May 26th, 2020

# Exercises for Programming Practice 2

Today’s topic is stacks, queues and deques. It is convenient to use “ArrayDeque” to handle them. Java ArrayDeque is described in the document “ArrayDeque”.

After understanding the document “ArrayDeque”, please challenge “Exercise 3”, “Exercise 4” and “Exercise 5”.

Note:

- ✓ Do not create “module-info.java”, when you create a Java Project.
- ✓ Do not set Package name in the window “New Java Class”.
- ✓ Do not use the title of exercise for “contents of the program”.
  Think about “contents of the program” yourself.

The deadline for submitting the programs is 18:00 on May 28th, 2020.

Exercise 3 (file name “Exercise3.java”)

Create a Java program named “Exercise3” to check the function of the *queue* using ArrayDeque. Use addLast method of ArrayDeque to “enqueue”, and removeFirst method to “dequeue”.

Make method “public void queueTest()” to do the following.

(1) enqueue(“AA”)

(2) enqueue(“BB”)

(3) enqueue(“CC”)

(4) enqueue(“DD”)

(5) display *queue*.

(6) dequeue() and display the return value.

(7) enqueue(“EE”)

(8) display *queue*.

The outline of the program is shown in Fig. 1.

```
/* comments */
import java.util.ArrayDeque;

public class Exercise3 {
    ArrayDeque<String> queue = new ArrayDeque<String>();

    public static void main(String[] args) {
        Exercise3 ex3 = new Exercise3();
        ex3.queueTest();
    }

    public void queueTest() {
        // Complete this part
    }
}
```

Fig. 1 The outline of the program “Exercise3.java”

*Expected Output:*

```
[AA, BB, CC, DD]
AA
[BB, CC, DD, EE]
```

Exercise 4 (file name “Exercise4.java”)

Create a Java program named “Exercise4” to check the function of the *stack* using ArrayDeque. Use push method of ArrayDeque to “push”, and pop method to “pop”.

Make method “public void stackTest()” to do the following.

(1) push (“AA”)

(2) push (“BB”)

(3) push (“CC”)

(4) display *stack*.

(5) pop() and display the return value.

(6) push (“DD”)

(7) display *stack*.

The outline of the program is shown in Fig. 2.

```
/* comments */
import java.util.ArrayDeque;

public class Exercise4 {
    ArrayDeque<String> stack = new ArrayDeque<String>();

    public static void main(String[] args) {
        Exercise4 ex4 = new Exercise4();
        ex4.stackTest();
    }

    public void stackTest() {
        // Complete this part
    }
}
```

Fig. 2 The outline of the program “Exercise4.java”

*Expected Output*:

```
[CC, BB, AA]
CC
[DD, BB, AA]
```

Exercise 5 (file name “Exercise5.java”)

Create a Java program named “Exercise5” to test for pars of matching delimiters, such as

Parentheses: ‘(’ and ‘)’

