Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

ОТЧЕТ
по лабораторной работе №6
по дисциплине СПП

Выполнил:
студ. гр.ПО-5
Харкевич Д.А.

Проверил:
Крощенко А.А.
Ст.преп. кафедры ИИТ

Брест 2021

Цель работы: введение в тестирование и отладку кода.

Вариант 13

Задание 1.

Часть 1. Написать тесты к методу, а затем реализовать сам метод по заданной спецификации.

Реализуйте и протестируйте метод `int indexOfDifference(String str1, String str2)` который сравнивает две строки и возвращает индекс той позиции, в которой они различаются. Например, `indexOfDifference("i am a machine", "i am a robot")` должно вернуть 7.

```
StringUtils.indexOfDifference(null, null) = NullPointerException
StringUtils.indexOfDifference("", "") = -1
StringUtils.indexOfDifference("", "abc") = 0
StringUtils.indexOfDifference("abc", "") = 0
StringUtils.indexOfDifference("abc", "abc") = -1
StringUtils.indexOfDifference("ab", "abxyz") = 2
StringUtils.indexOfDifference("abcde", "abxyz") = 2
StringUtils.indexOfDifference("abcde", "xyz") = 0
```

Часть 2. Самостоятельно разработайте спецификацию метода, реализуйте её в тестах, а затем реализуйте сам метод.

Напишите метод `String difference(String str1, String str2)` сравнивающий две строки и возвращающий разницу между ними. Другими словами, метод должен остаток от строки, где они различаются.

Реализация:

StringUtils.java

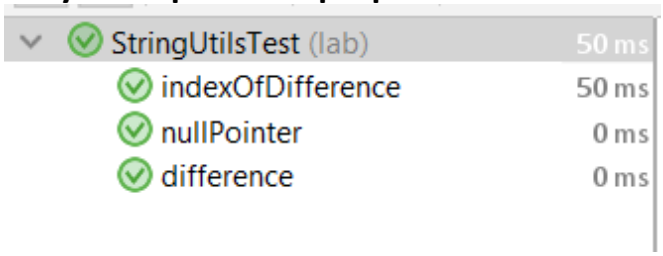
```
package lab;

public class StringUtils {

    public static int indexOfDifference(String str1, String str2) {
        if (str1 == null && str2 == null) {
            throw new NullPointerException();
        }
        if (str1.equals(str2)) {
            return -1;
        }
        if (str1 == "" || str2 == "") {
            return 0;
        }
        if (str1.length() > str2.length() &&
str1.substring(0,str2.length()).equals(str2)) {
            return str2.length();
        }
        if (str2.length() > str1.length() &&
str2.substring(0,str1.length()).equals(str1)) {
            return str1.length();
        }
        for (int i = 0; i < Math.min(str1.length(), str2.length()); i++) {
            if (str1.charAt(i) != str2.charAt(i))
                return i;
        }
```

```
        return 0;
    }

    public static String difference(String str1, String str2) {
        if (str1 == null && str2 == null) {
            throw new NullPointerException();
        }
        if (StringUtils.indexOfDifference(str1, str2) == -1)
            return "";
        if (StringUtils.indexOfDifference(str1, str2) == 0)
            return str1;
        return str1.substring((StringUtils.indexOfDifference(str1, str2)),
str1.length());
    }

}
```

StringUtilsTest.java

```
package lab;
import org.junit.Test;
import static org.junit.Assert.*;

public class StringUtilsTest {

    @Test(expected = NullPointerException.class)
    public void nullPointer() {
        StringUtils.indexOfDifference(null, null);
        StringUtils.difference(null, null);
    }

    @Test
    public void indexOfDifference() {
        assertEquals(StringUtils.indexOfDifference("", ""), -1);
        assertEquals(StringUtils.indexOfDifference("", "abc"), 0);
        assertEquals(StringUtils.indexOfDifference("abc", ""),0);
        assertEquals(StringUtils.indexOfDifference("abc", "abc"), -1);
        assertEquals(StringUtils.indexOfDifference("abxyz", "ab"), 2);
        assertEquals(StringUtils.indexOfDifference("ab", "abxyz"), 2);
        assertEquals(StringUtils.indexOfDifference("abcde", "abxyz"), 2);
        assertEquals(StringUtils.indexOfDifference("abcde", "xyz"), 0);
    }

    @Test
    public void difference() {
        assertEquals(StringUtils.difference("", ""), "");
        assertEquals(StringUtils.difference("abc", "abc"), "");
        assertEquals(StringUtils.difference("abc", ""), "abc");
        assertEquals(StringUtils.difference("", "vds"), "");
        assertEquals(StringUtils.difference("abcdef", "abc"), "def");
        assertEquals(StringUtils.difference("abcdef", "abcxys"), "def");
    }
}
```

Результат работы программы



Задание 2.

Импортируйте один из проектов по варианту:

- **Stack** — проект содержит реализацию стека на основе связного списка: `Stack.java`.

