Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Кубанский государственный технологический университет»

(ФГБОУ ВО «КубГТУ»)

Факультет информационных технологий и кибербезопасности

Кафедра информационных систем и программирования

**ОТЧЕТ**

по лабораторной работе №4.

Тема работы: «Тестирование методом белого ящика».

Наименование дисциплины: «Тестирование и отладка программного обеспечения».

Подготовил:

Студент группы 23-КБ-ПР1

Юданов М. Д.

Проверил:

Волик А. Г.

Краснодар

**2025**

**Цель работы**: изучить подход к тестированию методом белого ящика.

**Ход работы**:

13. Создать класс, реализующий метод для замены подстроки в строке на другую подстроку.

**Код программы:**

// Вариант 13

class StringReplacer {

def replace(originalString: String, searchString: String, replacementString: String): String = {

// 1: Проверяем, что строка поиска не пуста

if (searchString.isEmpty) return originalString // 2

val result = new StringBuilder // 3

var currentIndex = 0 // 4

// 5: Пока не дошли до конца строки

while (currentIndex < originalString.length) {

val remainingLength = originalString.length - currentIndex // 9

var matchFound = false // 6

if (remainingLength >= searchString.length) { // 10

// 12: Посимвольное сравнение без break

var tempSearchIndex = 0

var allMatch = true

while (tempSearchIndex < searchString.length && allMatch) {

if (originalString.charAt(currentIndex + tempSearchIndex) != searchString.charAt(tempSearchIndex)) {

allMatch = false

}

tempSearchIndex += 1

}

matchFound = allMatch

}

if (matchFound) { // 18

result.append(replacementString) // 19

currentIndex += searchString.length // 20

} else {

result.append(originalString.charAt(currentIndex)) // 21

currentIndex += 1 // 22

}

}

result.toString // 23

}

}

2. Строю управляющий граф программы, показывающий последовательность выполнения строк кода.

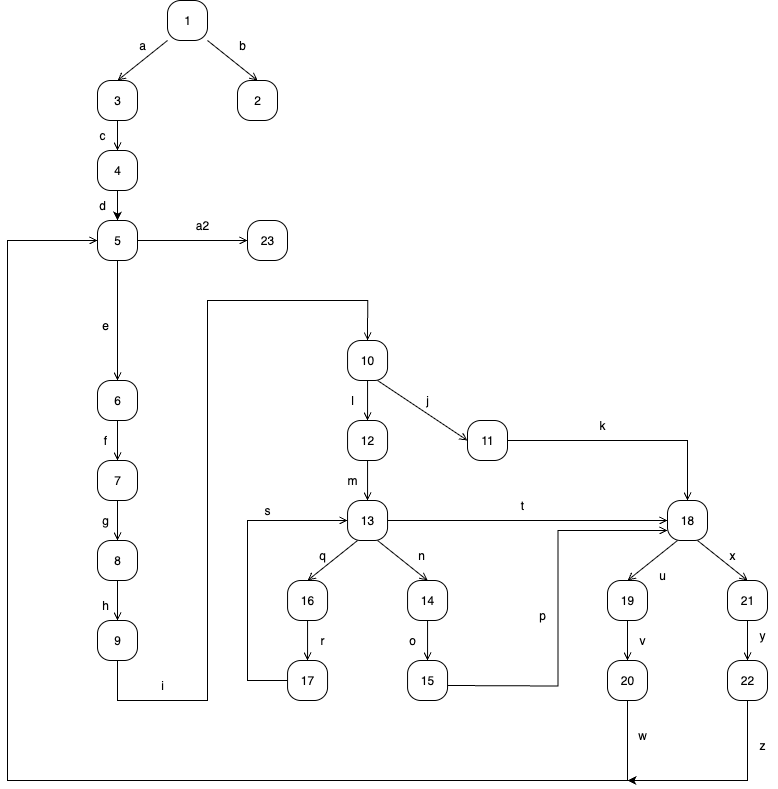


Рисунок 1 – Управляющий граф класса

3. Пишу набор тестирований для класса с разными входными значениями.

