МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №11

Специальность ПО5

Выполнил:
А.А. Игнатюк,
студент группы ПО-5

Проверил:
А.А. Крощенко,
ст. преп. кафедры ИИТ,
«__» _____ 2022 г.

Брест 2022

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

Вариант 5.

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 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.

```
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;
    }
}
```

Рисунок 1.1 - Исходный код класса Sum.

Спецификация ввода: Указанные в тестах значения
Спецификация вывода: Результаты тестирования

Код программы и результаты тестирования:

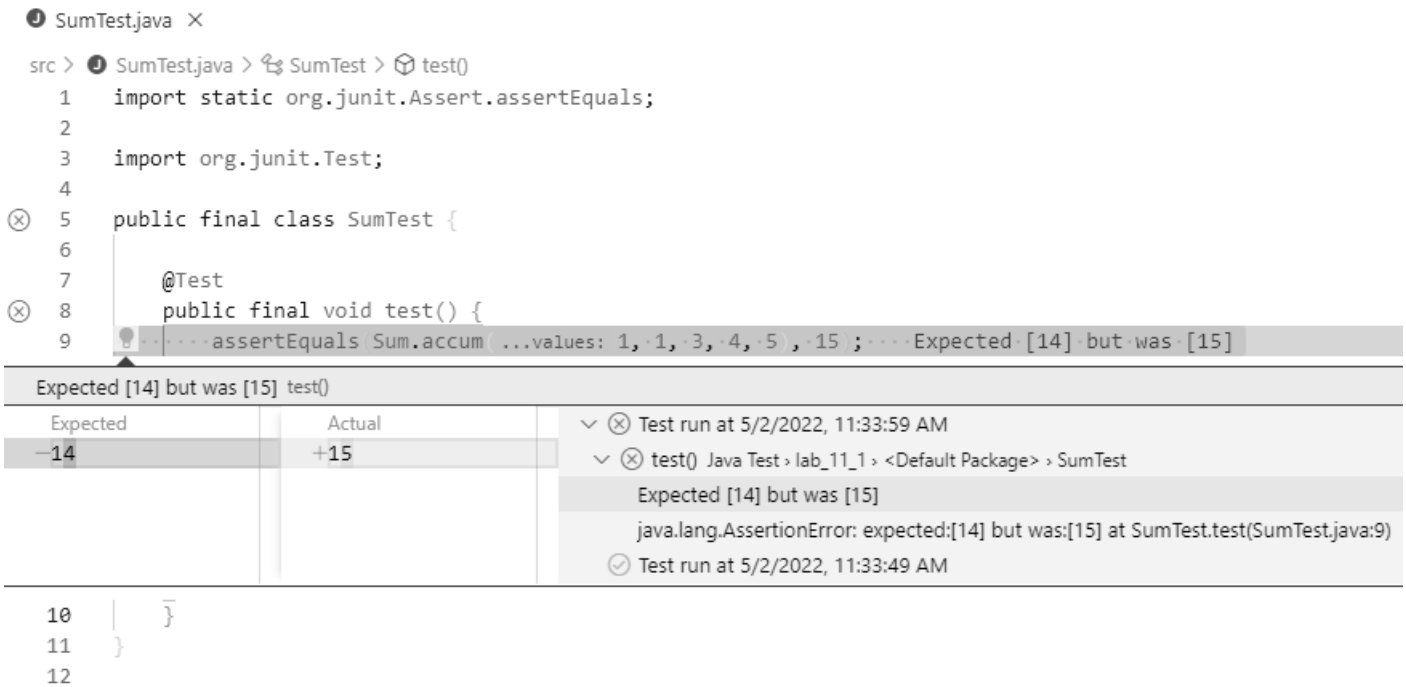


Рисунок 1.2 - Тестирование класса Sum (ошибка).

