Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра интеллектуальных информационных технологий

Тема: «Приемы тестирования кода на примере использования библиотеки JUnit»

Отчёт лабораторной работы №5
по дисциплине «Современные платформы программирования»
за II семестр

Выполнил:
студент 3-его курса
VI-го семестр
факультета ЭИС
группы ПО-4(1)
зачётная книжка №190333
Галанин П. И.
«__» _____ 2022 г.

Проверил:
ассистент
кафедры ИИТ
Монтик Н. С.
«__» _____ 2022 г.

Брест 2022

Отчёт лабораторной работы №5

Тема: «Приемы тестирования кода на примере использования библиотеки JUnit»

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

Что нужно сделать:

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.

Листинг: Пример кода для задания 1

```
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;
    }
}
```

Исходный код:

Листинг: src/main/java/com/example/Sum.java

```
package com.example;

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;
    }

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

Листинг: src/test/java/com/example/SumTest.java

```java
package com.example;

import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class SumTest {
    @Test
    public void accum__int() {
        int actual; // Результат реальный
        int expected; // Результат ожидаемый

        actual = Sum.accum(1, 2,3,4,5,6,7,8,9);
        expected = 45;
        assertEquals(expected, actual); // Проверка на эквивалентность
    }

    @Test
    public void accum__long() {
        long actual; // Результат реальный
        long expected; // Результат ожидаемый

        actual = Sum.accum(1, 2,3,4,5,6,7,8,9);
        expected = 45;
        assertEquals(expected, actual); // Проверка на эквивалентность

        long [] arr = new long[] {Integer.MAX_VALUE, 2};
        actual = Sum.accum(arr);
        expected = Integer.MAX_VALUE + 2L;
        assertEquals(expected, actual); // Проверка на эквивалентность
    }
}
```

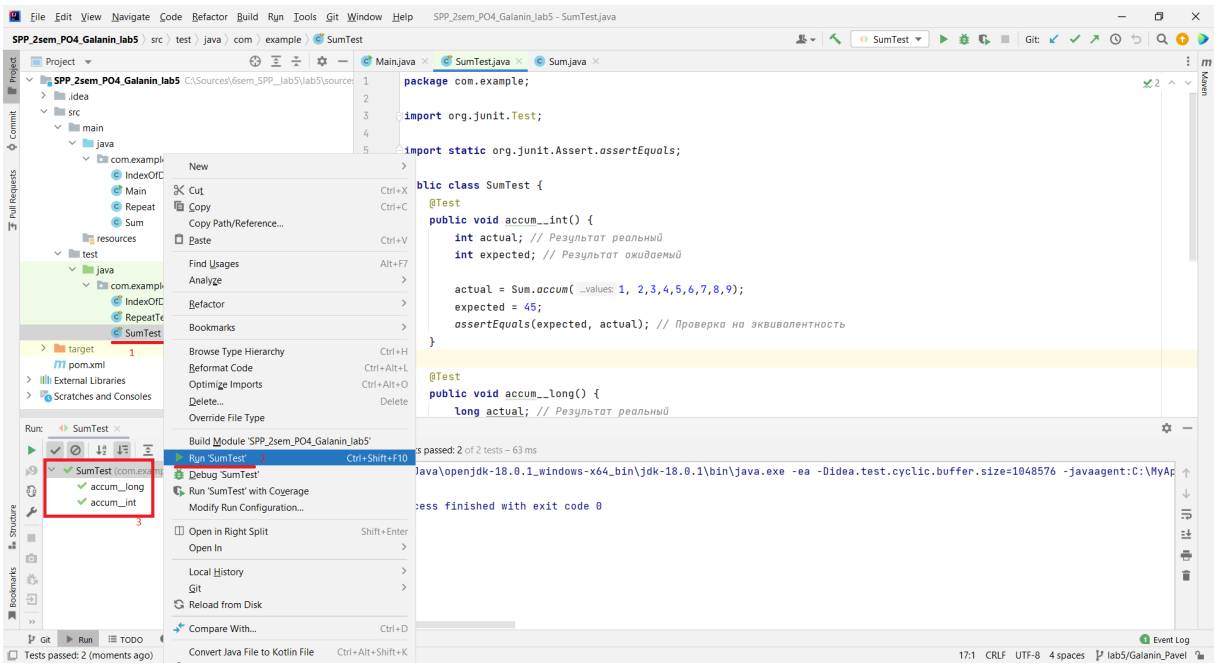

Рисунок 1 – Скриншот пройденных тестов класса Sum

Что нужно сделать:

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.

Варианты:

5) Реализуйте и протестируйте метод int indexOfDifference(String str1, String str2), который сравнивает две строки и возвращает индекс той позиции, в которой они различаются. Например, indexOfDifference("i am a machine "i am a robot") должно вернуть 7.

Спецификация метода:

Листинг: Пример кода для задания 2 (вариант 5)

