**PROGRAMMING PATTERNS**

420-301-VA

#### **ASSIGNMENT-1**

### Instructions

* Labs’ answers must be demoed to the teacher in class.
* Assignments are individual works, except when specified by the teacher some work may be done in groups.
* You are supposed to work on this LAB during the session hours otherwise you risk falling behind.
* First, try to solve each problem without assistance except asking clarifying questions, preferably from the teacher.
* Do not look at solutions or get assistance from colleagues, online resources, or AI before spending considerable time trying to do the work from scratch yourself, some problems may require hours to solve, and you may need to look away work on another assignment and get back to the problem with a fresh eye that is normal and differs from individual to individual.
* The lab work is based on the theory sessions and a direct implementation of the concepts. Make sure you understand the concepts and can do the examples, or follow the logic, discussed in class and identify what the lab is asking you to do relative to the discussed concepts.
* Feel free to ask more questions to the teacher or look at different examples that are not the solutions to the lab from any other resource.
* Note that the due date is different for each section.
* Note that if you could not work on a lab by the deadline, you should still work on it, submit it and discuss the case with the teacher.

## Two Dimensional Array

1. This exercise involves two dimensional arrays, with repetitive and conditional logic.

Suppose that we have the scores of 6 students for 4 exams as below:

|  | **Student 1** | **Student 2** | **Student 3** | **Student4** | **Student 5** | **Student 6** |
| --- | --- | --- | --- | --- | --- | --- |
| **Exam1** | 80 | 61 | 75 | 65 | 80 | 75 |
| **Exam 2** | 75 | 70 | 60 | 65 | 81 | 75 |
| **Exam 3** | 81 | 65 | 78 | 65 | 81 | 75 |
| **Exam 4** | 78 | 69 | 75 | 78 | 80 | 75 |

Implement a program with separate methods that encapsulate the code and abstract the required logic as per the following:

* 1. Stores the grades in a two dimensional array. The array will only hold the numbers, not the labels of the columns and rows ‘student 1’, ‘student 2’, … ‘Eam1’, ‘Exam2’,....
  2. Given an exam number, calculates the average for that exam counting all students and outputs the average to the user.
  3. Given a student number, calculates the average score for that student for all exams and outputs the average to the user.

## Stack

1. Stack Java collections

What is a Stack?

A stack can be imagined as a pile of objects or elements one on top of each other, the last element we put on top of the stack will be the first element we remove, this is referred to as last-in-first-out (LIFO) mechanism.

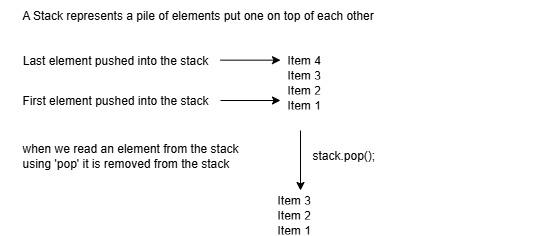
| Stack of books |
| --- |



A Stack data structure allows the storage of related elements and then the retrieval of elements based on the last-in-first-out (LIFO) rule. Which means that the last element added to the stack will be returned when the programmer retrieves the first element of the stack, thus the elements are read in reverse order to which they were stored in.

When we add an element to a stack, the operation is referred to as ‘push’, we push an element onto the stack.

When we read an element from the stack, the operation is referred to as ‘pop’, noting that when we ‘pop’ an element from the stack it is removed from the stack and it is the last element that was stored onto the stack.



Java already supports a Stack data structure through the Stack collection class.

#### Stack declaration

| **Stack<String> stack = new Stack<String>();** |
| --- |

* 1. Implement a program that uses a stack to store string values inputted by the user and then prints them in reverse order.

Custom array based stack

As an exercise, you will implement a custom array based stack class.

* 1. Implement a class called CustomArrayStack.
  2. Implement a single dimensional array as a private property of the CustomArrayStack class.
  3. Implement a method called push that takes a value and adds it at the beginning of the array.
  4. Implement a method called pop that removes and returns the first element of the array.
  5. Implement the client (main) class that uses the CustomArrayStack to add some sample data using the push method and then prints out all the elements using the pop method.