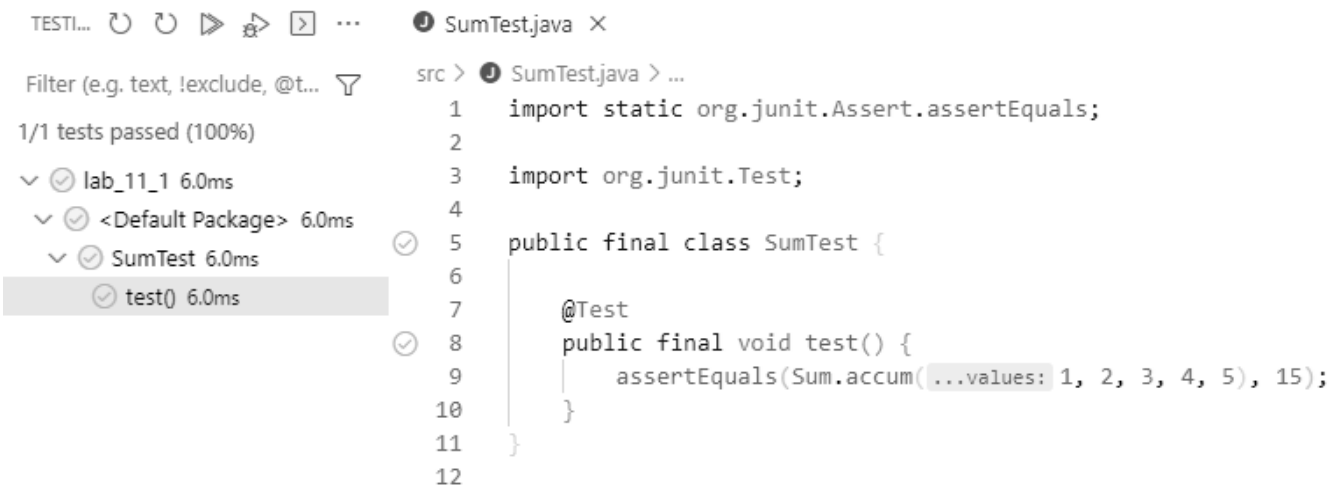


Рисунок 1.3 - Тестирование класса Sum (успех).