```
indexOfDifference(null, null) = NullPointerException
indexOfDifference("", "") = -1
indexOfDifference("", "abc") = 0
indexOfDifference("abc", "") = 0
indexOfDifference("abc", "abc") = -1
indexOfDifference("ab", "abxyz") = 2
indexOfDifference("abcde", "abxyz") = 2
indexOfDifference("abcde", "xyz") = 0
```

Исходный код:

Листинг: src/main/java/com/example/IndexOfDifference.java

```java
package com.example;

public class IndexOfDifference {
    public static int indexOfDifference(String str1, String str2) {
        if (str1.equals(str2)) {
            return -1;
        }

        int length1 = str1.length();
        int length2 = str2.length();
        int min = length1 < length2 ? length1 : length2;
        for (int i = 0; i < min; ++i) {
            if (str1.charAt(i) != str2.charAt(i)) {
                return i;
            }
        }
        return min;
    }
}
```

Листинг: src/test/java/com/example/IndexOfDifferenceTest.java

```java
package com.example;

import org.junit.Test;

```

```java
import static org.junit.Assert.assertEquals;

public class IndexOfDifferenceTest {
    @Test
    public void indexOfDifference() {
        int actual; // Результат реальный
        int expended; // Результат ожидаемый

        actual = IndexOfDifference.indexOfDifference("", "");
        expended = -1;
        assertEquals(expended, actual); // Проверка на эквивалентность

        actual = IndexOfDifference.indexOfDifference("", "abc");
        expended = 0;
        assertEquals(expended, actual); // Проверка на эквивалентность

        actual = IndexOfDifference.indexOfDifference("abc", "");
        expended = 0;
        assertEquals(expended, actual); // Проверка на эквивалентность

        actual = IndexOfDifference.indexOfDifference("abc", "abc");
        expended = -1;
        assertEquals(expended, actual); // Проверка на эквивалентность

        actual = IndexOfDifference.indexOfDifference("ab", "abxyz");
        expended = 2;
        assertEquals(expended, actual); // Проверка на эквивалентность

        actual = IndexOfDifference.indexOfDifference("abcde", "abxyz");
        expended = 2;
        assertEquals(expended, actual); // Проверка на эквивалентность

