Section I: CONDITIONALS & LOOPS

The key topic for this section includes:

1. Conditional structures that select different paths through the program depending on particular conditions, such as the value of a variable.
2. Iteration structures that repeat segments of code to complete a task that has an iterative step or that inherently requires repetition.

TASKS:

1. Learn how to use control structures and loops
2. Carry out tracing with pencil and paper the steps that a computer goes through when executing a simple program
3. Trace and debug the program in IDE
4. Get familiar with methods in String class

Part A

1. A soft drink vending machine has control software that uses a `switch` statement to decide what drink to give to the buyer. Write a program with `switch` fragment to represent this (the drinks can be served by just printing their names).
    a. If button 1 is pressed, cola is served.
    b. If button 2 is pressed, lemonade is served.
    c. If button 3 is pressed, orange is served.
    d. Button 6 serves both cola and lemonade while all other buttons serve no drinks.

2. Trace the code below and figure out what will be printed for the case where: `a=1` and `b=-8`

```
int a = Integer.parseInt(args[0]);
int b = Integer.parseInt(args[1]);
int c;
if (a>0) {
      if (a<b)
            c=b-a;
      else
            c=a-b;
}
else
      c=0;
c=c*c;
System.out.print(c);
```

3. Write the following `for` loops:
    a. prints on a separate line every number from 15 to 2 (both inclusive)
    b. prints on a separate line every number from 15 to 2 (both exclusive)
    c. prints on a separate line every even number between 2 to 20 (both inclusive)

4. Trace the following code for the situations:

   a. `int x = 3;`

   b. `int x = 7;`

```
for (int i = 0; i < 5; i++){
  int j = x - i;
  if (j % 3 == 0){
      System.out.println("i: " + i + ", j: "  + j);
  }
  else{
    i++;
 }
}
```

Part B

NOTE on Eclipse

When there are multiple classes in a project, you may follow the procedure to execute a specific Java class which required input argument(s):

1. Locate the "**Package Explor**…" tab on the left.
2. Highlight the class name to choose the class to be run.
3. Click **Run−>Run…** and a dialog named "Run" will pop up.
4. In the "**Main**" tab, click button **"Search…**". A dialog will pop up. Choose the class name to be run, and click "**OK**".
5. In the **"Arguments"** tab; inside the **"Program arguments"** box, type the command line argument(s); finally, press the **"Run"** button on the bottom right corner of the dialog**.**

Exercise 1. Even or Odd?

Write code that reads in a single number from the command-line argument and prints whether the number is even or odd.

Exercise 2. Factorial (n!=1*2*3*…*n)

Write a program that takes a single command-line argument and calculates the factorial of that number.

Exercise 3. Body Mass Index (BMI) by if-else

Body Mass Index (BMI) is used to estimate the risk of weight-related problems and it is calculated as:

BMI = mass / height^2; where mass in kilograms, and height in meters. Health assessment can be estimated according to:

- BMI < 18.5 Underweight
- 18.5 ≤ BMI < 25 Normal weight
- 25 ≤ BMI < 30 Overweight
- 30 ≤ BMI Obese

Write a program to read body weight and height, calculate BMI and then print health assessment.

Exercise 4. Converting student's grade to mark range by `switch`

Input the student's grade: H, D, C, P, or F (case insensitive), output the corresponding mark range for the grade.

(Use ***charAt****(int index)* method in String class, which returns the character at the specified index.)

Exercise 5. Triangle

Write a program that takes a single command-line argument, which is a number between 1 and 9, and prints a triangle such that the base of the triangle has all numbers from 1 to that number with spaces between numbers. An example of the system in operation is:

```
> java Triangle 4
1
1 2
1 2 3
1 2 3 4
```

Exercise 6. Pyramid

Write a program that takes in an odd positive integer *n* from the command line and draw a pyramid on the screen, with the highest level having 1 block, the 2nd 3 blocks, …, and the last *n* blocks. An example in operation:

```
> java Pyramid 7
   *
  ***
 *****
*******
```

Exercise 7. Scissor-Rock-Paper Game (Advanced Question)

The rule for this popular game is: the scissor cuts the paper; the rock knocks the scissor; and the paper wraps the rock.

Write a program to simulate the game. Your program:

(1) randomly generates a number (0, 1, or 2) to represent scissor, rock, or paper;
(2) prompts the user to enter "scissor", "rock", or "paper"; and
(3) displays the computer or the user wins, loses, or draws.

Section II: ARRAYS

The key topic for this section is arrays that are data structures to store values of the same type. An array can be used to group items of the same type together and access them by their indices.

TASK

1. Learn how to create one-dimensional & two-dimensional arrays
2. Practice reading and writing elements of arrays, and traversing an array and carrying out operations on its elements.
3. Develop capacity for tracing and debugging your code.

Part A

1. Declare, create and initialize the following arrays, and each with 3 elements.
   a. A one-dimensional array of `double`s called `prices`.
   b. A one-dimensional array of `String`s called `studentNames`.
2. You are given the array of `double` values called `prices`. Write a code fragment that prints every value in the array on a separate line. Every value should be prefixed with a dollar sign (`$`).
3. What are the values of the elements of the array `myInts` after the following code runs?

```
int[] myInts = new int[6];
for (int i = 0; i < myInts.length; i++) {
  myInts[i] = 2 * i + 1;
}
for (int i = 0; i < myInts.length; i++) {
  if (i % 2 == 1) {
    myInts[i] = 2 * myInts[i - 1];
  }
}
```

Part B

Exercise 1. One-dimensional Array: Basic Statistics

Write a program to create an array that is comprised of elements of double type. The size of the array will be defined by the user input via command-line argument and the array elements are to be read from command-line arguments. Your program will calculate and print statistics of the array including the sum, the average, the minimum and the maximum value, and count how many elements are smaller or larger than the average.

Exercise 2. Two-dimensional Array

Given a two-dimensional square array of integers, with size *N* * *N*, write code to:

a. Initialize the array by command-line inputs
b. Calculate the statistics of the average, the minimum, and the maximum of the array
c. Calculates the sum of every column, and then the sum of every row
d. Calculate the sum on both diagonals.

Exercise 3. Search for the first match of a given string in a string array

Write a program to read a series of words from the command-line arguments, and find the index of the first match of a given word. For example,

```
java FirstMatch lady That young lady looks after the old lady.
```

would find the index for the first match of “lady” in the sentence “That young lady looks after the old lady”, and display the index of its first occurrence in the sentence on console, e.g.,
“The index of the first match of ‘lady’ is 2”.

Exercise 4. Calculate the average of non-zero values of an int array

Write a program to read a series of integer values from the command-line arguments, calculate the average of non-zero values in the array. For example,

```
java NonzeroAverage 1 3 0 2 4 0 6
```

will calculate the average of 5 non-zero array elements that are 1,3,2,4,6 and display their average value (i.e., 3.2) on the console.