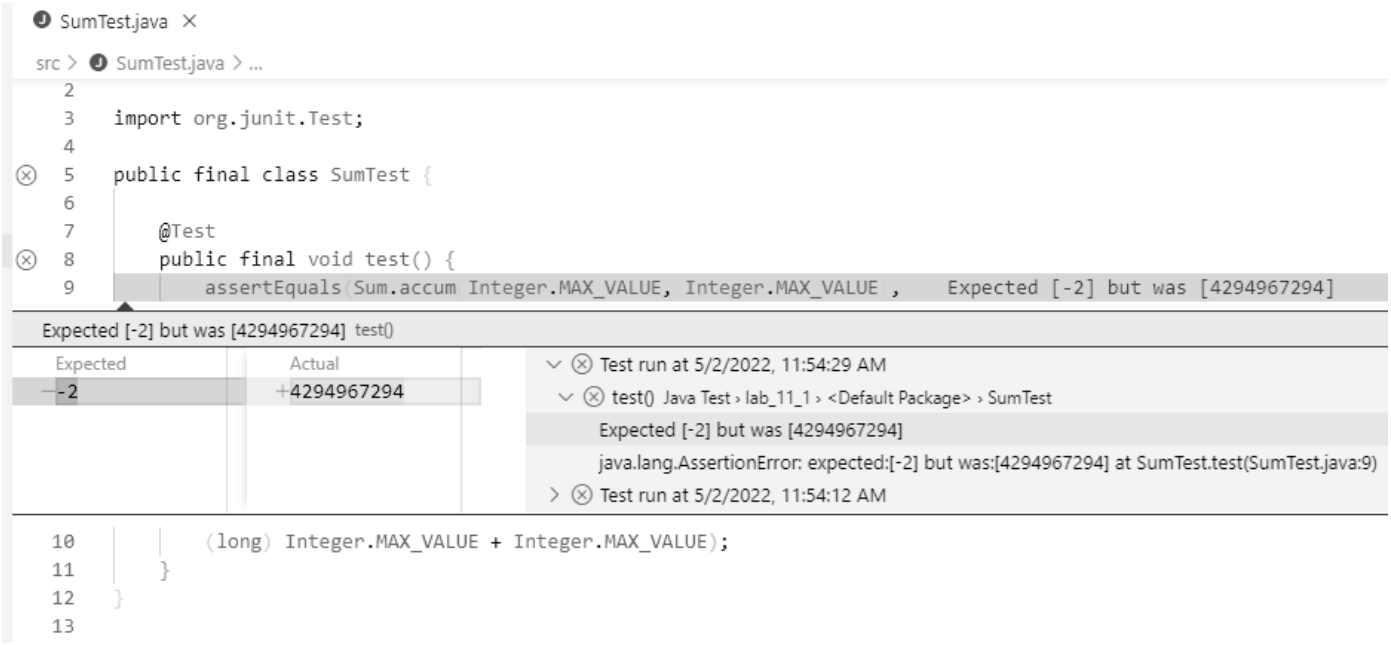


Рисунок 1.4 - Тестирование класса Sum (переполнение).

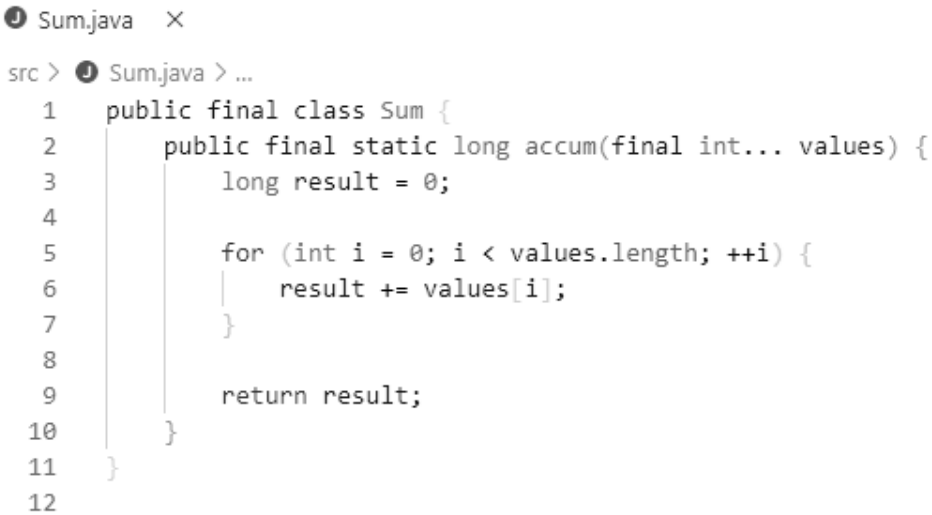


Рисунок 1.5 - Модифицированный исходный код класса Sum.

