Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №11**

По дисциплине «СПП»

за 6-й семестр

Выполнил:

студент 3 курса

группы ПО-5

Харкевич Д.А.

Проверил:

Крощенко А.А.

Брест, 2022

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

**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.

**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.

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.

**Задание 3 – Поиск ошибок, отладка и тестирование классов**

1) Импорт проекта Импортируйте один из проектов по варианту:

• Queue – содержит реализацию очереди на основе связного списка: Queue.java.

Написание тестов

• Все функции вашего класса должны быть покрыты тестами.

• Использовать фикстуры для инициализации начального состояния объекта.

• Итого, должно быть несколько тестовых классов, в каждом из которых целевая структура данных создается в фикстуре в некотором инициализированном состоянии (пустая, заполненная и тд), а после очищается.

• Написать тестовый набор, запускающий все тесты.

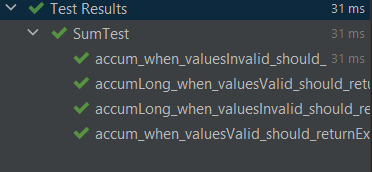
**Задание 1**

**Код програмы:**

**sumTest.java**

package spp.task1;  
  
import org.junit.jupiter.api.Test;  
  
import static spp.task1.Sum.*accum*;  
import static org.junit.Assert.*assertEquals*;  
import static spp.task1.Sum.*accumLong*;  
  
public class SumTest {  
 @Test  
 public void accum\_when\_valuesValid\_should\_returnExceptedValue() {  
 var result = *accum*(1, 2, 5);  
 *assertEquals*(result, 8);  
 }  
  
 @Test  
 public void accum\_when\_valuesInvalid\_should\_returnExceptedValue() {  
 var result = *accum*(Integer.*MAX\_VALUE*, 12, 5);  
 *assertEquals*(result, Integer.*MIN\_VALUE* + 12 + 5 - 1);  
 }  
  
 @Test  
 public void accumLong\_when\_valuesValid\_should\_returnExceptedValue() {  
 var result = *accumLong*(1, 2, 5);  
 *assertEquals*(result, 8);  
 }  
  
 @Test  
 public void accumLong\_when\_valuesInvalid\_should\_returnExceptedValue() {  
 var result = *accumLong*(Integer.*MAX\_VALUE*, 12, 5);  
 *assertEquals*(result, (long) (Integer.*MAX\_VALUE*) + (long) 12 + (long) 5);  
 }  
}

**Результаты:**

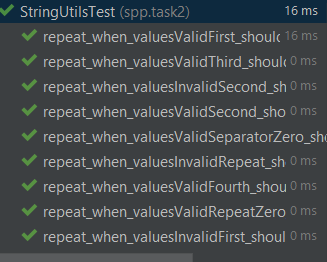


**Задание 2**

**Код программы:**

package spp.task2;  
  
import org.junit.Test;  
import static org.junit.Assert.*assertEquals*;  
  
import static spp.task2.StringUtils.*repeat*;  
  
public class StringUtilsTest {  
 @Test  
 public void repeat\_when\_valuesValidRepeatZero\_should\_returnExceptedValue() throws Exception {  
 var result = *repeat*("e", "|", 0);  
 *assertEquals*(result, "");  
 }  
  
 @Test  
 public void repeat\_when\_valuesValidFirst\_should\_returnExceptedValue() throws Exception {  
 var result = *repeat*("e", "|", 3);  
 *assertEquals*(result, "e|e|e");  
 }  
  
 @Test  
 public void repeat\_when\_valuesValidSecond\_should\_returnExceptedValue() throws Exception {  
 var result = *repeat*("ABC", ",", 2);  
 *assertEquals*(result, "ABC,ABC");  
 }  
  
 @Test  
 public void repeat\_when\_valuesValidSeparatorZero\_should\_returnExceptedValue() throws Exception {  
 var result = *repeat*("DBE", "", 2);  
 *assertEquals*(result, "DBEDBE");  
 }  
  
