Министерство образования Республики Беларусь
Учреждение образования
“Брестский государственный технический университет”
Кафедра ИИТ

Отчёт
По лабораторной работе №11
По дисциплине СПП

Выполнил
Студент группы ПО-3
3-го курса
Кулинкович И. Т.

Проверил
Крощенко А. А.

Брест 2021

Лабораторная работа №11

ВАРИАНТ 13

Задание 1 – Введение в JUnit

• Создаете новый класс и скопируйте код класса Sum;
• Создаете тестовый класс SumTest;
• Напишите тест к методу Sum.accum и проверьте его исполнение. Тест должен проверять работоспособность функции accum.
• Очевидно, что если передать слишком большие значения в Sum.accum, то случится переполнение. Модифицируйте функцию Sum.accum, чтобы она возвращала значение типа long и напишите новый тест, проверяющий корректность работы функции с переполнением. Первый тест должен работать корректно.

```
public class Sum {
      public static int accum ( int ... values ) {
            int result = 0;
            for ( int i = 0; i < values . length ; i ++) {
                  result += values [ i ];
            }
            return result ;
      }
}
```

Задание 2 – Тестирование функций

Подготовка к выполнению:
• Создайте новый проект в рабочей IDE;
• Создайте класс StringUtils, в котором будут находится реализуемые функции;
• Напишите тесты для реализуемых функций.

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

Реализуйте функцию String loose(String str, String remove), удаляющую из первой строки все символы, которые есть так же во второй.
Спецификация метода:

```
loose (null , null ) = NullPointerException
loose (null , *) = null
loose ("", *) = ""
loose (* , null ) = *
loose (* , "") = *
loose (" hello ", "hl") = "eo"
loose (" hello ", "le") = “ho"
```

Задание 3 – Поиск ошибок, отладка и тестирование классов

Импортируйте один из проектов по варианту:
Stack – проект содержит реализацию стека на основе связного списка: Stack.java.

Провести поиск ошибок, проверить внутреннюю корректность программы, реализовать пропущенные части программы, написать тесты.

Код программы

src: live.ilyusha.Sum

```
package live.ilyusha;
```

```
public class Sum {
    public static long accum(int... values) {
        long result = 0;
        for (int i = 0; i < values.length; i++) {
            result += values[i];
        }
        return result;
    }
}
```

src_test: live.ilyusha.SumTest

```
import live.ilyusha.Sum;
import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class SumTest {
    @Test
    public void accum() {
        assertEquals(7, Sum.accum(1, 4, 2));
    }

    @Test
    public void accumLong() {
        assertEquals(Integer.MAX_VALUE + 1L, Sum.accum(Integer.MAX_VALUE,
1));
    }
}
```

src: live.ilyusha.StringUtils

```
package live.ilyusha;

public class StringUtils {
    public static String loose(String str, String remove) {
        if (remove == null && str == null)
            throw new NullPointerException();
        else if (remove == null)
            return str;
        if (str == null)
            return null;
        String result = "";
        for (Character c : str.toCharArray()) {
            if (!remove.contains(c.toString()))
                result = result.concat(c.toString());
        }
        return result;
    }
}
```

src_test: live.ilyusha.StringUtilsTest

```
import live.ilyusha.StringUtils;
import org.junit.Test;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertNull;

public class StringUtilsTest {
```

```
    @Test(expected = NullPointerException.class)
    public void looseByNullRemove() {
        assertEquals("", StringUtils.loose(null, null));
    }

    @Test
    public void looseSuccess() {
        assertNull(StringUtils.loose(null, "any"));
        assertEquals("", StringUtils.loose("", "anyystr"));
        assertEquals("anyyyy", StringUtils.loose("anyyyy", null));
        assertEquals("anyyyy", StringUtils.loose("anyyyy", ""));
        assertEquals("eo", StringUtils.loose("hello", "hl"));
        assertEquals("ho", StringUtils.loose("hello", "le"));
    }
}
```

src: live.ilyusha.Stack

```
package live.ilyusha;

public class Stack<Item> {
    private int N;
    private Node first;

    private class Node {
        private Item item;
        private Node next;
    }

    public Stack() {
        first = null;
        N = 0;
        assert check();
    }

    public boolean isEmpty() {
        return size() == 0;
    }

    public int size() {
        return N;
    }

    public void push(Item item) {
        Node oldFirst = first;
        first = new Node();
        first.item = item;
        first.next = oldFirst;
        N++;
        assert check();
    }

    public Item pop() {
        if (isEmpty()) {
            throw new IllegalStateException("The stack is empty.");
        }
        Item item = first.item;
        first = first.next;
        N--;
        assert check();
```

```
        return item;
    }

    public Item peek() {
        if (isEmpty()) {
            throw new IllegalStateException("The stack is empty.");
        }
        return first.item;
    }

    public String toString() {
        StringBuilder s = new StringBuilder();
        for (Node current = first; current != null; current = current.next) {
            Item item = current.item;
            s.append(item + " ");
        }
        return s.toString().replaceFirst("\\s$","");
    }

    private boolean check() {
        int numberOfNodes = 0;
        for (Node x = first; x != null; x = x.next) {
            numberOfNodes++;
        }
        if (numberOfNodes != N) {
            return false;
        }

        if (N == 0) {
            return first == null;
        } else if (N == 1) {
            return first != null && first.next == null;
        } else {
            return first.next != null;
        }
    }
}
```