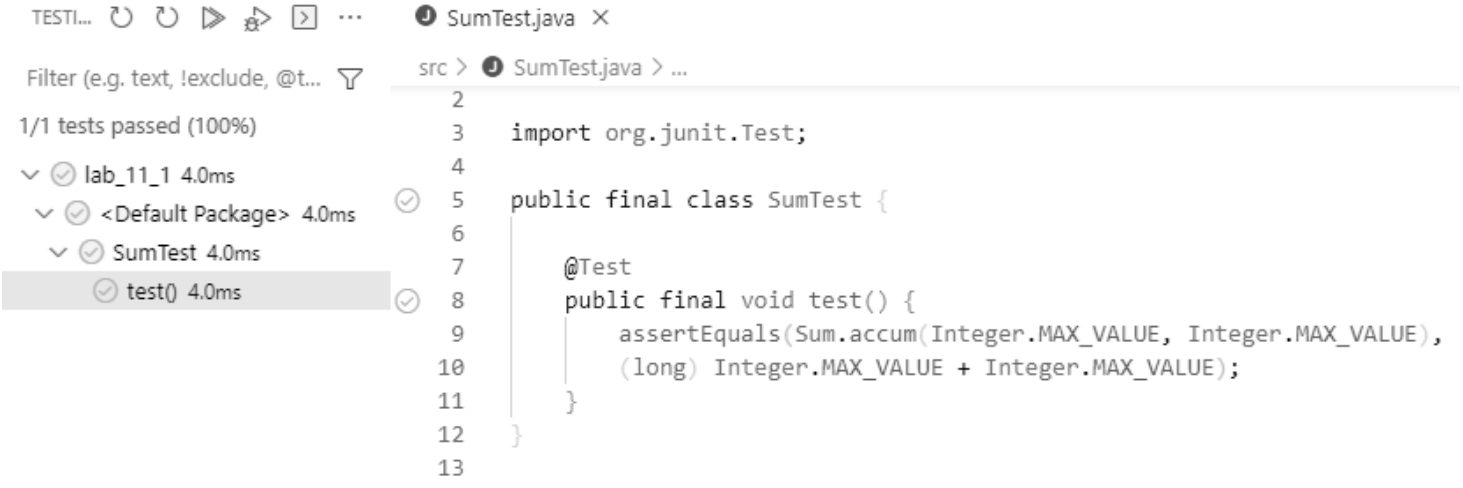


Рисунок 1.6 - Тестирование класса Sum (переполнение отсутствует).

Задание 2.

• 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:

```
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
```

Рисунок 2.1 - Спецификация метода.

Спецификация ввода: Указанные в тестах значения
Спецификация вывода: Результаты тестирования

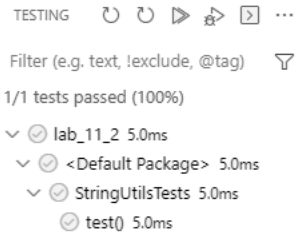
Код программы и результаты тестирования:

StringUtils.java

src > StringUtils.java > ...

```java
public final class StringUtils {
    public final static int indexOfDifference(final String str1, final String str2) throws NullPointerException {
        if (str1 == null || str2 == null) {
            throw new NullPointerException();
        }

        if (str1.equals( anObject: "") && str2.equals( anObject: "")) {
            return -1;
        }

        int i = 0;

        for (; i < str1.length() && i < str2.length(); ++i) {
            if (str1.charAt(i) != str2.charAt(i)) {
                return i;
            }
        }

        return i;
    }
}

```

Рисунок 2.2 - Исходный код файла StringUtils.java.



StringUtilsTests.java

src > StringUtilsTests.java > ...

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

import org.junit.Test;
import org.junit.function.ThrowingRunnable;

public final class StringUtilsTests {
    @Test
    public final void test() {
        assertThrows(NullPointerException.class, new ThrowingRunnable() {
            @Override
            public void run() throws Throwable {
                StringUtils.indexOfDifference( str1: null, str2: null);
            }
        });

        assertThrows(NullPointerException.class, new ThrowingRunnable() {
            @Override
            public void run() throws Throwable {
                StringUtils.indexOfDifference( str1: "i am a machine", str2: null);
            }
        });

        assertThrows(NullPointerException.class, new ThrowingRunnable() {
            @Override
            public void run() throws Throwable {
                StringUtils.indexOfDifference( str1: null, str2: "i am a robot");
            }
        });

