# Overview

In this lab, you will create 2 classes named Stack.java and LabStack.java from scratch. In Stack.java you will implement your own version of the java Stack class. A Stack is a Last In First Out (LIFO) data structure. It supports two basic operations called push and pop. The push operation adds an element at the top of the stack, and the pop operation removes an element from the top of the stack. You will be implementing the push(), pop(), isEmpty(), and peek() methods. LabStack.java uses Stack.java; Here you will utilize the methods from Stack.java.

# Objectives

* Practice creating a class
* Creating and using static methods
* Practice with Java Stacks
* Applying test cases to your program

# Steps

1. Create a folder on your local machine for your Java program, you can name it whatever you like
2. Start Visual Studio Code (VS Code)
3. In VS Code, Open that newly created folder.
4. Now it’s time to start coding:
   1. Create a class called Stack
   2. Create the following in Stack.java:
      1. An array of ints set to size 10 to represent the Stack, an int variable for the index
      2. A constructor that sets the index to -1
      3. A method called **isEmpty** (returns a boolean) that has no parameters. This method returns true if the Stack is empty and false if the Stack is not empty.

Example (results in red)

| Stack s = new Stack();  System.out.println(s.isEmpty()); |
| --- |

true

* + 1. A method called **push**(returns a Boolean) that takes an int as a parameter. This method pushes, or adds, the int to the stack if there is available space. Returns true if it got added successfully and false if wasn't able to be added to the stack

Example (results in red)

| Stack s = new Stack();  System.out.println(s.push(10));  System.out.println(s.push(20)); |
| --- |

true

true

* + 1. A method called **pop** that takes has no parameter and returns an int. This method removes the next int from the stack and returns it.

Example (results in red)

| Stack s = new Stack();  s.push(10);  s.push(20);  s.push(30);  System.out.println(s.pop());  System.out.println(s); |
| --- |

30

20, 10

* + 1. A method called **peek** that has no parameter. This method returns the next int from the Stack without removing, or popping, it from the Stack

Example (results in red)

| Stack s = new Stack();  s.push(10);  s.push(20);  s.push(30);  System.out.println(s.peek());  System.out.println(s); |
| --- |

30

30, 20, 10

* + 1. Override the **toString()** method to be able to print the values in the stack separated by a comma. You should **NOT** pop() any of the values

Example (results in red)

| Stack s = new Stack();  s.push(10);  s.push(20);  s.push(30);  System.out.println(s); |
| --- |

30, 20, 10

* 1. Create a class called LabStack
  2. Create the following in LabStack:
     1. A **static** method called **popN** that takes a stack **st** and an integer **n** as parameters. This method pops **n** items from **st** and returns an **ArrayList<Integer>** that contains all the values that were popped.

Example (result in red)

| Stack st = new Stack();  st.push(10);  st.push(20);  st.push(30);  ArrayList<Integer> result = LabStack.popN(st, 2);  System.out.println(result); |
| --- |

[30, 20]

* + 1. A **static** method called **popAll** that takes a stack **st** as a parameter. This method pops all items from **st** and returns an **ArrayList<Integer>** that contains all the values that were popped.

Example (output in red)

| Stack st = new Stack();  st.push(10);  st.push(20);  st.push(30);  ArrayList<Integer> result = LabStack.popAll(st);  System.out.println(result); |
| --- |

[30, 20, 10]

* + 1. A **static** method called **reverse** that takes an int array **arr** as a parameter. This method creates a local stack variable, puts all values from **arr** into the stack, then pops them to store them in a new int array. It returns the new int array, which should contain all the values from **arr** reversed.

Example (result in red)

| int[] vals = new int[]{10, 20, 30, 40, 50};  int[] rev = LabStack.reverse(vals);  System.out.println(Arrays.toString(rev)); |
| --- |

[50, 40, 30, 20, 10]