МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»

Институт компьютерных наук и кибербезопасности

Высшая школа технологий искусственного интеллекта

Отчёт по дисциплине «Методы тестирования ПО »

Лабораторная работа №4 «Автоматизированное тестирования ПО»

Студент: _____ Салимли Айзек Мухтар Оглы

Преподаватель: _____ Курочкин Михаил Александрович

«____»_____ 20___ г.

Санкт-Петербург, 2025

Содержание

Введение

Тестирование - это процесс выполнения программы с целью обнаружения ошибок. Ошибки бывают двух видов:

1. Программа не делает то, что от неё требуется или делает это с ошибками.
2. Программа делает то, что от неё не требуется.

Задача тестирования - как можно более экономически эффективно и в ограниченные сроки обнаружить в программном обеспечении максимально возможное количество несоответствий спецификации (ошибок).

Автоматизированное тестирование - это метод тестирования программного обеспечения, который предполагает использование инструментов и фреймворков автоматизации для выполнения одного и того же набора тест-кейсов снова и снова. Ключевое различие между ручным и автоматизированным тестированием заключается в том, что ручное тестирование полностью зависит от человека, сидящего за компьютером. В то время как автоматизированные тесты могут быть написаны один раз и выполняться многократно практически без участия человека.

Автоматизированное тестирование программного обеспечения позволяет значительно ускорить процесс проверки, сокращая время тестирования с недели до нескольких часов даже при большом объеме тест-кейсов. Это не только повышает эффективность работы, но и снижает человеческий фактор, который может привести к пропуску ошибок из-за усталости или отвлечения. Автоматизация также позволяет стандартизировать тесты, гарантируя, что каждый сценарий будет проверен одинаково независимо от исполнителя. Это важно для обеспечения качества и уменьшения субъективности в результатах тестирования.

1 Постановка задачи

Цели работы:

- Разработать юнит-тесты для четырех методов библиотеки calculator.jar, используя JUnit, и загрузить результаты на GitHub, создав pull-request для проверки.
- Используя TestNG и Selenium WebDriver для тестирования сайта в Safari.
- Необходимо написать два теста, проверяющих отображение страниц, организовать тесты по Java Code Convention, и запустить их через TestNG suitexml.

2 Описание средств автоматизации тестирования

2.1 JUnit

JUnit - это фреймворк для модульного тестирования приложений на языке Java. Он позволяет разработчикам писать и выполнять повторяемые тесты для проверки корректности работы кода. **Основные особенности JUnit:**

- Аннотации: JUnit предоставляет аннотации, такие как @Test, @Before, @After, @BeforeClass, @AfterClass, которые помогают в организации тестов.
- Простота в использовании: Легкий в освоении и использовании, что делает его популярным среди разработчиков.
- Поддержка параметризованных тестов: С помощью аннотаций @ParameterizedTest и @ValueSource можно передавать параметры в тесты.
- Совместимость с различными средами разработки: JUnit интегрируется с большинством IDE, таких как InteliJ IDEA, Eclipse, NetBeans.

Этапы написания тестов:

1. Реализация теста: Написание тестового метода и аннотирование его с помощью @Test.
2. Настройка и очистка: Использование аннотаций @Before и After для выполнения операций перед и после теста.
3. Запуск тестов: Использование встроенных средств IDE или командной строки для выполнения тестов.

2.2 Selenium

Selenium WebDriver - это библиотека для автоматизации работы с веб-браузерами. Она позволяет писать тесты, которые взаимодействуют с веб-страницами так же, как это делал бы пользователь. **Основные особенности Selenium WebDriver:**

- Поддержка различных браузеров: WebDriver работает с множеством браузеров, включая Chrome, Firefox, Edge, Safari и Opera.
- Многоязычность: Поддержка различных языков программирования, таких как Java, C#, Python, Ruby, JavaScript.
- Интерактивность: WebDriver отправляет команды непосредственно браузеру и получает ответы, что позволяет точнее воспроизводить действия пользователя.
- Расширяемость: Поддержка дополнительных функций через различные плагины и библиотеки.

Основные компоненты:

- WebDriver: Управляет браузером и выполняет команды тестов.
- WebElement: Представляет элементы веб-страницы, с которыми взаимодействует тест.
- By: Используется для поиска элементов на странице с помощью различных стратегий локаторов (CSS-селекторы, XPath и т.д.).

Этапы работы с WebDriver:

1. Инициализация: Создание экземпляра WebDriver и открытие браузера.
2. Навигация: Переход к необходимой веб-странице.
3. Интеракции: Взаимодействие с элементами страницы (клики, ввод текста и т.д.).

4. Проверки: Выполнение утверждений для проверки состояния элементов и результатов операций.
5. Завершение: Закрытие браузера и завершение сеанса WebDriver.