src_test: live.ilyusha.StackTest

```
import live.ilyusha.Stack;
import org.junit.Before;
import org.junit.Test;

import java.util.Optional;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertFalse;

public class StackTest {
    private Stack<Integer> stack;

    @Before
    public void prepareStack() {
        stack = new Stack<>();
        stack.push(10);
        stack.push(20);
        stack.push(30);
    }
```

```
    @Test
    public void isEmptyCorrect() {
        assertFalse(stack.isEmpty());
    }

    @Test
    public void sizeCorrectValue() {
        assertEquals(3, stack.size());
    }

    @Test
    public void pushIncreasesSize() {
        stack.push(40);
        assertEquals(4, stack.size());
    }

    @Test
    public void popDecreasesSize() {
        stack.pop();
        assertEquals(2, stack.size());
    }

    @Test
    public void peekMaintainsSize() {
        stack.peek();
        assertEquals(3, stack.size());
    }

    @Test
    public void peekCorrectValue() {
        assertEquals(new Integer(30), stack.peek());
    }

    @Test
    public void toStringCorrectValue() {
        assertEquals("30 20 10", stack.toString());
    }
}
```

src_test: live.ilyusha.StackErrorsTest

```
import live.ilyusha.Stack;
import org.junit.Before;
import org.junit.Test;

public class StackErrorsTest {
    private Stack<Integer> stack;

    @Before
    public void prepareStack() {
        stack = new Stack<>();
        stack.push(10);
        stack.push(20);
        stack.push(30);
    }

    @Test(expected = IllegalStateException.class)
    public void popEmptyFailure() {
        while (true) {
            stack.pop();
```

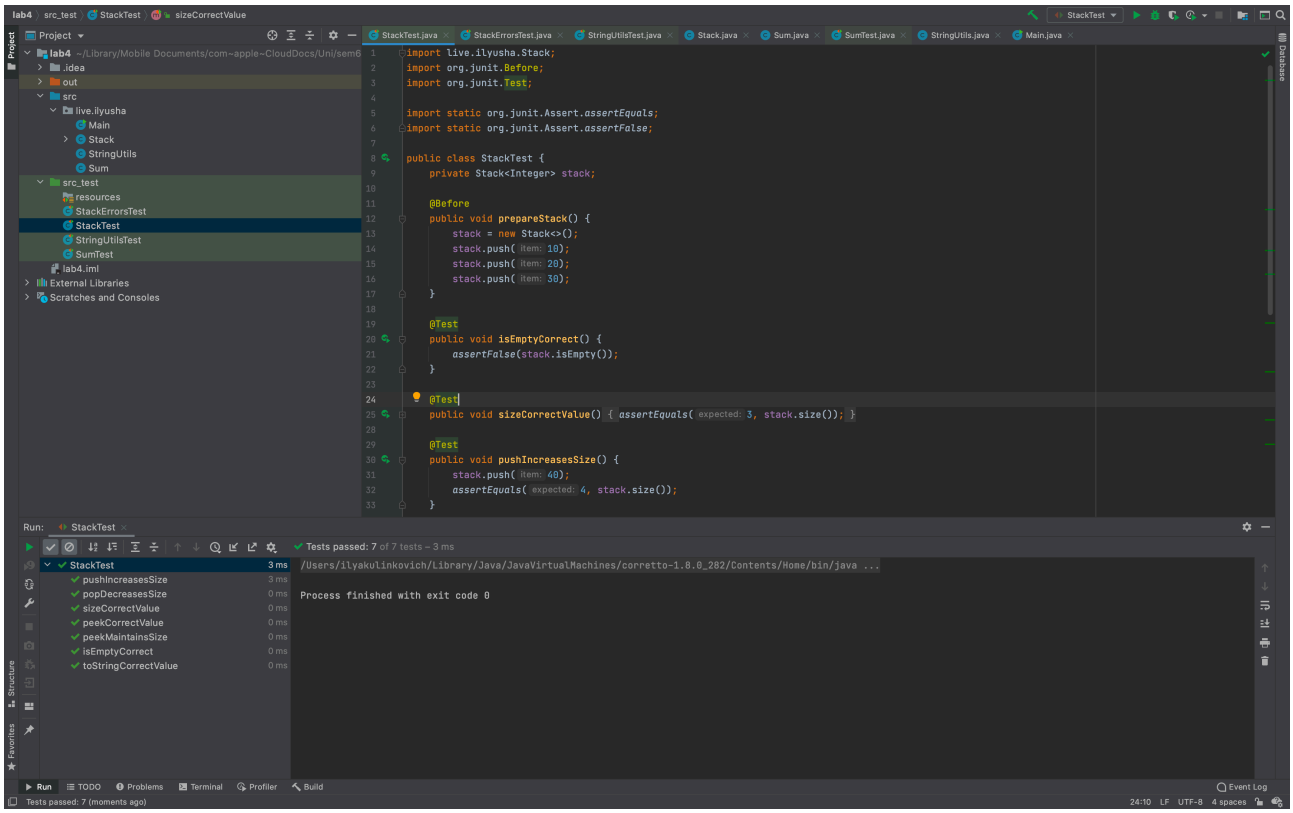
```
        }
    }

    @Test(expected = IllegalStateException.class)
    public void peekEmptyFailure() {
        for (int i = 0; i < 3; i++) {
            stack.pop();
        }
        stack.peek();
    }
}
```

Спецификация ввода
Данные не вводятся.

Спецификация вывода
В результате выполнения теста среда разработки сообщает об статусе выполнения тестов, статус успешный.

Рисунки с результатами работы программы



Вывод
В данной лабораторной работе я освоил приемы тестирования кода на примере использования библиотеки JUnit.