Objective(s):

a. To practice array-based data structure.

b. To be familiar with Reverse Polish Notation creation process.

Task 1. Create directory named pack5_StackQueue. Implement your stack using array called MyStackA.java using given template (Do not expand from MyArray.java previously created). The file will be inside package code.

MyStackA has the following methods :

- void push(double d)
- double pop()
- double top()
- boolean isFull()
- boolean isEmpty()

```
package code;

public class MyStackA {
    int MAX_SIZE=100;
    double stack[] = new double[MAX_SIZE];
    int top = 0;

    /* your code here */

    public String toString() {
        StringBuffer sb = new StringBuffer();
        sb.append("top->");
        for(int i=top-1; i>=0; i--) {
            sb.append("[");
            sb.append(stack[i]);
            sb.append("]->");
        }
        sb.append("bottom");
        return new String(sb);
    }
}
```

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 when calling L5_RPN_Main i.e., java L5_RPN_Main "3 1 – 4 5 + *".

Else the default postfix string would be "8 5 - 4 2 + 3 / *"

```
package code;
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) {
        MyStackA stack = new MyStackA();}
       /* your code */
  }
}
```

```
import code.MyStackA;

public class L5_RPN_Main {
    private static void testTokenizer(String toBeRPN) {
        StringTokenizer st = new StringTokenizer(toBeRPN);
        int i = 0;
        String t = "";
        while (st.hasMoreTokens()) {
          t = st.nextToken();
          if (MyRPN.isNumeric(t))
            println("Token " + i++ + " = " + t);
          else
            println("Token " + i++ + " = " + t
                                  + " is an opearator");
        }

    }
    public static void main(String[] args) {
        // 3 1 - 4 5 + *
        String postfixString = "8 5 - 4 2 + 3 / *";
        if (args.length > 0)
            postfixString = args[0];
        //testTokenizer(postfixString);
        println(postFixString);
        println("= "+ MyRPN.computeRPN(postFixString));
  }
```

Submission:

MyStackA_XXYYYY.java and MyRPN_XXYYYY.java

Due date: TBA