Реализация:

StackClient.java

```
package stack;

import java.util.Scanner;


/**
 * A test client.
 */
public class StackClient {
    public static void main(String[] args) {
        Stack<String> s = new Stack<String>();

        Scanner scanner = new Scanner(System.in);

        while (scanner.hasNext()) {
            String item = scanner.next();
            if (!item.equals("-")) {
                s.push(item);
            } else if (!s.isEmpty()) {
                System.out.println(s.pop() + " ");
            }
        }

        System.out.println(s.size());
    }
}
```

Stack.java

```
package stack;

import java.util.NoSuchElementException;
/**
 * The <tt>Stack</tt> class represents a last-in-first-out (LIFO) stack of
 * generic items. It supports the usual <em>push</em> and <em>pop</em>
 * operations, along with methods for peeking at the top item, testing if the
 * stack is empty, and iterating through the items in LIFO order.
 * <p>
 */
//TODO FIXME Find Bugs & Write Tests
public class Stack<Item> {
    private int N; // size of the stack
    private Node first; // top of stack

    // helper linked list class
    private class Node {
        private Item item;
        private Node next;
    }

    /**
     * Create an empty stack.
     */
    public Stack() {
        first = null;
        N = 0;
```

```
        assert check();
    }

    /**
     * Is the stack empty?
     */
    public boolean isEmpty() {
        return N == 0;
    }

    /**
     * Return the number of items in the stack.
     */
    public int size() {
        return N;
    }

    /**
     * Add the item to the stack.
     */
    public void push(Item item) {
        Node oldfirst = first;
        first = new Node();
        first.item = item;
        first.next = oldfirst;
        N++;
        assert check();
    }

    /**
     * Delete and return the item most recently added to the stack.
     *
     * @throws java.util.NoSuchElementException if stack is empty.
     */
    public Item pop() {
        if(isEmpty()){
            throw new NoSuchElementException();
        }
        Item item = first.item; // save item to return
        first = first.next; // delete first node
        N--;
        assert check();
        return item; // return the saved item
    }

    /**
     * Return the item most recently added to the stack without deletion.
     *
     * @throws java.util.NoSuchElementException if stack is empty.
     */
    public Item peek() {
        if(isEmpty()){
            throw new NoSuchElementException();
        }
        return first.item;
    }

    /**
     * Return string representation.
     */
    public String toString() {
        StringBuilder s = new StringBuilder();
        for (Node current = first; current != null; current = current.next) {
            Item item = current.item;
```

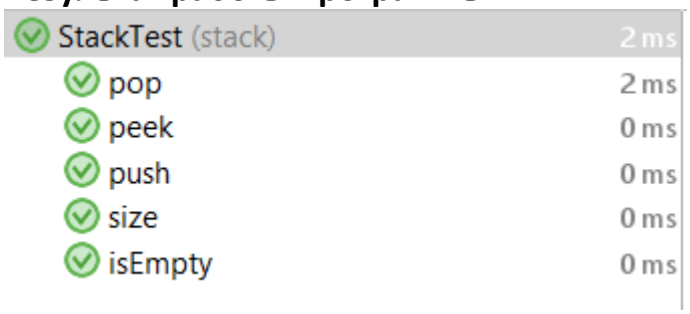
```
            s.append(item).append(" ");
        }
        return s.toString();
    }

    // check internal invariants
    private boolean check() {
        if (N == 0) {
            if (first != null) {
                return false;
            }
        } else if (N == 1) {
            if (first == null) {
                return false;
            }
            if (first.next != null) {
                return false;
            }
        } else {
            if (first.next == null) {
                return false;
            }
        }

        // check internal consistency of instance variable N
        int numberOfNodes = 0;
        for (Node x = first; x != null; x = x.next) {
            numberOfNodes++;
        }
        if (numberOfNodes != N) {
            return false;
        }

        return true;
    }
}
```

StackTest.java

```
package stack;

import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.*;

public class StackTest {
    Stack<String> stack;

    @Before
    public void init() {
        stack = new Stack<>();
    }
    @Test
    public void isEmpty() {
        assertTrue(stack.isEmpty());
        stack.push("TestItem");
        assertFalse(stack.isEmpty());
    }

    @Test
    public void size() {
        assertEquals(0,stack.size());
        stack.push("1");
        stack.push("2");
```

```
        stack.push("3");
        assertEquals(3, stack.size());
    }

    @Test
    public void push() {
        assertEquals(0,stack.size());
        stack.push("1");
        assertEquals(1, stack.size());
        stack.push("2");
        assertEquals(2, stack.size());
        assertEquals("2", stack.peek());
    }

    @Test
    public void pop() {
        assertEquals(0,stack.size());
        stack.push("1");
        stack.push("2");
        stack.pop();
        assertEquals(1, stack.size());
        stack.pop();
        assertEquals(0,stack.size());
    }

    @Test
    public void peek() {
        assertEquals(0,stack.size());
        stack.push("1");
        assertEquals("1", stack.peek());
        stack.push("2");
        assertEquals("2", stack.peek());
        stack.pop();
        assertEquals("1", stack.peek());
    }
}
```

Результат работы программы:



Вывод: ознакомился с тестированием и отладкой кода.