



**COMSATS UNIVERSITY ISLAMABAD,  
ABBOTTABAD**

**Software Testing**

**Lab Mid**

<b>IntelliJ IDE</b>
---------------------

***Submitted by:***

Arfah Ali FA21-BSE-080

***Submitted to:***

Sir Mukhtiar Zamin

## Algorithm No. 02

I have been assigned with the task of solving an Algorithm 2 which was an algo to check whether the given string is a palindrome or not. I have sketched a CFG, paths regarding that CFG and then draw some testcase that pass through those paths. All of them were passed through the given algorithm. I have done this task using IntelliJ IDE.

---

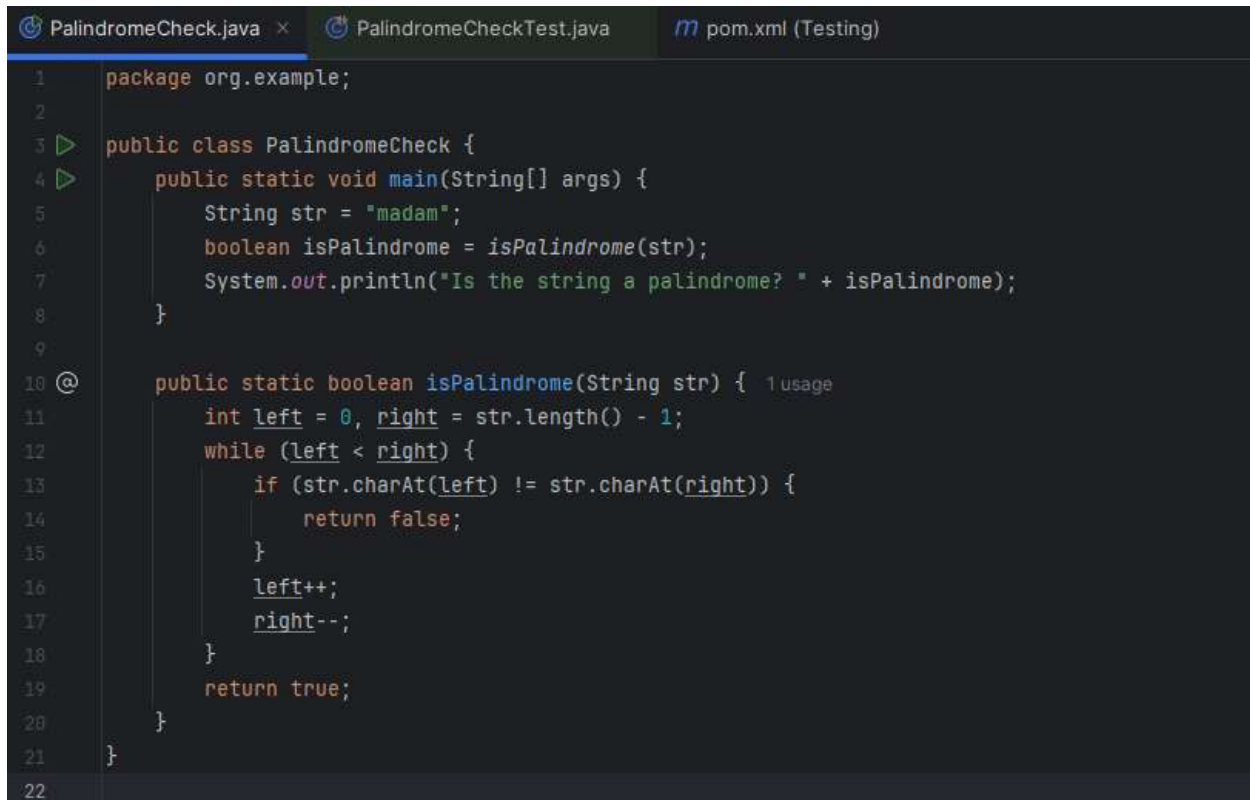
- 2. Check if a String is a Palindrome: Compares characters from the start and end moving towards the center to check for equality.

```
public class PalindromeCheck {
    public static void main(String[] args) {
        String str = "madam";
        boolean isPalindrome = isPalindrome(str);
        System.out.println("Is the string a palindrome? " +
isPalindrome);
    }

    public static boolean isPalindrome(String str) {
        int left = 0, right = str.length() - 1;
        while (left < right) {
            if (str.charAt(left) != str.charAt(right)) {
                return false;
            }
            left++;
            right--;
        }
        return true;
    }
}
```

