Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №5

за 6 семестр

По дисциплине: «СПП»

Выполнил:

Студент 3 курса

Группы ПО-6(1)

Мартынович Д. М.

Проверил:

Монтик Н.С.

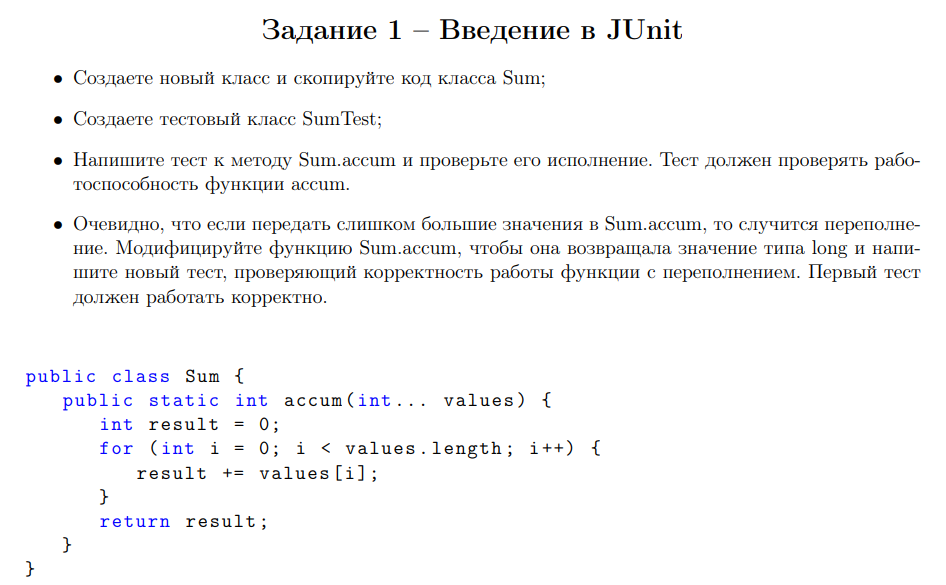
2023

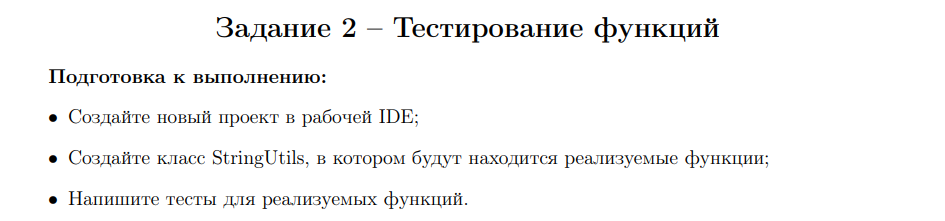
Лабораторная работа №5

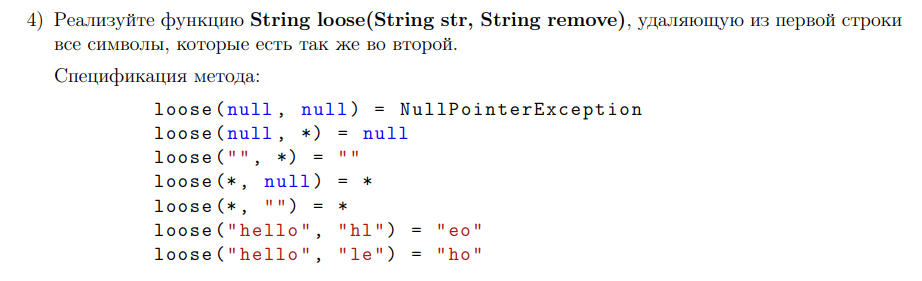
**Цель работы:** освоить приемы тестирования кода на примере использования библиотеки JUnit.

