St. Joseph's University, NY
COM 210 - Algorithms and Data Structures

Professor: Andrew Lane
Student name: Armando Escobar Castillo, Joseph Giglio
Classwork lab #: 3
Date: 4/21/2023

Abstract: This project is about understanding stacks, their data structure, and their algorithms and implementing them in an exercise where we are challenged to use a stacks to check if parentheses are balanced in a string.

Introduction: To do this challenge, we decided to code the push(), pop(), isEmpty(), and isFull() methods of a Stack. Then, to solve the problem of checking if parentheses are balanced in a string, we coded the isBalanced() method.

Procedure: We created a class called MyStack, where we coded and implemented the algorithms of the Stack data structure. First, we created three instance variables:

int maxSize //Stores the length of the stack.
int top //Stores the index of the top element in the stack.
char[] charArray //Stores the elements in the stack.

Then we created the following methods:

```
public static void push(char newChar) {
    top++; // increment top
    charArray[top] = newChar; // add the new character to the top of
the stack
  }
```

```
 public static void pop() {
    top--; // decrement top
  }
```

```
public static boolean isEmpty() {
    return (top == -1); // return true if top is -1, indicating that
the stack is empty
  }
```

```
public static boolean isFull() {
    return (top == maxSize - 1); // return true if top is equal to
maxSize-1, indicating that the stack is full
  }
```

Then for the purpose of this challenge, we needed a method that could check if a string is balanced. The logic algorithm of this method is the following: Take an input string from the user, then iterate through each character and check if it is an opening parenthesis “(“; if it is, then push the parenthesis into the stack. If the character is a closing parenthesis “)” pop a parenthesis from the stack. After iterating through the entire string, if the stack is empty, it means that the parentheses are balanced; if something is still remaining in the stack, then the parentheses are not balanced.

```
public static boolean isBalanced(String myString) {

    for (int i = 0; i < charArray.length; i++) { // loop through each
character in myString

        if (myString.charAt(i) == '(') { // if the current character
is an opening parenthesis
          push(myString.charAt(i)); // push it onto the stack

        } else if (myString.charAt(i) == ')') { // if the current
character is a closing parenthesis
          if (isEmpty()) { // check if the stack is empty
            return false; // if it is empty, return false because
there is no matching opening parenthesis
          }
          pop(); // pop an element from the stack because we have
found a matching pair of parentheses
        }
    }

    return isEmpty(); // return true if the stack is empty,
indicating that all pairs of parentheses have been matched
  }
```

Implementation: We created a scanner object to input from the command line. Then we created a string variable to store the user input string. Then, we initialized the character array from the instance variables and set it to the length of the input string. Following that, we initialized the top of the array at -1 and the maximum size at the length of the input string. Then we defined a simple if statement using the method to check if the parentheses are balanced with the following logic: If isBalanced(input) == true, print “The parentheses are balanced”; otherwise, print “the parentheses are not balanced.

Conclusion: This lab has helped us understand in detail the logical structure of Stacks by manually coding the necessary methods to use this data structure. Then we were able to use Stacks to solve a real-world problem by checking if the parentheses in a string are balanced. Overall, this project has deepened our understanding of Stacks and their applications.