---

## PalindromeCheck.java



```
1 package org.example;
2
3 public class PalindromeCheck {
4     public static void main(String[] args) {
5         String str = "madam";
6         boolean isPalindrome = isPalindrome(str);
7         System.out.println("Is the string a palindrome? " + isPalindrome);
8     }
9
10    public static boolean isPalindrome(String str) { 1 usage
11        int left = 0, right = str.length() - 1;
12        while (left < right) {
13            if (str.charAt(left) != str.charAt(right)) {
14                return false;
15            }
16            left++;
17            right--;
18        }
19        return true;
20    }
21 }
22
```

## PalindromeCheckTest.java

```
PalindromeCheck.java  PalindromeCheckTest.java x  pom.xml (Testing)
1  package org.example;
2  import junit.framework.TestCase;
3  import org.junit.Test;
4  public class PalindromeCheckTest extends TestCase {
5      @Test
6      public void test_empty_string() {
7          assertFalse(PalindromeCheck.isPalindrome(str: ""));
8      }
9      @Test
10     public void test_one_Character() {
11         assertTrue(PalindromeCheck.isPalindrome(str: "s"));
12     }
13     @Test
14     public void test_2_diff_Characters() {
15         assertTrue(PalindromeCheck.isPalindrome(str: "ty"));
16     }
17     @Test
18     public void test2_same_Characters() {
19         assertTrue(PalindromeCheck.isPalindrome(str: "zz"));
20     }
21     @Test
22     public void test_a_word_that_is_Palindrome() {
23         assertTrue(PalindromeCheck.isPalindrome(str: "mam"));
24     }
25     @Test
26     public void test_a_word_that_is_not_a_Palindrome() {
27         assertFalse(PalindromeCheck.isPalindrome(str: "arfah"));
28     }
29     @Test
30     public void test_special_characters() {
31         assertTrue(PalindromeCheck.isPalindrome(str: "!!"));
32     }
33
34
35 }
36
```

## Tested Results:

The screenshot displays the Eclipse IDE interface. The top-left pane shows the project structure with 'Testing' as the active project. The top-right pane shows the source code of 'PalindromeCheckTest.java', which includes imports for 'org.junit.framework.TestCase' and 'org.junit.Test', and two test methods: 'test\_empty\_string()' and 'test\_one\_character()'. The bottom-left pane shows the 'Run' configuration for 'PalindromeCheckTest'. The bottom-right pane displays the test results, indicating that 2 tests failed and 5 passed out of 7 total tests. The failed tests are 'test\_empty\_string' and 'test\_2\_diff\_Characters', both failing with 'AssertionFailedError'. The console output shows the command used to run the tests and the resulting error messages for the failed tests.

```
package org.example;
import junit.framework.TestCase;
import org.junit.Test;

public class PalindromeCheckTest extends TestCase {

    @Test
    public void test_empty_string() {
        assertEquals(PalindromeCheck.isPalindrome(""));
    }

    @Test
    public void test_one_character() {

