Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет» Кафедра ИИТ

Лабораторная работа №11

По дисциплине «СПП» за 6 семестр

Выполнил:

Студент группы ПО-3

Кабачук Д. С.

Проверил:

Крощенко А. А.

Брест 2021

**Вариант 11**

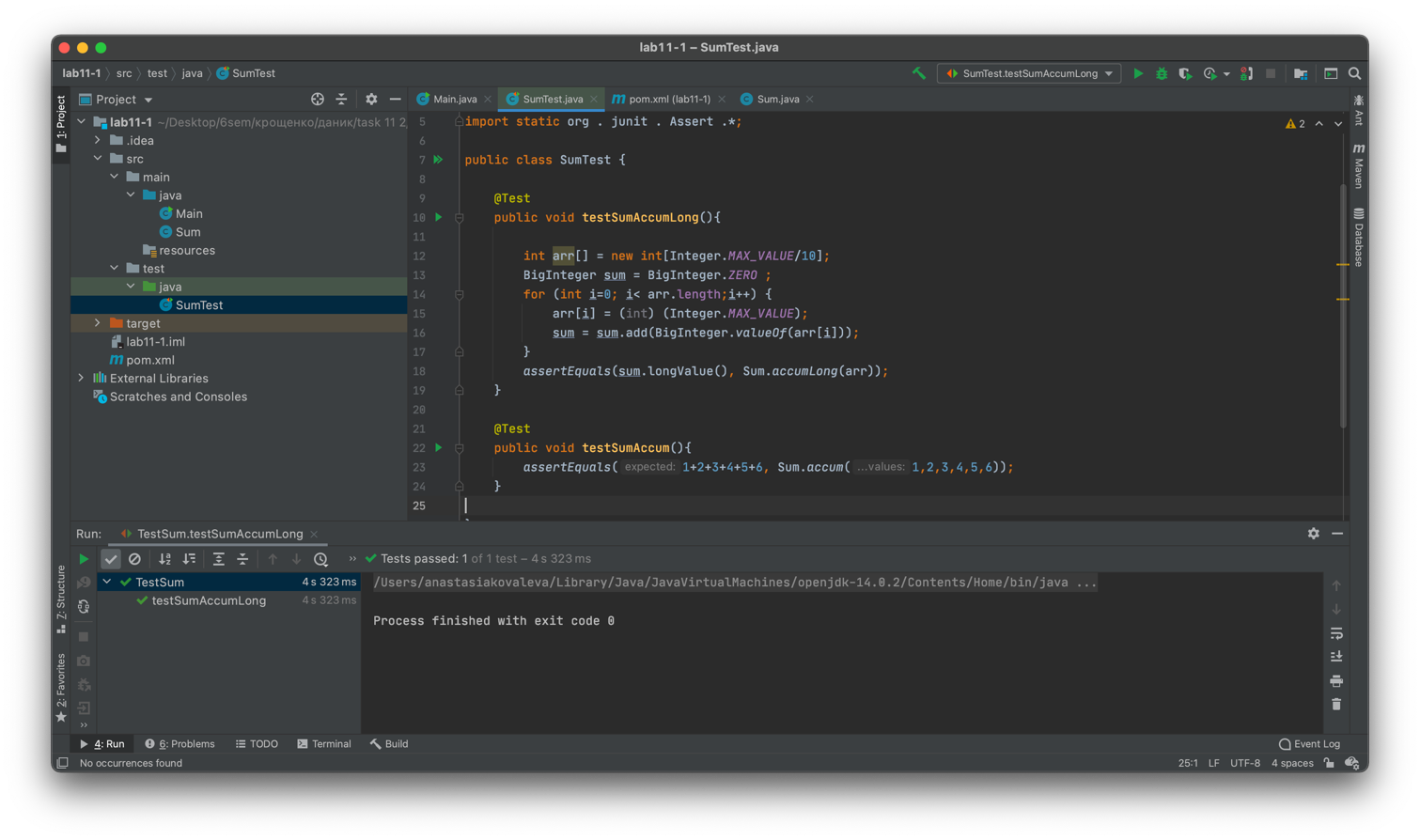
**Цель**: освоить приемы тестирования кода на примере использования библиотеки JUnit.