Вариант 11

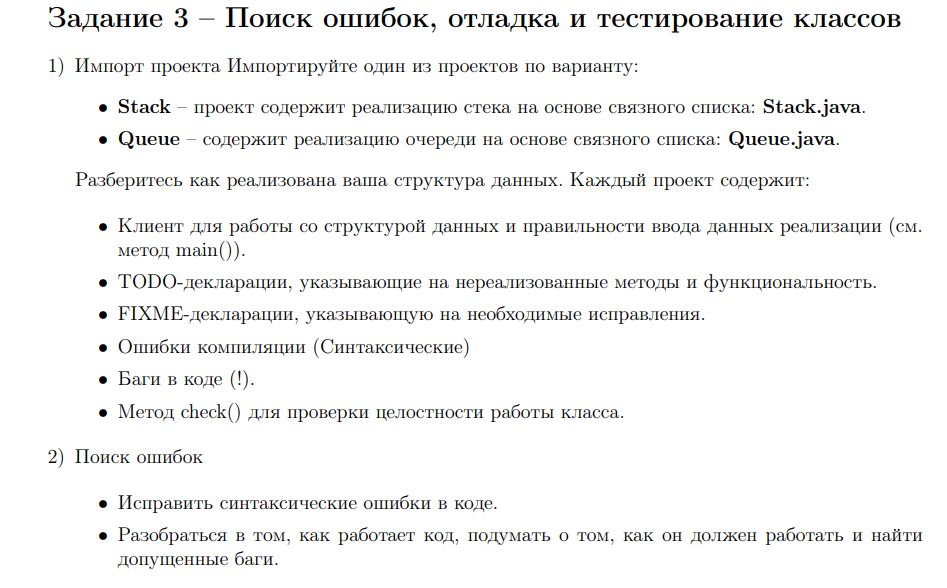
**Задание1:**

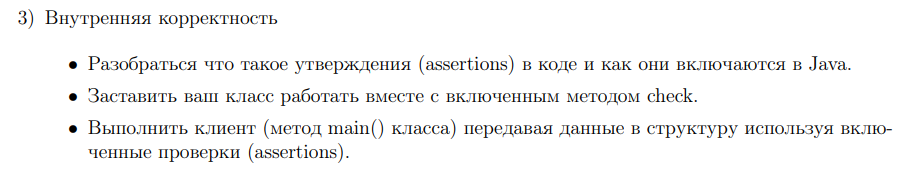
**Задание2:**

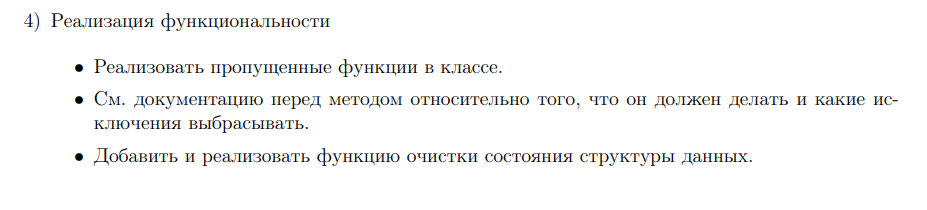


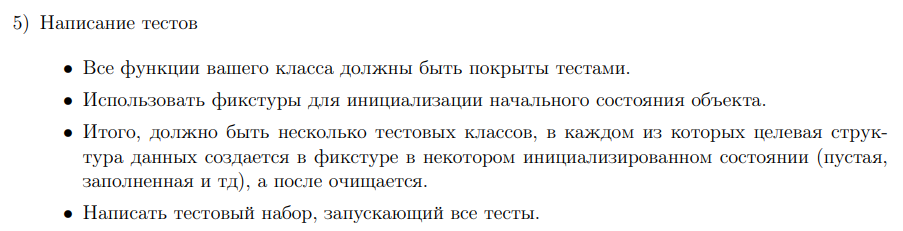


**Задание3:**









**Код программы:**

**IntSum.java**

package task1;

public class IntSum {

public int accum ( int[] values ) {

int result = 0;

for ( int i = 0; i < values.length; i++) {

result += values[i];

}

return result;

}

}

**IntSumTest.java**

package task1;

import org.junit.Test;

import static org.junit.Assert.\*;

public class IntSumTest {

@Test

public void accum() {

int[] val = new int[]{1, 2, 3, 4, 5};

IntSum is = new IntSum();

int actual = is.accum(val); // реальность

int expected = 15; // ожидание

assertEquals(expected, actual); // проверка на эквивалентность

}

}

**LongSum.java**

package task2;

public class LongSum {

public static long accum ( long[] values ) {

long result = 0;

for ( int i = 0; i < values.length; i++) {

result += values[i];

}

return result;

}

}

**LingSumTest.java**

package task2;

import org.junit.Test;

import static org.junit.Assert.\*;

public class LongSumTest {

@Test

public void accum() {

long[] val = new long[]{Integer.MAX\_VALUE, 2};

LongSum ls = new LongSum();

long actual = ls.accum(val); // реальность

System.out.println(actual);

long expected = Integer.MAX\_VALUE + 2L; // ожидание

System.out.println(expected);

assertEquals(expected, actual); // проверка на эквивалентность

}

}

**StringUtils.java**

package task3;

public class StringUtils {

public static String loose(String str, String remove){

if(str == null && remove != null){

return null;

}

else if(str == null && remove == null){

throw new NullPointerException();

}

else if(str.equals(" ")){

return str;

}

else{

String out;

String st = str.toLowerCase();

String rem = remove.toLowerCase();

char[] t = new char[st.length()];

for(int k = 0; k < st.length(); k++){

for(int i = 0; i < rem.length(); i++){

if(st.charAt(k) == rem.charAt(i)){

st = st.replace(st.charAt(k), ' ');