        assertEquals(StringUtils.indexOfDifference( str1: "", str2: ""), -1);
        assertEquals(StringUtils.indexOfDifference( str1: "i am a machine", str2: ""), 0);
        assertEquals(StringUtils.indexOfDifference( str1: "", str2: "i am a robot"), 0);
        assertEquals(StringUtils.indexOfDifference( str1: "ab", str2: "abxyz"), 2);
        assertEquals(StringUtils.indexOfDifference( str1: "abcde", str2: "abxyz"), 2);
        assertEquals(StringUtils.indexOfDifference( str1: "abcde", str2: "xyz"), 0);
        assertEquals(StringUtils.indexOfDifference( str1: "i am a machine", str2: "i am a robot"), 7);
    }
}

```

Рисунок 2.3 - Исходный код файла StringUtilsTests.java.

Задание 3.

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.

Спецификация ввода: Пользовательский ввод строковых значений в консоль, указанные в тестах значения

Спецификация вывода: Вывод, хранящихся в стеке, значений в консоль, результаты тестирования

Код программы и результаты тестирования:

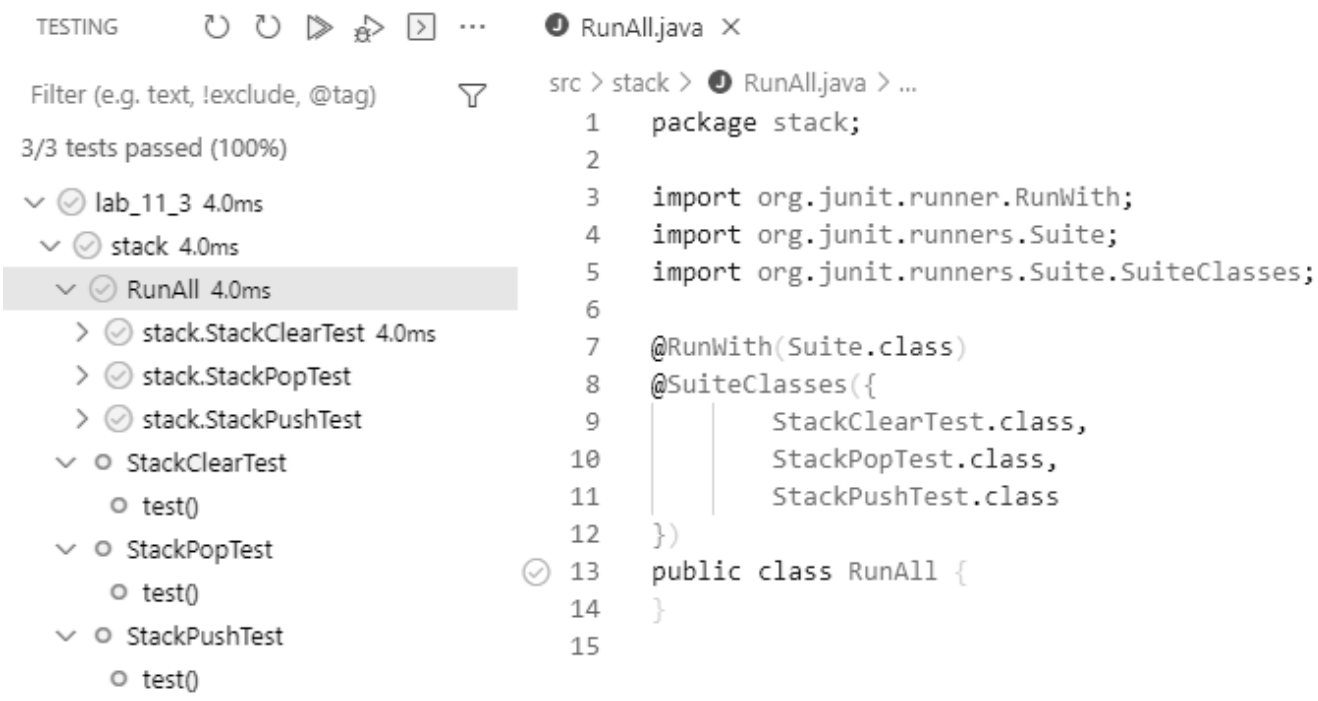


Рисунок 3.1 - Исходный код файла RunAll.java.

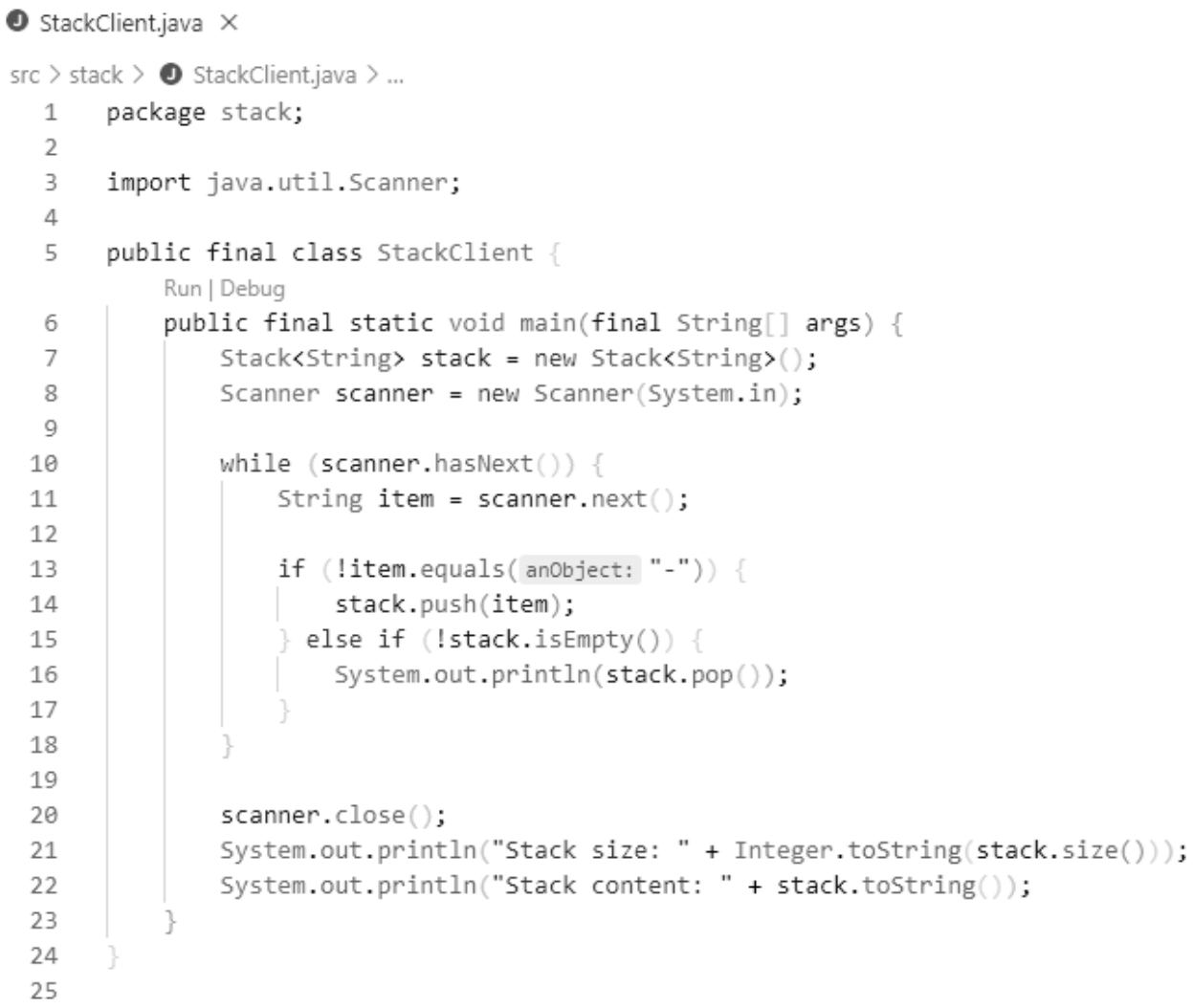


Рисунок 3.2 - Исходный код файла StackClient.java.

Stack.java

src > stack > Stack.java > ...

