Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет» Кафедра ИИТ

Лабораторная работа №11

Выполнил:

Студент группы ПО-5

Пищик А. В.

Проверил:

Крощенко А. А.

Брест 2022

**Вариант 11**

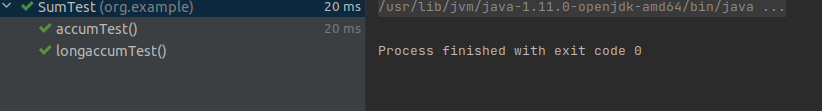
**Цель**: освоить приемы тестирования кода на примере использования библиотеки JUnit.

**Задание 1**

* 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 рабо- тоспособность функции accum.

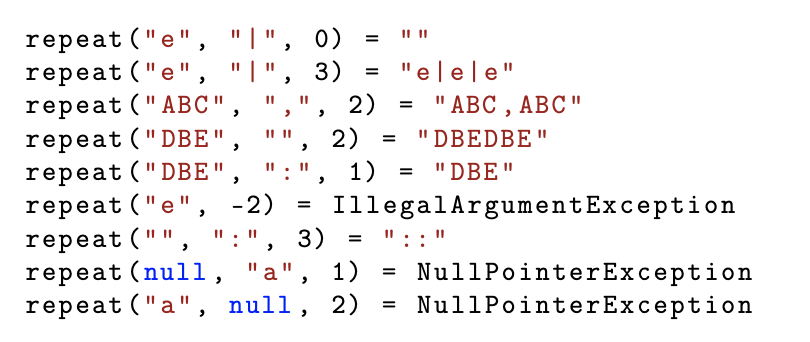
**SumTest**

package org.example.Test;  
import org.example.Sum;  
import org.junit.jupiter.api.Test;  
import static org.junit.jupiter.api.Assertions.*assertEquals*;  
  
public class SumTest {  
 @Test  
 public void accumTest() {  
 int TestaccumInt = Sum.*accum*(23, 55, 64, 1, -24);  
 *assertEquals*(119, TestaccumInt);  
 }  
  
 @Test  
 public void longaccumTest(){  
 long TestaccumLong = Sum.*longaccum*(312387847,682874745 );  
 *assertEquals*(995262592, TestaccumLong);  
 }  
}

**Задание 2**

Разработайте метод String repeat(String str, String separator, int repeat), который строит строку из указанного паттерна, повторённого заданное количество раз, вставляя строку-разделитель при каждом повторении.

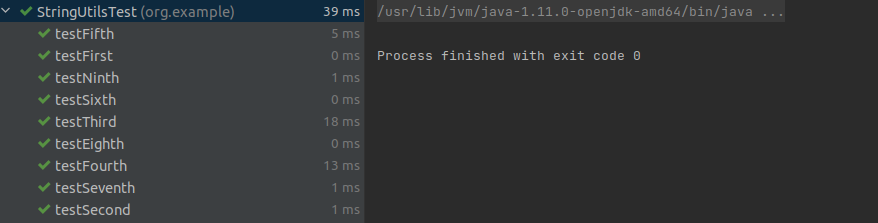
Спецификация метода:



**StringUtilsTest.java**

package org.example.Test;import org.example.StringUtils;import org.junit.Test;import org.junit.jupiter.api.Assertions;import static org.junit.jupiter.api.Assertions.*assertEquals*;public class StringUtilsTest { @Test public void testFirst() { Assertions.*assertEquals*("", StringUtils.*repeat*("e","|",0)); } @Test public void testSecond() { *assertEquals*("e|e|e", StringUtils.*repeat*("e", "|", 3)); } @Test public void testThird() { *assertEquals*("ABC , ABC ", StringUtils.*repeat*(" ABC ", ",", 2)); } @Test public void testFourth() { *assertEquals*("DBEDBE", StringUtils.*repeat*("DBE", "", 2)); } @Test public void testFifth() { *assertEquals*("DBE", StringUtils.*repeat*(" DBE ", ":", 1)); } @Test(expected = IllegalArgumentException.class) public void testSixth() { StringUtils.*repeat*("e", -2); } @Test public void testSeventh() { *assertEquals*("::", StringUtils.*repeat*("", ":", 3)); } @Test(expected = NullPointerException.class) public void testEighth() { StringUtils.*repeat*(null , "a", 1); } @Test(expected = NullPointerException.class) public void testNinth() { StringUtils.*repeat*("a", null , 2); }} **StringUtils.java**

package org.example;public class StringUtils { public static String repeat(String str, String separator, int repeat) { if (repeat < 0) { throw new IllegalArgumentException(); } if (str == null || separator == null) { throw new NullPointerException(); } if (repeat == 0) { return ""; } if (repeat == 1) { str = str.trim(); return str; } if (separator.equals(",")) { str = str.trim(); String temp = ""; for (int i = 1; i <= repeat; i++) { temp += str + " "; if (i < repeat) { temp += separator + " "; } } return temp; } if (str.equals("")) { str = str.trim(); for (int i = 1; i < repeat; i++) { str += separator; } return str; } if (separator.equals("")) { String temp = str; str = ""; str = str.trim(); for (int i = 0; i < repeat; i++) { str += temp; } return str; } String result = ""; for(int i = 1; i <= repeat; i++){ result += str; if(i < repeat){ result += separator; } } return result; } public static String repeat(String str, int repeat) { throw new IllegalArgumentException(); } static String repeat(String str) { throw new NullPointerException(); }}

**Задание 3**

Поиск ошибок, отладка и тестирование классов.

