LAB_ANP_C6339_CLASSES

Student ID: AF0339439

YELLA UDAY KUMAR

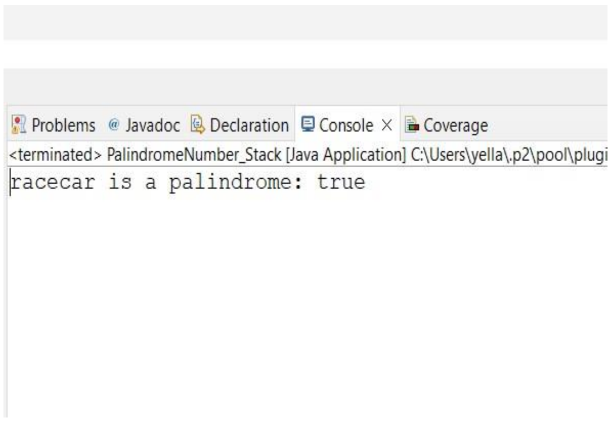
Assignment 1:

Write a Java program to check if a given string is a palindrome. Use a stack to help you compare characters from the beginning and end of the string. Do not use any String functionalities to reverse the String. Use Stack data structure. You can use (A string is called Palindrome if the reverse of the string is the same as the original string. Example: “racecar”.)

Program:

```java
package Stack_Data_Structure;

import java.util.Stack;

public class PalindromeNumber_Stack {
    public static boolean isPalindrome(String input) {
        Stack<Character> stack = new Stack<>();

        // Push each character onto the stack
        for (char c : input.toCharArray()) {
            stack.push(c);
        }

        // Check if characters match while popping from the stack
        for (char c : input.toCharArray()) {
            char poppedChar = stack.pop();
            if (c != poppedChar) {
                return false; // Characters don't match, not a palindrome
            }
        }

        return true; // All characters match, it's a palindrome
    }

    public static void main(String[] args) {
        // Example palindrome and non-palindrome strings
        String palindromeString = "racecar";


        // Check if each string is a palindrome and print the result
        System.out.println("" + palindromeString + " is a palindrome: " +
isPalindrome(palindromeString));

    }
}
```

Output:


Problems Javadoc Declaration Console Coverage

```
<terminated> PalindromeNumber_Stack [Java Application] C:\Users\yella\.p2\pool\plugi
racecar is a palindrome: true
```

Assignment 2 :

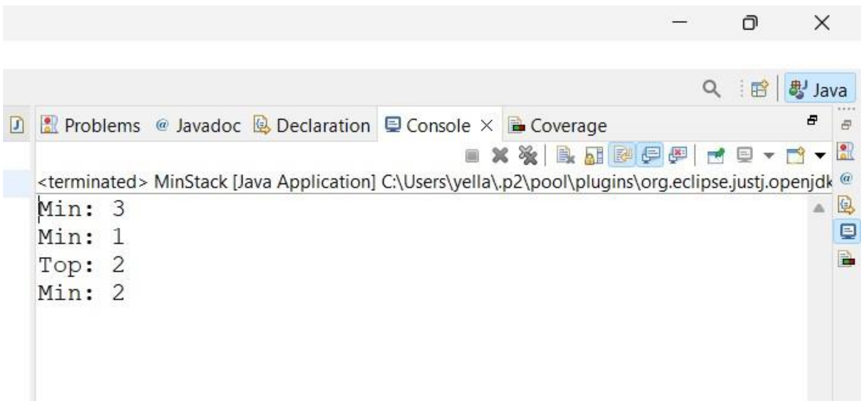
Create a Java class called MinStack of Integers that supports standard stack operations (push, pop, top, and isEmpty) and also provides a getMin method that returns the minimum element in the stack.

(Hints: Keep another stack that will hold the minimum value on the top of the stack.

Program:

```
}
import java.util.Stack;
public class MinStack {
    private Stack<Integer> mainStack;
    private Stack<Integer> minStack;

    public MinStack() {
        mainStack = new Stack<>();
        minStack = new Stack<>();
    }

    public void push(int x) {
        mainStack.push(x);

        if (minStack.isEmpty() || x <= minStack.peek()) {
            minStack.push(x);
        }
    }

    public void pop() {
        if (!mainStack.isEmpty()) {
            int poppedValue = mainStack.pop();

            if (poppedValue == minStack.peek()) {
                minStack.pop();
            }
        }
    }

    public int top() {
        if (!mainStack.isEmpty()) {
            return mainStack.peek();
        }
        throw new IllegalStateException("Stack is empty");
    }

    public int getMin() {
        if (!minStack.isEmpty()) {
            return minStack.peek();
        }
        throw new IllegalStateException("Stack is empty");
    }

    public boolean isEmpty() {
        return mainStack.isEmpty();
    }

    public static void main(String[] args) {
        MinStack minStack = new MinStack();

        minStack.push(3);
        minStack.push(5);
        System.out.println("Min: " + minStack.getMin());

        minStack.push(2);
        minStack.push(1);
        System.out.println("Min: " + minStack.getMin());

        minStack.pop();
        System.out.println("Top: " + minStack.top());
        System.out.println("Min: " + minStack.getMin());
    }
```

Output:



```
<terminated> MinStack [Java Application] C:\Users\yella\.p2\pool\plugins\org.eclipse.justj.openjdk
Min: 3
Min: 1
Top: 2
Min: 2
```