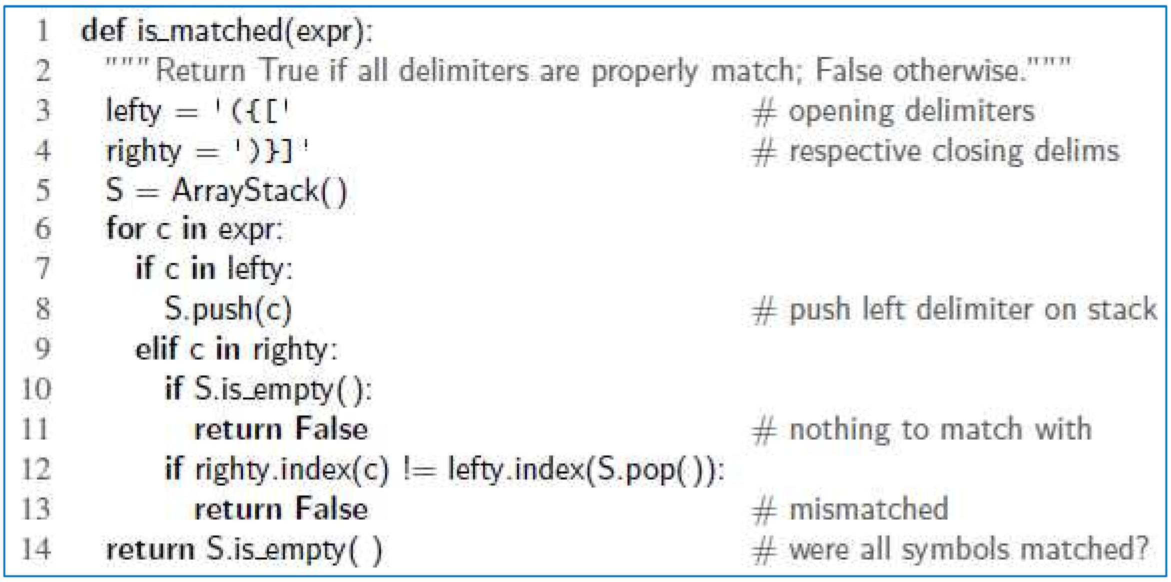
Braces: ‘{’ and ‘}’

Brackets: ‘[’ and ‘]’

The opening symbols and closing symbols are recorded in ArrayList<Character> left and ArrayList<Character> right respectively in the constructor.

In the “check” method, the argument string is converted to a character array using String method “toCharArray”, and “is_matched” method is called.

The algorithm of “is_matched” is shown in Fig. 3.



```
 1 def is_matched(expr):
 2   """Return True if all delimiters are properly match; False otherwise."""
 3   lefty = '({['                                  # opening delimiters
 4   righty = ')}]'                                 # respective closing delims
 5   S = ArrayStack()
 6   for c in expr:
 7     if c in lefty:
 8       S.push(c)                                  # push left delimiter on stack
 9     elif c in righty:
10       if S.is_empty():
11         return False                             # nothing to match with
12       if righty.index(c) != lefty.index(S.pop()):
13         return False                             # mismatched
14   return S.is_empty()                            # were all symbols matched?
```

Fig. 3 The algorithm for matching delimiters in as arithmetic expression.

The outline of the program is shown in Fig. 4. The third and fourth lines of the algorithm are implemented in constructor of Exercise5.java”.

Complete method “public Boolean is_matched(char[] expr)” according the algorithm in Fig. 3.

Tips:

(a) Use ArrayDeque deque for ArrayStack on line 5 in Fig. 3.

(b) Use “contains” method of ArrayList to check if a character is the opening symbols or the closing symbols.

(c) Use “indexOf” method of ArrayList to check the position of ArrayList to know the kind of delimiter.

```
/* comments */
import java.util.ArrayDeque;
import java.util.ArrayList;

public class Exercise5 {
    ArrayDeque<Character> deque = new ArrayDeque<Character>();
    ArrayList<Character> left = new ArrayList<Character>();
    ArrayList<Character> right = new ArrayList<Character>();

    public static void main(String[] args) {
        Exercise5 ex5 = new Exercise5();
        ex5.check("(){([()])}");
        ex5.check("([()])}");
    }

    public Exercise5() {
        left.add('(');
        left.add('{');
        left.add('[');
        right.add(')');
        right.add('}');
        right.add(']');
    }

    public void check(String str) {
        char[] expr = str.toCharArray();
        if(is_matched(expr)) {
            System.out.println("Correct: " + str);
        } else {
            System.out.println("Incorrect: " + str);
        }
    }

    public boolean is_matched(char[] expr) {
        // Complete this part
    }
}
```

Fig. 4 The outline of the program “Exercise14.java”

*Expected Output:*

```
Correct: (){([()])}
Incorrect: ([()])}
```