3 Описание выполнения работ

3.1 Лабораторная работа: JUnit тест

В лабораторной работе необходимо создать юниттесты для библиотеки calculator.jar, выбрав четыре метода. Необходимо использовать JUnit, включая аннотации @Before и @After для пред и постобработки, а также @Parameterized Test и @ValueSource для параметризации тестов. Нужно создать и добавить репозиторий на GitHub, добавить библиотеку и pom.xml, выполнить работу в новой ветке, затем создать pull-request.

3.1.1 Класс TestCalculator

Класс "TestCalculator"инициализирует объект 'Calculator' и задает допустимую погрешность, чтобы подготовить среду для выполнения юнит-тестов.

Листинг 1: Класс calculator

```
public class TestCalculator {
        protected static Calculator cale;
        protected static final double TOLERANCE = 0.1;

        @Before All
        public static void setUp(); {
                cale = new Calculator ();
        }
}
```

3.1.2 Класс TestSinus

Класс "TestSinus' выполняет юнит-тесты для метода вычисления синуса в классе 'Calculator'. Он проверяет корректность вычисления синуса для различных углов, симметричность функции синуса для положительных и отрицательных углов, а также периодичность функции синуса при добавлении 2π к углу. Тесты параметризованы с использованием аннотаций 'ParameterizedTest' и 'OCsvSource'

Листинг 2: Класс TestSinus

```
\begin{lstlisting}[caption={TestSinus}]
import org.junit.jupiter.params.ParameterizedTest;
import org.junit.jupiter.params.provider.CsvSource;
import static org.junit.jupiter.api.Assertions.assertEquals;

class TestSinus extends TestCalculator {
        @ParameterizedTest
    @CsvSource({
        "0,    0",
        "0.48, 0.5",
        "0.765,0.75",
        "0.89, 1",
        "1,    1.57",
        "0,    3.14",
        "-0.99,4.71"
    })
    void testCorrectnessOfSinusCalculation(double expectedOutcome, double
        angleInRadians) {
        double result = calc.sin(angleInRadians);
        assertEquals(expectedOutcome, result, TOLERANCE,
            "Computed sine value does not match expected.");
    }
```

```
22
23      @ParameterizedTest
24      @CsvSource({
25          "0,     0",
26          "0.5,  -0.5",
27          "0.75, -0.75",
28          "1,    -1",
29          "1.57, -1.57",
30          "3.14, -3.14",
31          "4.71, -4.71"
32      })
33      void testSinusFunctionSymmetry(double positiveAngle, double
            negativeAngle) {
34          double sinPositive = calc.sin(positiveAngle);
35          double sinNegative = calc.sin(negativeAngle);
36          assertEquals(-sinPositive, sinNegative, TOLERANCE,
37              "Sine of negative angle does not equal negative sine of the
                    corresponding positive angle.");
38      }
39
40      @ParameterizedTest
41      @CsvSource({
42          "0", "1.047", "2.094", "3.142", "4.189", "5.236"
43      })
44      void testSinusFunctionPeriodicity(double angle) {
45          double expected = calc.sin(angle);
46          double actual   = calc.sin(angle + 2 * Math.PI);
47          assertEquals(expected, actual, TOLERANCE,
48              "Sine of the angle does not remain unchanged after adding 2pi.")
                    ;
49      }
50  }
```