}

}

}

out = st.replaceAll(" ", "");

return out;

}

}

}

**StringUtilsTest.java**

package task3;

import org.junit.Test;

import static org.junit.Assert.\*;

public class StringUtilsTest {

@Test(expected = NullPointerException.class)

public void looseByNullRemove() {

StringUtils su = new StringUtils();

String actual1 = su.loose(null, null);

String expected1 = "";

assertEquals(expected1, actual1);

}

@Test

public void loose() {

StringUtils su = new StringUtils();

String actual2 = su.loose(null, "help");

String expected2 = null;

assertEquals(expected2, actual2);

assertNull(actual2);

String actual3 = su.loose("help", "");

String expected3 = "help";

assertEquals(expected3, actual3);

String actual4 = su.loose(" ", "help");

String expected4 = " ";

assertEquals(expected4, actual4);

String actual5 = su.loose("help", " ");

String expected5 = "help";

assertEquals(expected5, actual5);

String actual6 = su.loose("Hello My World", "hmwl");

String expected6 = "eoyord";

assertEquals(expected6, actual6);

}

}

**Stack.java**

package task4;

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;

}

}

}

**StackTest.java**

package task4;

import org.junit.Before;

import org.junit.Test;

import java.util.Optional;

import static org.junit.Assert.\*;

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());

}

}

**StackErrorTest.java**

package task4;

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();

}

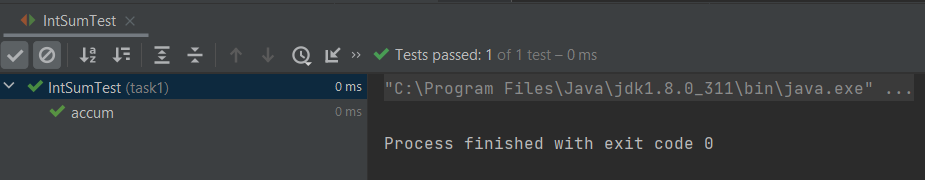
stack.peek();

}

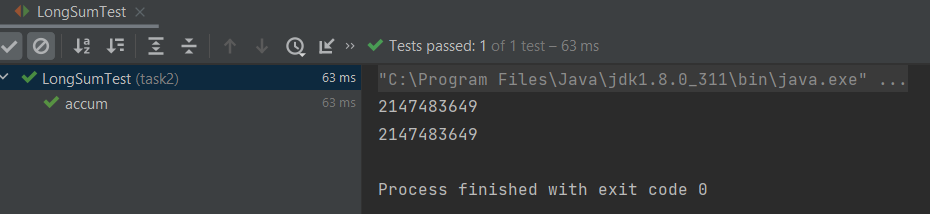
}

**Результат программы:**

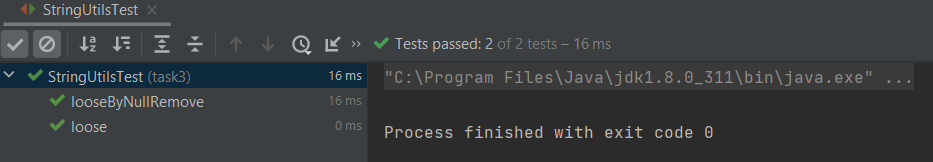
IntSum



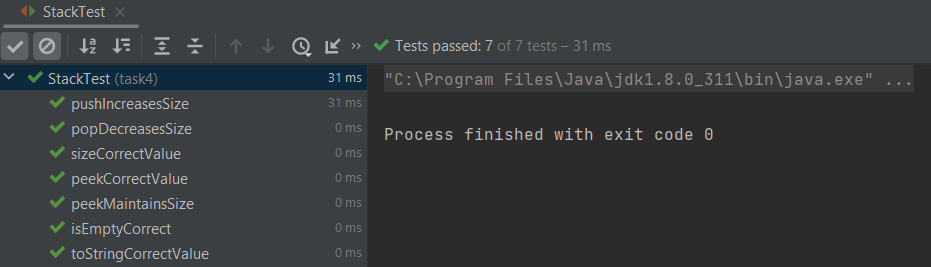
LongSum

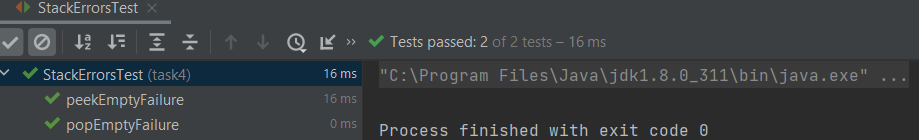


StringUtils



Stack





**Вывод:** освоил приемы тестирования кода на примере использования библиотеки JUnit.