Objective(s):

- a. To practice array-based data structure.
- b. To be familiar with Reverse Polish Notation creation process.

Task 1. Implement MyStack in \Lab05\pack. The class has the following methods:

```
public void push(T d)
public T pop()
public T peek()
public int size()
public boolean isEmpty()
public String toString()
```

```
package Lab05.pack;
import java.util.ArrayList;
public class MyStack<T> { // extends ArrayList<T> {
/* extends would expose MyStack object for calling
any of the ArrayList's methods */
    private ArrayList<T> items = new ArrayList<>();
    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append("[");
        for (int i = size() - 1; i > 0; i--)
            sb.append(items.get(i) + ", ");
        if (items.size() > 0)
            sb.append(items.get(∅));
        sb.append("]");
        return sb.toString();
    }
}
```

Task 2 Implement MyRPN.java which contains double computeRPN(String postfixString)

Notice

- how to process each token of the StringTokenizer st.
- how regular
 expression of Pattern
 pattern is used.
- you may supply your customed postfix string.

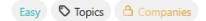
```
package Lab05.pack;
import java.util.StringTokenizer;
import java.util.regex.Pattern;
public class MyRPN {
    private static Pattern pattern = Pattern.compile("-
?\\d+(\\.\\d+)?");
    public static boolean isNumeric(String strNum) {
        if (strNum == null)
            return false;
        return pattern.matcher(strNum).matches();
    }
    public static double computeRPN(String rpn) {
        /* your code */
        return stack.pop();
    }
}
```

```
static void demo_1() {
    String toBeRPN = "8 5 - 4 2 + 3 / *";
    StringTokenizer st = new StringTokenizer(toBeRPN);
    int i = 0;
   String t = "";
    while (st.hasMoreTokens()) {
        t = st.nextToken();
        if (MyRPN.isNumeric(t))
            System.out.println("Token " + i++ + " = " + t);
        else
            System.out.println("Token " + i++ + " = " + t + "
                                              is an operator");
    }
static void demo 2() {
   String postfixString = "8 5 - 4 2 + 3 / *";
    System.out.println(postfixString + " = " +
                              MyRPN.computeRPN(postfixString));
}
```

Task 3 Solve below leetcode problem.

844. Backspace String Compare

Solved **⊘**



Given two strings s and t, return true if they are equal when both are typed into empty text editors. ** means a backspace character.

Note that after backspacing an empty text, the text will continue empty.

Example 1:

```
Input: s = "ab#c", t = "ad#c"
Output: true
Explanation: Both s and t become "ac".
```

Example 2:

```
Input: s = "ab##", t = "c#d#"
Output: true
Explanation: Both s and t become "".
```

Example 3:

Submission: MyStackA XXYYYY.java, MyRPN XXYYYY.java and Solution XXYYYY.java

Due date: TBA