```

Run: PalindromeCheckTest

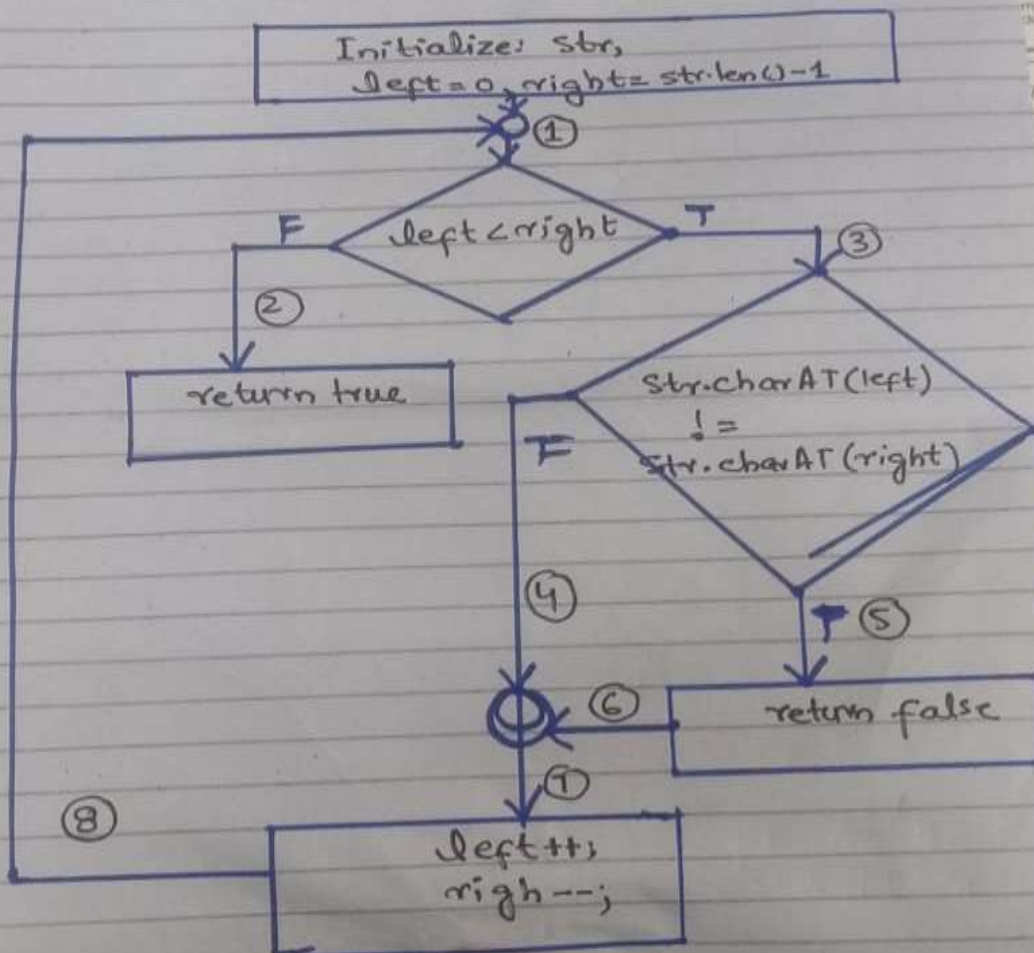
Tests failed: 2, passed: 5 of 7 tests - 18ms

- test\_2\_same\_Characters: 7ms
- test\_a\_word\_that\_is\_not: 1ms
- test\_one\_Character: 1ms
- test\_special\_characters: 1ms
- test\_empty\_string: 8ms
- test\_2\_diff\_Characters: 2ms
- test\_a\_word\_that\_is\_Pals: 1ms

Process finished with exit code -1

# CFG:

Name: Arfah Ali  
Reg:No: FA21-BSE-080  
Algorithm Name: check palindrom  
String (2)



Paths:

- ① 1 → 2(F)
- ② 1 → 3(T) → 5(T) → 6 → 7 → 8
- ③ 1 → 3(T) → 4(F) → 7 → 8

**Paths:**

1. **Path 1:** 1 → 2 (F)
2. **Path 2:** 1 → 3 (T) → 5 (T) → 6 → 7 → 8
3. **Path 3:** 1 → 3 (T) → 4 (F) → 7 → 8

**Test Cases:**

I gave some wrong inputs so that program passes through the False paths as well. One with one string, One with multiple but wrong so that it passes the condition inside the loop, and one with more than one correct order of characters. I have checked special characters order and an empty string as well. Check the implementation in the provided code.

Test Case ID	Description	Input Data	Expected outcome	Actual outcome	Status/Verdict
TC1	Keep it an empty string, which is a palindrome.	""	It is a palindrome	It is not a palindrome	Fail
TC2	Enter one character which is also a palindrome.	"s"	It is a palindrome	It is a palindrome	Pass
TC3	Enter 2 different characters which do not make palindrome.	"ty"	It is not a palindrome	It is a palindrome	Fail
TC4	Enter 2 same characters which make a palindrome string.	"zz"	It is a palindrome	It is a palindrome	Pass
TC5	Enter a word that is a palindrome.	"mam"	It is a palindrome	It is a palindrome	Pass
TC6	Enter a complete word that is not a palindrome.	"arfah"	It is not a palindrome	It is not a palindrome	pass
TC7	Enter special characters	"!!"	It is a palindrome	It is a palindrome	pass

**THE END**