# Assignment 4 – Part 1: Stacks

## Ryan Brinson

## 9/15/2023

## Output:

A screenshot of a computer program

Description automatically generated

## Code:

// Name: Ryan Brinson

// Class: CS 3305 W04

// Term: Fall 2023

//  Instructor:  Carla McManus

//  Assignment:  04-Part-1-Stacks

import java.util.Scanner;

public class A4P1 {

    static final int STACKA = 4;

    static final int STACKB = 5;

    public static void main(String[] args) {

        Array arr = new Array(STACKA,STACKB);

        Scanner input = new Scanner(System.in);

        // Ask the user for values

        arr = GetValues(input, arr);

        // Test the pop function of the Array

        arr = TestArray(arr);

        // Get new values to put back in the stacks

        arr = GetValues(input, arr);

        // Print those values to the user

        PrintArray(arr);

        // Pops the stacks and then prints out the results

        PopTheTop(arr);

        PrintArray(arr);

    }

    // Pops the top of both stacks

    public static Array PopTheTop(Array ar){

        System.out.println("\nPop the Top of both stacks");

        ar.pop\_a();

        ar.pop\_b();

        return ar;

    }

    // Print the current values in each stack

    public static void PrintArray(Array ar){

        System.out.println("\nValues in stack a: ");

        ar.print\_a();

        System.out.println("\nValues in stack b: ");

        ar.print\_b();

    }

    // Method to collect values to put into the array

    public static Array GetValues (Scanner in, Array ar){

        System.out.println("\nEnter 5 values for the Array");

        // In this cause the total number of values in the array to be

        // filled is 5. 3 into stack a and 2 into stack b.

        for (int i = 0; i < 3; i++) {

            ar.push\_a(in.nextInt());

        }

        for (int i = 0; i < 2; i++) {

            ar.push\_b(in.nextInt());

        }

        // Return ar to the original array to update the values

        return ar;

    }

    // Method to test that pop in both a and b is working properly

    public static Array TestArray(Array ar){

        // Assign a test arrays to what is stored in the stacks

        int[] test\_a = ar.return\_a();

        int[] test\_b = ar.return\_b();

        // Temp value to hold temporary values

        int temp;

        // Initialize two values to the same as where the

        // current stack pointers are

        int i = ar.get\_top\_a();

        int j = ar.get\_top\_b();

        // While stack a is not empty

        while (!ar.is\_empty\_a()){

            // Pop the top of A into temp

            temp = ar.pop\_a();

            // Test that it's in fact the right value

            // It they are not equal, print out an error

            if (test\_a[i] != temp)

                System.err.println("Error: " + test\_a[i] + " != " + temp);

            // If it is the right value output the two side by side

            else{

                System.out.println("Test: " + test\_a[i] + " Value: " + temp);

                i--;

            }

        }

        // The mothod moves on to check stack b

        while (!ar.is\_empty\_b()){

            // Pops the top of stack B into temp

            temp = ar.pop\_b();

            // Checks if the values are equal

            // If they aren't, prints out error

            if (test\_b[j] != temp)

                System.err.println("Error: " + test\_b[j] + " != " + temp);

            // If they match, print the results

            else{

                System.out.println("Test: " + test\_b[j] + " Value: " + temp);

                j--;

            }

        }

        // Return ar to update the Array in main

        return ar;

    }

}

class Array {

    private int[] StackA;

    private int[] StackB;

    private int top\_a;

    private int top\_b;

    Array(int stackASize, int stackBSize){

        // Stack default value is -1, representing an empty stack

        top\_a = -1;

        top\_b = -1;

        StackA = new int[stackASize];

        StackB = new int[stackBSize];

    }

    // ----- Print Stack -----

    public void print\_a(){

        for (int i = 0; i <= top\_a; i++){

            System.out.println(StackA[i]);

        }

    }

    public void print\_b(){

        for (int i = 0; i <= top\_b; i++){

            System.out.println(StackB[i]);

        }

    }

    // ----- is Full ----

    public boolean is\_full(){

        // Check if both stacks have reached there limit

        return (top\_a >= StackA.length)

                &&

                (top\_b >= StackB.length);

    }

    // ----- Is Empty -----

    public boolean is\_empty\_a(){

        return top\_a < 0;

    }

    public boolean is\_empty\_b(){

        return top\_b < 0;

    }

    // ----- Push -----

    public void push\_a(int n){

        if(top\_a == (StackA.length - 1)) System.out.println("Stack full");

        else {

            StackA[top\_a + 1] = n;

            top\_a++;

        }

    }

    public void push\_b(int n){

        if(top\_b == (StackB.length - 1)) System.out.println("Stack full");

        else {

            StackB[top\_b + 1] = n;

            top\_b++;

        }

    }

    // ----- Pop -----

    public int pop\_a(){

        int temp = StackA[top\_a];

        StackA[top\_a] = 0;

        top\_a--;

        return temp;

    }

    public int pop\_b(){

        int temp = StackB[top\_b];

        StackB[top\_b] = 0;

        top\_b--;

        return temp;

    }

    // ----- Return Stack A and B -----

    public int[] return\_a(){

        return StackA.clone();

    }

    public int[] return\_b(){

        return StackB.clone();

    }

    // ----- Return Stack Pointer -----

    public int get\_top\_a(){

        return top\_a;

    }

    public int get\_top\_b(){

        return top\_b;

    }

}