1) Импорт проекта Импортируйте один из проектов по варианту:

• Stack – проект содержит реализацию стека на основе связного списка: Stack.java.

2) Поиск ошибок

3) Внутренняя корректность

4) Реализация функциональности

5) Написание тестов

**Stack.java**

package org.example;import java.util.NoSuchElementException;public class Stack<Item> { private int size; private Node first; private class Node { private Item item; private Node next; } public Stack() { first = null; size = 0; assert check(); } public boolean isEmpty() { return (size < 0); } public int size() { return size; } public void push(Item item) { Node oldfirst = first; first = new Node(); first.item = item; first.next = oldfirst; size++; assert check(); } public Item pop() { if(size() == 0) { throw new NoSuchElementException(); } Item item = first.item; first = first.next; size--; assert check(); return item; } public Item peek() { if(size() == 0) { throw new NoSuchElementException(); } else { return first.item; } } public String toString() { StringBuilder s = new StringBuilder(); for (Node current = first; current != current.next; current = current.next) { Item item = current.item; s.append(item); if(null == current.next){ return s.toString(); } s.append(" - "); } return s.toString(); } private boolean check() { if (size == 0) { if (first != null) { return false; } } else if (size == 1) { if (first == null) { return false; } if (first.next != null) { return false; } } else { if (first.next == null) { return false; } } int numberOfNodes = 0; for (Node x = first; x != null; x = x.next) { numberOfNodes++; } if (numberOfNodes != size) { return false; } return true; }}

**StackClient.java**

package org.example;import java.util.Scanner;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()); }}

**StackTest.java**

package org.example.Test;import org.junit.runner.RunWith;import org.junit.runners.Suite;@RunWith( Suite. class )@Suite.SuiteClasses( { // Список тестовых классов в наборе для запуска StackTest1.class, StackTest2.class, StackTest3.class,})public class StackTest {}

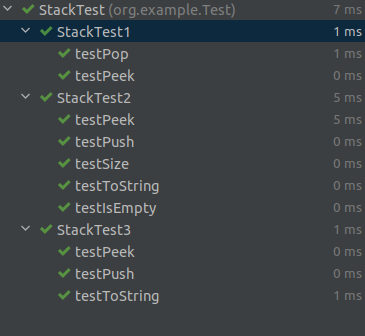
**StackTest1.java**

package org.example.Test;import org.example.Stack;import org.junit.Before;import org.junit.Test;public class StackTest1 { Stack<String> stack; @Before public void setUpBeforTest() { stack = new Stack<>(); } @Test(expected = java.util.NoSuchElementException.class) public void testPop() { stack.pop(); } @Test (expected = java.util.NoSuchElementException.class) public void testPeek() { stack.peek(); }} **StackTest2.java**

package org.example.Test;import org.example.Stack;import org.junit.Before;import org.junit.Test;import static org.junit.jupiter.api.Assertions.*assertEquals*;public class StackTest2 { Stack<String> stack; @Before public void setUpBeforTest() { stack = new Stack<>(); } @Test public void testIsEmpty() { *assertEquals*(false, stack.isEmpty()); } @Test public void testSize() { *assertEquals*(0, stack.size()); } @Test public void testPush() { stack.push("555"); *assertEquals*("555", stack.pop()); } @Test public void testPeek() { stack.push("555"); *assertEquals*("555", stack.peek()); } @Test public void testToString() { stack.push("555"); stack.push("444"); stack.push("333"); *assertEquals*("333 - 444 - 555", stack.toString()); }}

**StackTest3.java**

package org.example.Test;import org.example.Stack;import org.junit.Before;import org.junit.Test;import static org.junit.jupiter.api.Assertions.*assertEquals*;public class StackTest3 { Stack<String> stack; @Before public void setUpBeforTest() { stack = new Stack<>(); stack.push("555"); stack.push("444"); stack.push("333"); } @Test public void testPush() { *assertEquals*("333", stack.pop()); } @Test public void testPeek() { *assertEquals*("333", stack.peek()); } @Test public void testToString() { *assertEquals*("333 - 444 - 555", stack.toString()); }}

****

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