname@gmail.com. The application takes as input this email address, parses the email and replies to the user with first name, last name and host name

A sample run is given below for your convenience. User input is shown in bold

```
Please enter your email address (firstname.lastname@bahria.edu.pk):
khalid.amin@bahria.edu.pk
First Name: Khalid
Last Name: Amin
Host Name: bahria.edu.pk
```

NOTE: Use **substring(int beginIndex)** OR **substring(int beginIndex, int endIndex)** and **indexOf(String str)** methods of String.

Exercise 2 *(Roman.java)*

Write a program that converts a positive integer into the Roman number system. The Roman number system has digits I (1), V (5), X (10), L (50), C(100), D(500) and M(1000). Numbers up to **3999** are formed according to the following rules:

a) As in the decimal system, the thousands, hundreds, tens and ones are expressed separately.

b) The numbers 1 to 9 are expressed as:

1	I	6	VI
2	II	7	VII
3	III	8	VIII
4	IV	9	IX
5	V		

(An I preceding a V or X is subtracted from the value, and there cannot be more than three I's in a row.)

c) Tens and hundreds are done the same way, except that the letters X, L, C, and C, D, M are used instead of I, V, X, respectively.

Your program should take an input, such as 1978, and convert it to Roman numerals, MCMLXXVIII.

Exercise 3 *(BMICalculator.java)*

Write a program that calculates the user's body mass index (BMI) and categorizes it as underweight, normal, overweight, or obese, based on the table from the United States Centers for Disease Control:

BMI	Weight Status
Below 18.5	Underweight
18.5 – 24.9	Normal
25.0-29.9	Overweight
30.0 and above	Obese

To calculate BMI based on weight in pounds (lb) and height in inches (in), use this formula (rounded to tenths):

$$\text{BMI} = \frac{\text{mass(lb)}}{(\text{height(in)})^2} \times 703$$

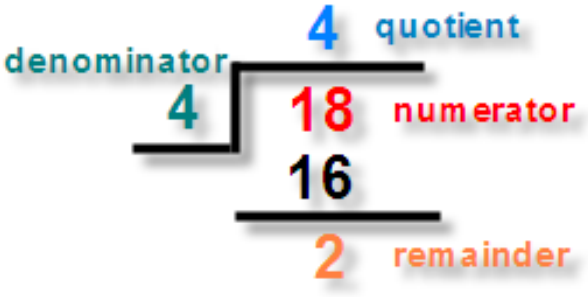
Prompt the user to enter weight in pounds and height in inches.

[SAMPLE RUN]

```
Enter your weight in whole pounds: 110
Enter your height in whole inches: 60
You have a BMI of 21.5, and your weight status is normal.
```

Exercise 4 *(Arithmatic.java)*

Write a java program to compute **quotient** and **remainder** of a number without using division ('/') operator and modulo ('%') operator.



Exercise 5 *(BloodType.java)*

Blood types are important for blood transfusion. The blood types must be matched since if not matched properly, the recipient's blood can form clots and these can lead to heart attacks, embolisms and strokes.

The following table summarize blood groups compatibility:

	Donor							
Recipient	**O-**	**O+**	**A-**	**A+**	**B-**	**B+**	**AB-**	**AB+**
O-	x							
O+	x	x						
A-	x		x					
A+	x	x	x	x				
B-	x				x			
B+	x	x			x	x		
AB-	x		x		x		x	
AB+	x	x	x	x	x	x	x	x

Write a program by using ONE and TWO dimensional arrays to help the user to clarify blood compatibility issues.

```
[Sample Run]
Enter Blood type: O+
Can Receive from: O-, O+
Can Donate to: O+, A+, B+, AB+
Do you want to continue for another value (y/n)? y
Enter Blood type: A+
Can Receive from: O-, O+, A-, A+
Can Donate to: A+, AB+
Do you want to continue for another value (y/n)? n
```

Exercise 6 *(Occurrances.java)*

Write a program to take an array and find the number of occurrences each number had. The output should be something like this:

Number	Occurrences
0	1
2	2
3	2
4	1
21	4
29	4
37	4
42	4
50	1

NOTE: **Sort** the array elements first and then calculate the frequency of each element. Use **single for-loop** to calculate and print the frequency of each element.

Exercise 7 *(Histogram.java)*

Given the following array, display its data graphically by plotting each numeric value as a bar of asterisks (*****) as shown in the diagram.

```
int[] array = {10, 19, 5, 1, 7, 14, 0, 7, 5};
```

```
Element  Value   Histogram
0        10      **********
1        19      *******************
2        5       *****
3        1       *
4        7       *******
5        14      **************
6        0
7        7       *******
8        5       *****
```

Exercise 8 *(Transpose.java)*

Suppose you have the following matrix:

2	0	4	2	6	3
9	9	1	0	4	1
7	1	2	3	7	4
2	2	2	7	1	6
1	5	8	7	4	1

Design then implement a Java program that will produce its transpose and print it along with the original one.

HINT:
The transpose of matrix A = [a_{ij}] is the *m*x*n* matrix A^T defined by $A^T = [a_{ji}]$
So, the transpose of the matrix above is:

2	9	7	2	1
0	9	1	2	5
4	1	2	2	8
2	0	3	7	7
6	4	7	1	4
3	1	4	6	1

Exercise 9 *(MatricesMultiplication.java)*

Suppose you have the following matrices:

$$M1 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{pmatrix} \text{ and } M2 = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{pmatrix}$$

Write a Java program that will calculate its product and print the resultant matrix.

$$M1 \times M2 = \begin{pmatrix} 70 & 80 & 90 \\ 158 & 184 & 210 \end{pmatrix}$$

Exercise 10 *(Scores.java)*

Write a program that calculates the total score for students in a class. Suppose the scores are stored in a three-dimensional array named **scores**. The first index in **scores** refers to a student, the second refers to an exam, and the third refers to the part of the exam. Suppose there are **7 students**, **5 exams**, and each exam has **two parts**--the multiple-choice part and the programming part. So, **scores[i][j][0]** represents the score on the multiple-choice part for the i's student on the j's exam. Your program displays the total score for each student.

```
{{7.5, 20.5}, {12, 22.5}, {22, 33.5}, {43, 21.5}, {15, 2.5}},
{{4.5, 21.5}, {12, 22.5}, {12, 34.5}, {12, 20.5}, {14, 9.5}},
{{5.5, 30.5}, {9.4, 2.5}, {13, 33.5}, {11, 23.5}, {16, 2.5}},
{{6.5, 23.5}, {9.4, 32.5}, {13, 34.5}, {11, 20.5}, {16, 7.5}},
{{8.5, 25.5}, {9.4, 52.5}, {13, 36.5}, {13, 24.5}, {16, 2.5}},
{{9.5, 20.5}, {9.4, 42.5}, {13, 31.5}, {12, 20.5}, {16, 6.5}},
{{1.5, 29.5}, {9.4, 22.5}, {19, 30.5}, {10, 30.5}, {19, 5.}}};
```