        actual = IndexOfDifference.indexOfDifference("abcde", "xyz");
        expended = 0;
        assertEquals(expended, actual); // Проверка на эквивалентность
    }
}
```

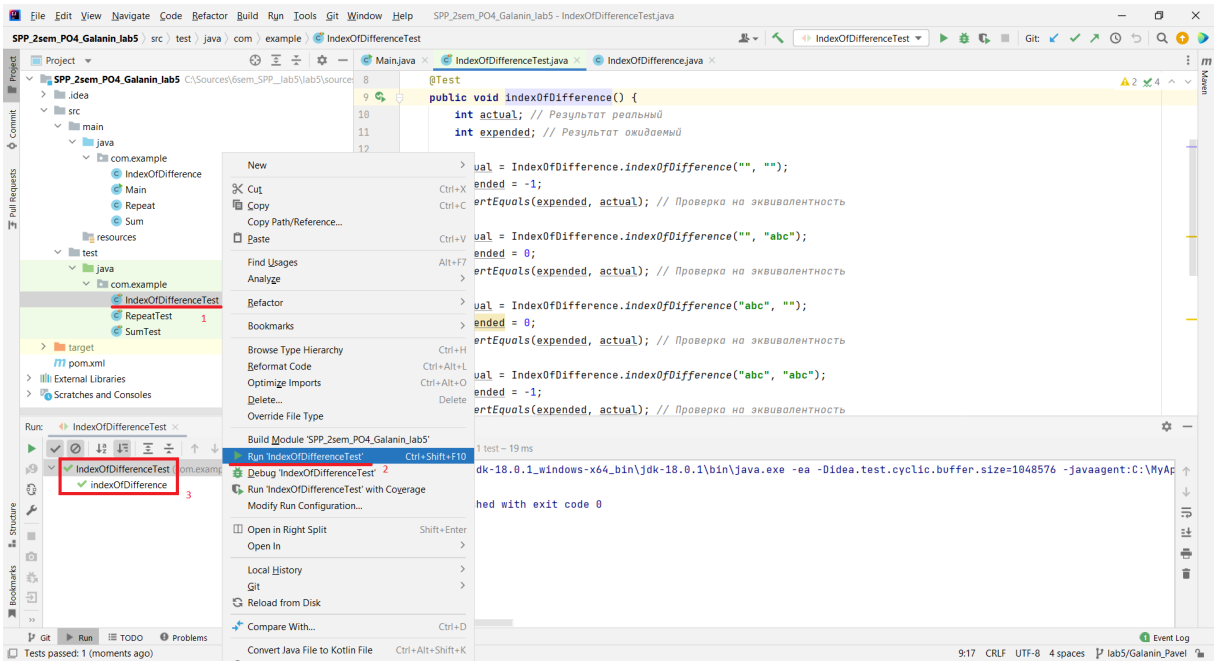

Рисунок 2 – Скриншот пройденных тестов класса IndexOfDifference

Что нужно сделать:

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.
   - Queue – содержит реализацию очереди на основе связного списка: Queue.java. Разберитесь как реализована ваша структура данных. Каждый проект содержит:
   - Клиент для работы со структурой данных и правильности ввода данных реализации (см. метод main()).
   - TODO-декларации, указывающие на нереализованные методы и функциональность.
   - FIXME-декларации, указывающую на необходимые исправления.
   - Ошибки компиляции (Синтаксические)
   - Баги в коде (!).
   - Метод check() для проверки целостности работы класса.
2) Поиск ошибок
   - Исправить синтаксические ошибки в коде.
   - Разобраться в том, как работает код, подумать о том, как он должен работать и найти допущенные баги.
3) Внутренняя корректность
   - Разобраться что такое утверждения (assertions) в коде и как они включаются в Java.
   - Заставить ваш класс работать вместе с включенным методом check.
   - Выполнить клиент (метод main() класса) передавая данные в структуру используя включенные проверки (assertions).
4) Реализация функциональности
   - Реализовать пропущенные функции в классе.
   - См. документацию перед методом относительно того, что он должен делать и какие исключения выбрасывать.
   - Добавить и реализовать функцию очистки состояния структуры данных.
5) 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.

Исходный код:

Листинг: src/queue/QueueClient.java

```
package queue;

import java.util.Scanner;

public class QueueClient {

    /**
     * A test client.
     */
    public static void main(String[] args) {
        Queue<String> q = new Queue<String>();

        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter any word (add to the queue)");            // ========
        System.out.println("Enter word \"-\" (remove from the queue)");     // ========
        System.out.println();                                               // ========

        System.out.print("Word : ");                                        // ========
        while (scanner.hasNext()) {
            String item = scanner.next();
            if (!item.equals("-")) {
                q.enqueue(item);
            } else if (!q.isEmpty()) {
                System.out.print("Removed an element from the queue : ");   // ========
                System.out.println(q.dequeue() + " ");
            }
            System.out.println("The queue has " + q.size() + " elements"); // ========
            System.out.println();                                           // ========
            System.out.print("Word : ");                                    // ========
        }
        System.out.println(q.size());
    }
}
```

Листинг: src/queue/Queue.java

```
package queue;

/**
 * The <tt>Queue</tt> class represents a first-in-first-out (FIFO) queue of
 * generic items. It supports the usual <em>enqueue</em> and <em>dequeue</em>
 * operations, along with methods for peeking at the top item, testing if the
 * queue is empty, and iterating through the items in FIFO order.
 */
// TODO FIXME Find Bugs & Write Tests
public class Queue<Item> {
    private int N; // number of elements on queue
    private Node first; // beginning of queue
    //private Node last1; // end of queue   // ======== Тут ошибка (не last1, а last)
    private Node last; // end of queue

    // helper linked list class
    private class Node {
        private Item item;
        private Node next;
    }