 @Test  
 public void repeat\_when\_valuesValidThird\_should\_returnExceptedValue() throws Exception {  
 var result = *repeat*("DBE", ":", 1);  
 *assertEquals*(result, "DBE");  
 }  
  
 @Test  
 public void repeat\_when\_valuesValidFourth\_should\_returnExceptedValue() throws Exception {  
 var result = *repeat*("", ":", 3);  
 *assertEquals*(result, "::");  
 }  
  
 @Test(expected = IllegalArgumentException.class)  
 public void repeat\_when\_valuesInvalidRepeat\_should\_returnExceptedValue() throws Exception {  
 var result = *repeat*("AV", "DV", -5);  
 }  
  
 @Test(expected = NullPointerException.class)  
 public void repeat\_when\_valuesInvalidFirst\_should\_returnException() throws Exception {  
 var result = *repeat*(null, "a", 1);  
 }  
  
 @Test(expected = NullPointerException.class)  
 public void repeat\_when\_valuesInvalidSecond\_should\_returnException() throws Exception {  
 var result = *repeat*("a", null, 2);  
 }  
}

**Результаты:**

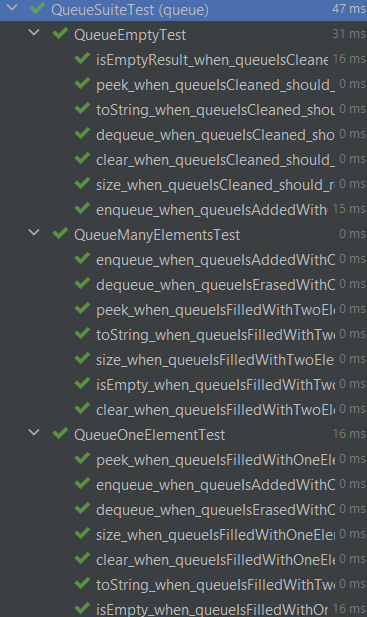
****

**Задание 3**

**Код программы:**

package queue;  
  
import org.junit.After;  
import org.junit.Test;  
  
import static org.junit.Assert.\*;  
  
public class QueueEmptyTest {  
 Queue<String> queue = new Queue<>();  
  
 @After  
 public void queueClear() throws Exception {  
 queue.clear();  
 }  
  
 @Test  
 public void clear\_when\_queueIsCleaned\_should\_returnEmptyResult() {  
 queue.clear();  
 *assertTrue*(queue.isEmpty());  
 *assertEquals*(0, queue.size());  
 *assertEquals*("", queue.toString());  
 }  
  
 @Test  
 public void isEmptyResult\_when\_queueIsCleaned\_should\_returnIsEmpty() {  
 *assertTrue*(queue.isEmpty());  
 }  
  
 @Test  
 public void size\_when\_queueIsCleaned\_should\_returnNullSize() {  
 *assertEquals*(0, queue.size());  
 }  
  
 @Test(expected = java.util.NoSuchElementException.class)  
 public void peek\_when\_queueIsCleaned\_should\_returnException() {  
 queue.peek();  
 }  
  
 @Test  
 public void enqueue\_when\_queueIsAddedWithOneElement\_should\_returnWorkingResult() {  
 queue.enqueue("Test1");  
 *assertFalse*(queue.isEmpty());  
 *assertEquals*(1, queue.size());  
 *assertEquals*("Test1 ", queue.toString());  
 }  
  
 @Test(expected = java.util.NoSuchElementException.class)  
 public void dequeue\_when\_queueIsCleaned\_should\_returnException() {  
 queue.dequeue();  
 }  
  
 @Test  
 public void toString\_when\_queueIsCleaned\_should\_returnEmptyString() {  
 *assertEquals*("", queue.toString());  
 }  
}

**Результаты:**

****

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