```java
package stack;

import java.util.NoSuchElementException;

public final class Stack<Item> {
    private Integer N = 0;
    private Node first = null;

    private final class Node {
        private Item item;
        private Node next;

        public Node(final Item item, final Node next) {
            this.item = item;
            this.next = next;
        }
    }

    public Stack() {
        assert check();
    }

    public final Boolean isEmpty() {
        return this.N == 0;
    }

    public final Integer size() {
        return this.N;
    }

    public final void push(final Item item) {
        Node oldfirst = this.first;
        first = new Node(item, oldfirst);
        ++this.N;

        assert check();
    }

    public final Item pop() {
        if (this.isEmpty()) {
            throw new NoSuchElementException();
        }

        Item item = this.first.item;
        this.first = this.first.next;
        --this.N;

        assert check();
        return item;
    }

    public final Item peek() {
        if (this.isEmpty()) {
            throw new NoSuchElementException();
        }

        Item item = this.first.item;

        assert check();
        return item;
    }

    public final String toString() {
        StringBuilder sb = new StringBuilder();

        for (Node current = this.first; current != null; current = current.next) {
            Item item = current.item;
            sb.append(item + " ");
        }

        return sb.toString();
    }

```

Рисунок 3.3 - Исходный код файла Stack.java.

Продолжение рисунка 3.3.