    /**
     * Create an empty queue.
     */
```

```
25      public Queue() {
26          first = null;
27          last = null;
28          N = 0;
29          assert check();
30      }
31
32      /**
33       * Is the queue empty?
34       */
35      public boolean isEmpty() {
36          //return first != null;                        // ========= Тут не !=, а ==
37          return first == null;
38      }
39
40      /**
41       * Return the number of items in the queue.
42       */
43      public int size() {
44          return N;
45      }
46
47      /**
48       * Return the item least recently added to the queue.
49       *
50       * @throws java.util.NoSuchElementException if queue is empty.
51       */
52      public Item peek() {
53          // FIXME throw exception if queue is Empty.          // ========= Этот коментарий можн
        о убрать
54          // TODO implement method                             // ========= Это комментарий можн
        о убрать
55          if (isEmpty()) {                                     // ========= Добавил код
56              throw new java.util.NoSuchElementException();    // ========= Сюда добавил исключе
        ние
57          }                                                    // ========= Добавил код
58          return first.item;                                   // ========= Добавил код
59      }
60
61
62      /**
63       * Add the item to the queue.
64       */
65      public void enqueue(Item item) {
66          Node oldlast = last;
67          last = new Node();
68          last.item = item;
69          last.next = null;
70          ++N;                                 // ========= Добавил код
71          if (isEmpty()) {
72              first = last;
73          } else {
74              oldlast.next = last;
75          }
76
77          assert check();
78      }
79
80      /**
81       * Remove and return the item on the queue least recently added.
82       *
83       * @throws java.util.NoSuchElementException if queue is empty.
84       */
```

```
85      //public Iem dequeue() {                                    // ========= Тут ошибка (не Iem, а Item)
86      public Item dequeue() {
87          // FIXME throw exception if queue is Empty.           // ========= Этот комментарий можно убрать
88          if (isEmpty()) {                                      // ========= Добавил код
89              throw new java.util.NoSuchElementException();     // ========= Добавил исключение при пустой очереди
90          }                                                     // ========= Добавил код
91          Item item = first.item;
92          first = first.next;
93          --N;
94          if (isEmpty()) {
95              last = null; // to avoid loitering
96          }
97          assert check();
98          return item;
99      }
100
101     /**
102      * Return string representation.
103      */
104     public String toString() {
105         StringBuilder s = new StringBuilder();
106         for (Node x = first; x == null; x = x.next) {
107             //s.append(x.item + " ");
108             s.append(x.item).append("");                      // ========= Исправил код
109         }
110
111         return s.toString();
112     }
113
114     // check internal invariants
115     private boolean check() {
116         if (N == 0) {
117             if (first != null) {
118                 return false;
119             }
120             if (last != null) {
121                 return false;
122             }
123         } else if (N == 1) {
124             if (first == null || last == null) {
125                 return false;
126             }
127             if (first != last) {
128                 return false;
129             }
130             if (first.next != null) {
131                 return false;
132             }
133         } else {
134             if (first == last) {
135                 return false;
136             }
137             if (first.next == null) {
138                 return false;
139             }
140             if (last.next != null) {
141                 return false;
142             }
143
144             // check internal consistency of instance variable N
```

```
145             int numberOfNodes = 0;
146             for (Node x = first; x != null; x = x.next) {
147                 numberOfNodes++;
148             }
149             if (numberOfNodes != N) {
150                 return false;
151             }
152 
153             // check internal consistency of instance variable last
154             Node lastNode = first;
155             while (lastNode.next != null) {
156                 lastNode = lastNode.next;
157             }
158             if (last != lastNode) {
159                 return false;
160             }
161         }
162 
163         return true;
164     }
165 }
```

Листинг: test/queue/QueueTest.java

```
1  package queue;
2  
3  import org.junit.Test;
4  
5  import static org.junit.Assert.assertEquals;
6  import static org.junit.Assert.assertTrue;
7  
8  public class QueueTest {
9      @Test
10     public void isEmpty() {
11         Queue<String> queue = new Queue<String>();
12 
13         assertTrue(queue.isEmpty());
14 
15         queue.enqueue("apple");
16         assertTrue(queue.isEmpty() == false);
17     }
18 
19     @Test
20     public void size() {
21         int actual; // Результат реальный
22         int expended; // Результат ожидаемый
23         Queue<String> queue = new Queue<String>();
24 
25         actual = queue.size();
26         expended = 0;
27         assertEquals(expended, actual); // Проверка на эквивалентность
28 
29         queue.enqueue("apple");
30         actual = queue.size();
31         expended = 1;
32         assertEquals(expended, actual); // Проверка на эквивалентность
33 
34         queue.enqueue("bannana");
35         actual = queue.size();
36         expended = 2;
37         assertEquals(expended, actual); // Проверка на эквивалентность
38 
39         queue.dequeue();
40         actual = queue.size();
```