**Код тестирований:**

import org.scalatest.funsuite.AnyFunSuite  
  
class StringReplacerTests extends AnyFunSuite {  
  
 var *replacer*: StringReplacer = \_  
  
  
 def beforeEach(): Unit = {  
 *replacer* = new StringReplacer()  
 }  
  
 def afterEach(): Unit = {  
 *replacer* = null  
 }  
  
 // MARK: - Граничные случаи и обработка ошибок  
  
 test("replace — with empty search string") {  
 beforeEach()  
 val original = "Hello"  
 val search = ""  
 val replacement = "World"  
 val result = *replacer*.replace(original, search, replacement)  
 assert(result == original, "Должно возвращать оригинальную строку, если строка поиска пуста")  
 afterEach()  
 }  
  
 test("replace — empty original string") {  
 beforeEach()  
 val original = ""  
 val search = "abc"  
 val replacement = "xyz"  
 val result = *replacer*.replace(original, search, replacement)  
 assert(result == "", "Должно возвращать пустую строку, если исходная строка пуста")  
 afterEach()  
 }  
  
 // MARK: - Основные пути выполнения (покрытие веток)  
  
 test("replace — no match") {  
 beforeEach()  
 val original = "abcdef"  
 val search = "xyz"  
 val replacement = "123"  
 val result = *replacer*.replace(original, search, replacement)  
 assert(result == original, "Должно возвращать оригинальную строку, если совпадений нет")  
 afterEach()  
 }  
  
 test("replace — single match at beginning") {  
 beforeEach()  
 val original = "abcde"  
 val search = "ab"  
 val replacement = "XX"  
 val result = *replacer*.replace(original, search, replacement)  
 assert(result == "XXcde", "Должно заменить подстроку в начале")  
 afterEach()  
 }  
  
 test("replace — single match in middle") {  
 beforeEach()  
 val original = "axbyc"  
 val search = "xb"  
 val replacement = "YY"  
 val result = *replacer*.replace(original, search, replacement)  
 assert(result == "aYYyc", "Должно заменить подстроку в середине")  
 afterEach()  
 }  
  
 test("replace — single match at end") {  
 beforeEach()  
 val original = "abcd"  
 val search = "cd"  
 val replacement = "XY"  
 val result = *replacer*.replace(original, search, replacement)  
 assert(result == "abXY", "Должно заменить подстроку в конце")  
 afterEach()  
 }  
  
 test("replace — multiple non-overlapping matches") {  
 beforeEach()  
 val original = "banana"  
 val search = "an"  
 val replacement = "!!"  
 val result = *replacer*.replace(original, search, replacement)  
 assert(result == "b!!!!a", "Должно заменить все непересекающиеся вхождения")  
 afterEach()  
 }  
  
 test("replace — with empty replacement string (deletion)") {  
 beforeEach()  
 val original = "applepineapple"  
 val search = "apple"  
 val replacement = ""  
 val result = *replacer*.replace(original, search, replacement)  
 assert(result == "pine", "Должно удалять подстроку")  
 afterEach()  
 }  
}

4. При обычных входных значениях (когда есть не пустая строка и есть разделители) алгоритм выполняется, проходя через все пути. В случае, когда на вход дается пустой массив разделителей, программа не выполнит 8, 9 и 10 строки. Соответственно строки с 15 до 22 включительно так же будут пропущены.

При вводе строки «Кот. Кит.» и массива разделителей, состоящего только из единственной точки, будет пройден полный путь:

abcd ezj ezj ezj efghij ezj ezj ezj ezj efghij klmno qr stu stu stu stu vwxy qr stu stu stu stu vwxy p

При вводе строки «Тут нет текста.» и пустого массива разделителей, будет пройден неполный путь:

abcd ezj ezj ezj ezj ezj ezj ezj ezj ezj ezj ezj ezj ezj ezj ezj klmno p

5. В случае, когда программа проходит первый путь, происходит полное покрытие кода. Это включает в себя покрытие всех операторов, условий и путей. Метод возвращает массив строк, состоящий из «Кот.» и «Кит.»

В случае, когда программа проходит второй путь, происходит неполное покрытие кода. Покрываются не все операторы и не все условия и пути. Метод возвращает пустой массив строк.

Двух случаев хватает, чтоб утверждать, что все решения покрыты.

6. С точки зрения покрытия кода тестами, можно считать код полностью покрытым, так как использованы были все методы покрытия.

7. Запускаю тесты и фиксирую результаты.

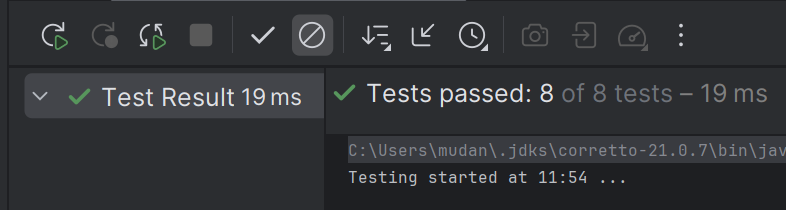


Рисунок 2 – Результаты тестирований.

8. Делаю вывод касательно проделанной работы.

**Вывод:** в ходе выполнения лабораторной работы я изучил подход к тестированию методом белого ящика. Также научился тестировать методы классов на основе тестового набора с использованием программных отладочных средств.