Assignment 4 – Part 1: Stacks

Ryan Brinson

9/15/2023

Output:

```
rbrinson2@DESKTOP-U8KJ4OP:~/Documents/CS3305/A4$ java A4P1

Enter 5 values for the Array
1
2
3
4
5
Test: 3 Value: 3
Test: 2 Value: 2
Test: 1 Value: 1
Test: 5 Value: 5
Test: 4 Value: 4

Enter 5 values for the Array
6
7
8
9
0

Values in stack a:
6
7
8

Values in stack b:
9
0

Pop the Top of both stacks

Values in stack a:
6
7

Values in stack b:
9
rbrinson2@DESKTOP-U8KJ4OP:~/Documents/CS3305/A4$
```

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;
    }


}
```