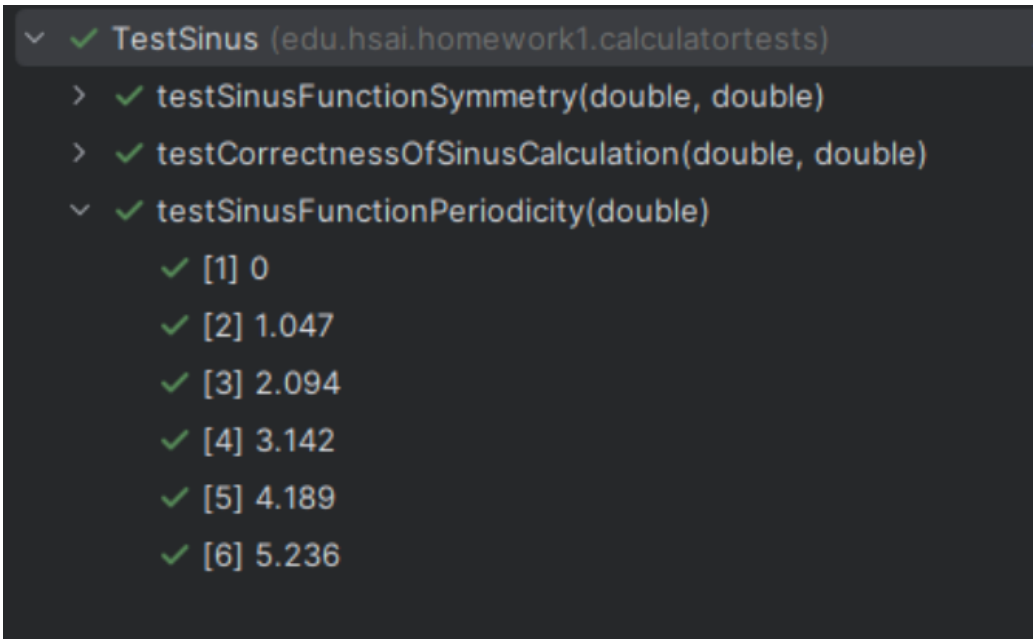

Рис. 1: Итоги теста TestSinus

3.1.3 Класс TestSumDouble

Класс 'TestSumDouble' выполняет юнит-тесты для метода сложения в классе 'Calculator' Он проверяет корректность сложения для граничных значений,общей корректности функции сложения, а также изменение знака через ноль. Тесты параметризованы с использованием аннотаций 'ParameterizedTest' и '@CsvSource'

Листинг 3: Класс TestSumDouble

```
import org.junit.jupiter.params.ParameterizedTest;
import org.junit.jupiter.params.provider.CsvSource;
import static org.junit.jupiter.api.Assertions.assertEquals;

class TestSumDouble extends TestCalculator {

    @ParameterizedTest
    @CsvSource({
        "1.7976931348623157E308,1.7976931348623157E308,0.0",
        "-1.7976931348623157E308,-1.7976931348623157E308,0.0"
    })
    void testBoundaryValues(double expected, double a, double b) {
        assertEquals(expected, calc.sum(a, b));
    }

    @ParameterizedTest
    @CsvSource({
        "23.0,17.0,6.0",
        "-10.0,-8.0,-2.0"
    })
    void testBasicCorrectness(double expected, double a, double b) {
        assertEquals(expected, calc.sum(a, b));
    }

    @ParameterizedTest
    @CsvSource({
        "1.0,-1.0,2.0",
        "-1.0,1.0,-2.0"
    })
    void testSignChangeThroughZero(double expected, double a, double b) {
        assertEquals(expected, calc.sum(a, b));
    }
}
```

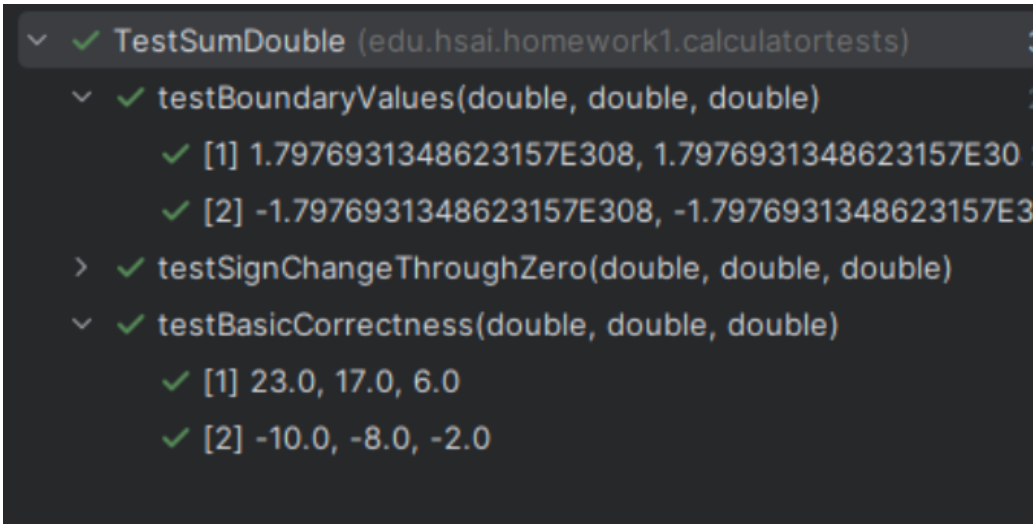

Рис. 2: Итоги теста TestSumDouble

3.1.4 Класс TanTest

Класс "TanTests' выполняет юнит-тесты для метода вычисления тангенса в классе 'Calculator'. Он проверяет, что тангенс стремится к бесконечности при углах, близких к $\frac{\pi}{2}$ и $\frac{\pi}{3}$, где косинус равен нулю, а также тестирует значения тангенса для выбранных углов. Тесты параметризованы с использованием аннотаций '@ParameterizedTest', '@CsvSource' и '®ValueSource'.