**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**

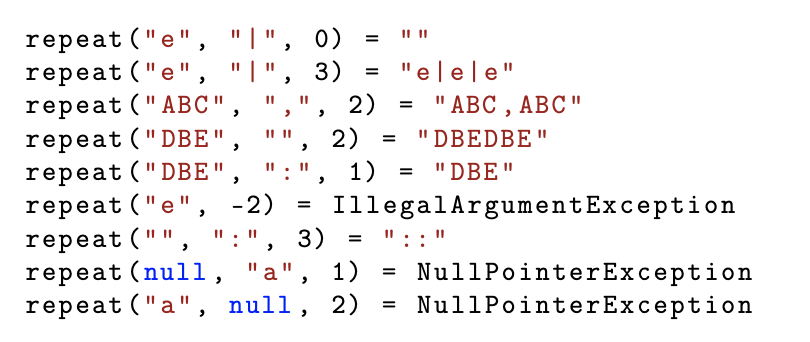
import org . junit .\*;  
  
import java.math.BigInteger;  
  
import static org . junit . Assert .\*;  
  
public class SumTest {  
  
 @Test  
 public void testSumAccumLong(){  
  
 int arr[] = new int[Integer.*MAX\_VALUE*/10];  
 BigInteger sum = BigInteger.*ZERO* ;  
 for (int i=0; i< arr.length;i++) {  
 arr[i] = (int) (Integer.*MAX\_VALUE*);  
 sum = sum.add(BigInteger.*valueOf*(arr[i]));  
 }  
 *assertEquals*(sum.longValue(), Sum.*accumLong*(arr));  
 }  
   
 @Test  
 public void testSumAccum(){  
 *assertEquals*(1+2+3+4+5+6, Sum.*accum*(1,2,3,4,5,6));  
 }  
  
}



**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.

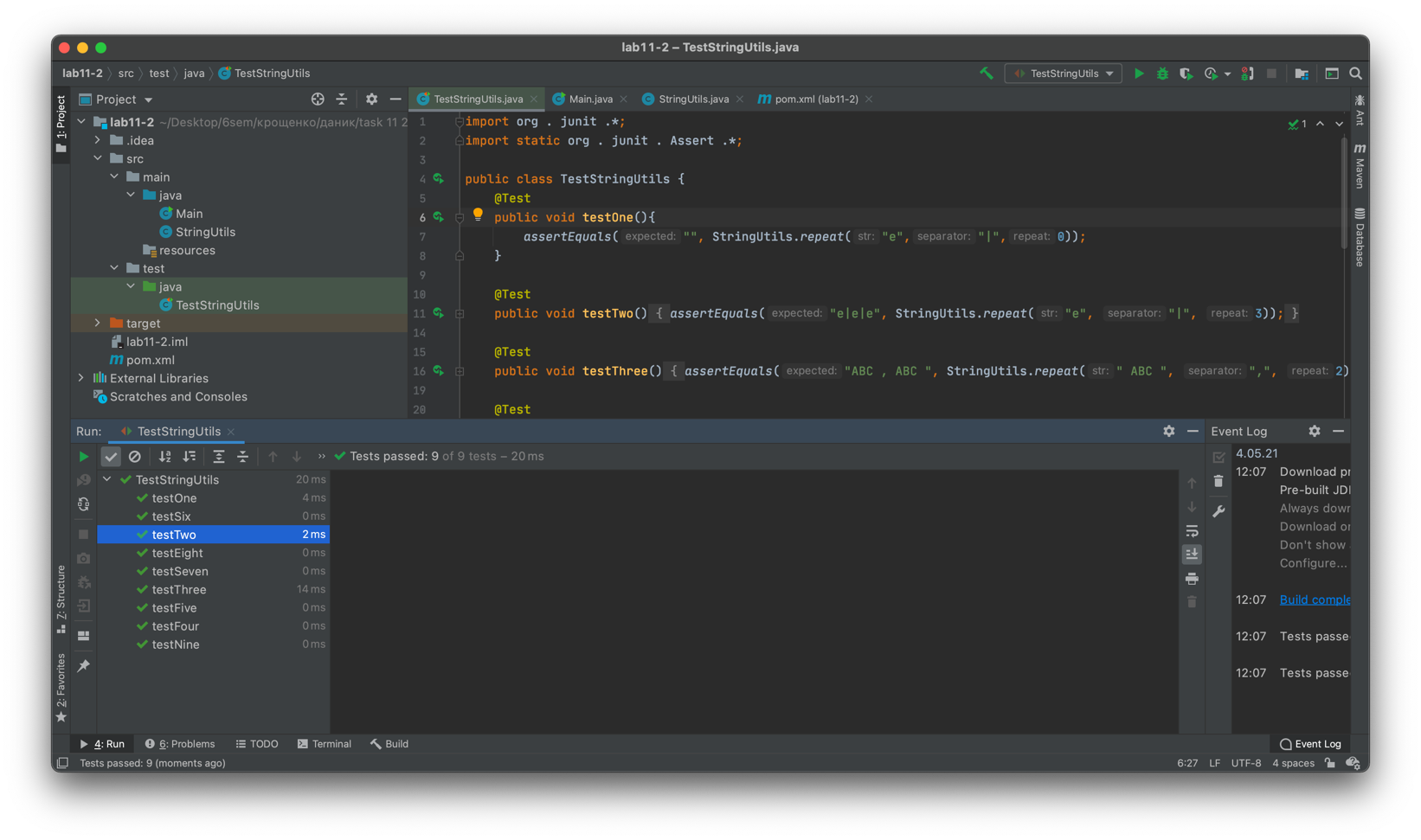
Спецификация метода:



import org . junit .\*;  
import static org . junit . Assert .\*;  
  
public class TestStringUtils {  
 @Test  
 public void testOne(){  
 *assertEquals*("", StringUtils.*repeat*("e","|",0));  
 }  
  
 @Test  
 public void testTwo(){  
 *assertEquals*("e|e|e", StringUtils.*repeat*("e", "|", 3));  
 }  
  
 @Test  
 public void testThree(){  
 *assertEquals*("ABC , ABC ", StringUtils.*repeat*(" ABC ", ",", 2));  
 }  
  
 @Test  
 public void testFour(){  
 *assertEquals*(" DBEDBE ", StringUtils.*repeat*(" DBE ", "", 2));  
 }  
  
 @Test  
 public void testFive(){  
 *assertEquals*("DBE", StringUtils.*repeat*(" DBE ", ":", 1));  
 }  
  
 @Test ( expected = IllegalArgumentException.class )  
 public void testSix(){  
 StringUtils.*repeat*("e", -2);  
 }  
  
 @Test  
 public void testSeven(){  
 *assertEquals*("::", StringUtils.*repeat*("", ":", 3));  
 }  
  
 @Test (expected = NullPointerException.class )  
 public void testEight(){  
 StringUtils.*repeat*(null , "a", 1);  
 }  
  
 @Test (expected = NullPointerException.class )  
 public void testNine(){  
 StringUtils.*repeat*("a", null , 2);  
 }  
}

**StringUtils**

public class StringUtils {  
  
 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 == 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(separator.equals("")){  
 String temp = "";  
 str = str.trim();  
 for (int i = 1; i <= repeat; i++){  
 if(i%2 != 0){  
 temp += " " + str;  
 }else {  
 temp += str + " ";  
 }  
 }  
 return temp;  
 }  
  
 String res = "";  
 for(int i = 1; i <= repeat; i++){  
 res += str;  
 if(i < repeat){  
 res += separator;  
 }  
 }  
  
 return res;  
 }  
  
 static String repeat(String str, int repeat){  
 throw new IllegalArgumentException();  
 }  
  
 static String repeat(String str){  
 throw new IllegalArgumentException();  
 }  
}

****

**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**

package stack;  
  
import java.util.NoSuchElementException;  
  
public class Stack<Item> {  
 private int N; // size of the stack  
 private Node first; // top of stack  
   
 private class Node {  
 private Item item;  
 private Node next;  
 }  
  
 public Stack() {  
 first = null;  
 N = 0;  
 assert check();  
 }  
  
 public boolean isEmpty() {  
 return (N < 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() {  
 // *FIXME throw exception if stack is Empty.* if(size() == 0){  
 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  
 }  
  
 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 (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;  
 }  
 }  
  
 int numberOfNodes = 0;  
 for (Node x = first; x != null; x = x.next) {  
 numberOfNodes++;  
 }  
 if (numberOfNodes != N) {  
 return false;  
 }  
  
 return true;  
 }  
}

**StackClient**

package stack;  
  
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());  
 }  
}

**RunAll**

package stack;  
  
import org . junit . runner . RunWith ;  
import org . junit . runners . Suite ;  
import org . junit . runners . Suite . SuiteClasses ;  
@RunWith ( Suite . class ) // Запустить класс как тестовый набор  
@SuiteClasses ({ // Список тестовых классов в наборе для запуска  
  
 StackTest1.class ,  
 StackTest2 .class ,  
 StackTest3 .class ,  
})  
class RunAll {  
  
}

**StackTest1**

package stack;  
  
import org . junit .\*;  
import static org . junit . Assert .\*;  
  
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**

package stack;  
  
import org . junit .\*;  
import static org . junit . Assert .\*;  
  
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**

package stack;  
  
import org . junit .\*;  
import static org . junit . Assert .\*;  
  
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.