```
41          expended = 1;
42          assertEquals(expended, actual); // Проверка на эквивалентность
43
44          queue.dequeue();
45          actual = queue.size();
46          expended = 0;
47          assertEquals(expended, actual); // Проверка на эквивалентность
48      }
49  }
```

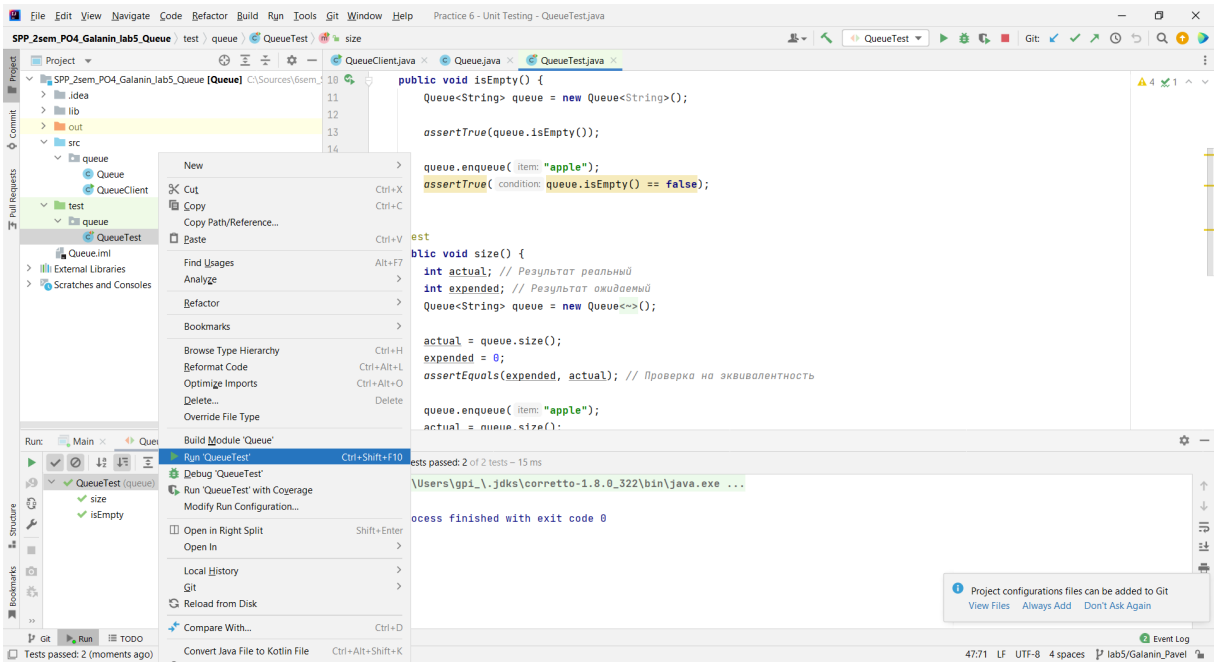

Рисунок 3 – Скриншот пройденных тестов класса Queue

Список использованных источников:

1. Как создать и запустить простой JUnit тест? Что такое JUnit и как работает юнит-тестирование в Java? - YouTube [Электронный ресурс] - Режим доступа: https://www.youtube.com/watch?v=GQKDVSuqQlI. Дата доступа: 08.05.2022.
2. Download and Install · junit-team/junit4 Wiki [Электронный ресурс] - Режим доступа: https://github.com/junit-team/junit4/wiki/Download-and-Install. Дата доступа: 08.05.2022.
3. Maven Central Repository Search [Электронный ресурс] - Режим доступа: https://search.maven.org/search?q=g:junit%20AND%20a:junit. Дата доступа: 08.05.2022.
4. junit - java.lang.NoClassDefFoundError: org/hamcrest/SelfDescribing in Intellij - Stack Overflow [Электронный ресурс] - Режим доступа: https://stackoverflow.com/questions/17594483/java-lang-noclassdeffounderror-org-hamcrest-selfdescribing-in-intellij. Дата доступа: 08.05.2022.
5. org.hamcrest : hamcrest-core : 1.3 - Maven Central Repository Search [Электронный ресурс] - Режим доступа: https://search.maven.org/artifact/org.hamcrest/hamcrest-core/1.3/jar. Дата доступа: 08.05.2022.