```
74      public final void clear() {
75          while (!this.isEmpty()) {
76              this.pop();
77          }
78      }
79
80      private final Boolean check() {
81          if (this.N == 0 && this.first != null) {
82              return false;
83          }
84
85          if (this.N != 0 && this.first == null) {
86              return false;
87          }
88
89          if (this.N > 1 && this.first.next == null) {
90              return false;
91          }
92
93          Integer checkN = 0;
94
95          for (Node x = this.first; x != null; x = x.next) {
96              ++checkN;
97          }
98
99          if (this.N != checkN) {
100             return false;
101         }
102
103         return true;
104     }
105 }
106
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\Documents\Visual Studio Code\Java\lab_11_3>  & 'C:\Program Files\Eclipse Foundation\jdk-11.0.12.7-hotspo
t\bin\java.exe' '@C:\Users\User\AppData\Local\Temp\cp_78sjyxqenhu36qyr22xnqvh8h.argfile' 'stack.StackClient'
a
b
c
d
-
d
-
c
-
b
-
a
-
x
y
z
-
z
Stack size: 2
Stack content: y x
PS D:\Documents\Visual Studio Code\Java\lab_11_3>
```

Рисунок 3.4 - Результат работы кода StackClient.

StackClearTest.java

src > stack > StackClearTest.java > ...

```java
package stack;

import static org.junit.Assert.assertEquals;

import java.util.ArrayList;
import java.util.Arrays;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;

public final class StackClearTest {
    Stack<String> stack = null;
    ArrayList<String> stackData = null;

    @Before
    public void beforTest() {
        this.stack = new Stack<String>();
        this.stackData = new ArrayList<String>(Arrays.asList( ...a: "a", "b", "c", "d", "e"));

        for (final String item : stackData) {
            this.stack.push(item);
        }
    }

    @After
    public void afterTest() {
        this.stack.clear();
        this.stackData.clear();
    }

    @Test
    public final void test() {
        this.stack.clear();
        assertEquals(this.stack.isEmpty(), actual: true);
    }
}

```

Рисунок 3.5 - Исходный код файла StackClearTest.java.

StackPopTest.java

src > stack > StackPopTest.java > ...

```java
package stack;

import static org.junit.Assert.assertEquals;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;

public final class StackPopTest {
    Stack<String> stack = null;
    ArrayList<String> stackData = null;

    @Before
    public void beforTest() {
        this.stack = new Stack<String>();
        this.stackData = new ArrayList<String>(Arrays.asList( ...a: "a", "b", "c", "d", "e"));

        for (final String item : stackData) {
            this.stack.push(item);
        }

        Collections.reverse(this.stackData);
    }

    @After
    public void afterTest() {
        this.stack.clear();
        this.stackData.clear();
    }

    @Test
    public final void test() {
        for (int i = 0, size = this.stackData.size(); i < size; ++i) {
            assertEquals(this.stack.isEmpty(), actual: false);
            assertEquals(this.stack.size(), this.stackData.size() - i);
            assertEquals(this.stack.peek(), this.stackData.get(i));
            this.stack.pop();
        }

        assertEquals(this.stack.isEmpty(), actual: true);
    }
}

```

Рисунок 3.6 - Исходный код файла StackPopTest.java.

StackPushTest.java

src > stack > StackPushTest.java > ...

```java
package stack;

import static org.junit.Assert.assertEquals;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;

public final class StackPushTest {
    Stack<String> stack = null;
    ArrayList<String> stackData = null;

    @Before
    public void beforTest() {
        this.stack = new Stack<String>();
        this.stackData = new ArrayList<String>(Arrays.asList(...a: "a", "b", "c", "d", "e"));
    }

    @After
    public void afterTest() {
        this.stack.clear();
        this.stackData.clear();
    }

    @Test
    public final void test() {
        assertEquals(this.stack.isEmpty(), actual: true);

        for (int i = 0, size = this.stackData.size(); i < size; ++i) {
            this.stack.push(this.stackData.get(i));
            assertEquals(this.stack.isEmpty(), actual: false);
            assertEquals(this.stack.size(), i + 1);
            assertEquals(this.stack.peek(), this.stackData.get(i));
        }

        Collections.reverse(this.stackData);
        StringBuilder sb = new StringBuilder();

        for (final String item : stackData) {
            sb.append(item + " ");
        }

        assertEquals(this.stack.toString(), sb.toString());
    }
}
```

Рисунок 3.7 - Исходный код файла StackPushTest.java.

Вывод: Освоил приемы тестирования кода на примере использования библиотеки JUnit.