Листинг 4: Класс TestCalculator

```
import org.junit.jupiter.params.ParameterizedTest;
import org.junit.jupiter.params.provider.ValueSource;
import org.junit.jupiter.params.provider.CsvSource;
import static org.junit.jupiter.api.Assertions.assertEquals;

class TanTests extends TestCalculator {
    @ParameterizedTest
    @ValueSource(doubles = {1.57, 4.71})
    void testTangentApproachingInfinity(double angle) {
        double computedValue = calc.tg(angle);
        assertEquals(Double.POSITIVE_INFINITY, computedValue, TOLERANCE,
            "Tangent should approach infinity.");
    }

    @ParameterizedTest
    @CsvSource({
        "0,    0",
        "0.58, 0.5",
        "1.00, 0.78",
        "1.74, 1.05"
    })
    void verifyTangentValuesForSpecificAngles(double expectedOutcome, double
        angleInRadians) {
        double result = calc.tg(angleInRadians);
        assertEquals(expectedOutcome, result, TOLERANCE,
            "Calculated tangent value does not match the expected value.");
    }
}
```

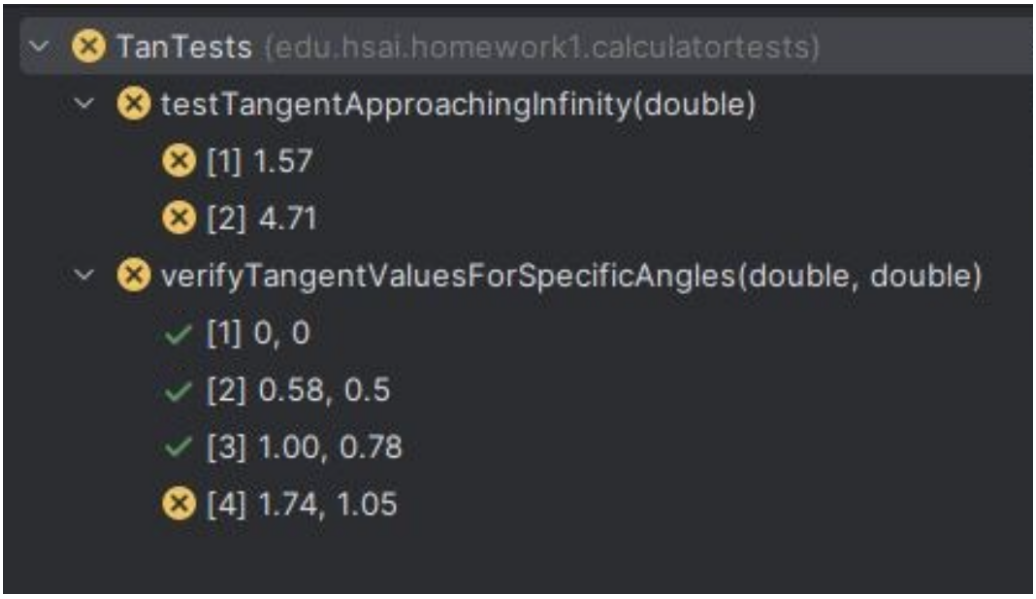

Рис. 3: Итоги теста TanTest

3.1.5 Класс DivDoubleTest

Класс 'DivDoubleTest выполняет юнит-тесты для метода деления в классе "Calculator". Он проверяет корректность деления для различных случаев: общую корректность операции, деление на ноль и деление числа самого на себя. Тесты параметризованы с использованием аннотаций '@ParameterizedTest' и '@CsSource'

Листинг 5: Класс DivDoubleTest

```
import org.junit.jupiter.params.ParameterizedTest;
import org.junit.jupiter.params.provider.CsvSource;
import static org.junit.jupiter.api.Assertions.assertEquals;

class DivDoubleTests extends TestCalculator {
    @ParameterizedTest
    @CsvSource({
        "10.0,    100.0, 10.0",
        "-4.5,    13.5,  -3.0",
        "0.0,     0.0,   15.0",
        "Infinity,1.0,   0.0"
    })
    void testGeneralCorrectness(double expected, double a, double b) {
        assertEquals(expected, calc.div(a, b), 0.0001);
    }

    @ParameterizedTest
    @CsvSource({
        "Infinity,  1.0,  0.0",
        "-Infinity, -1.0, 0.0",
        "NaN,       0.0,  0.0"
    })
    void testDivisionByZero(double expected, double a, double b) {
        assertEquals(expected, calc.div(a, b), 0.0);
    }

    @ParameterizedTest
    @CsvSource({
        "1.0,   100.0, 100.0",
        "-1.0,  -10.0,  10.0"
```

```
31      })
32      void testDivisionByItself(double expected, double a, double b) {
33          assertEquals(expected, calc.div(a, b), 0.0001);
34      }
35  }
```

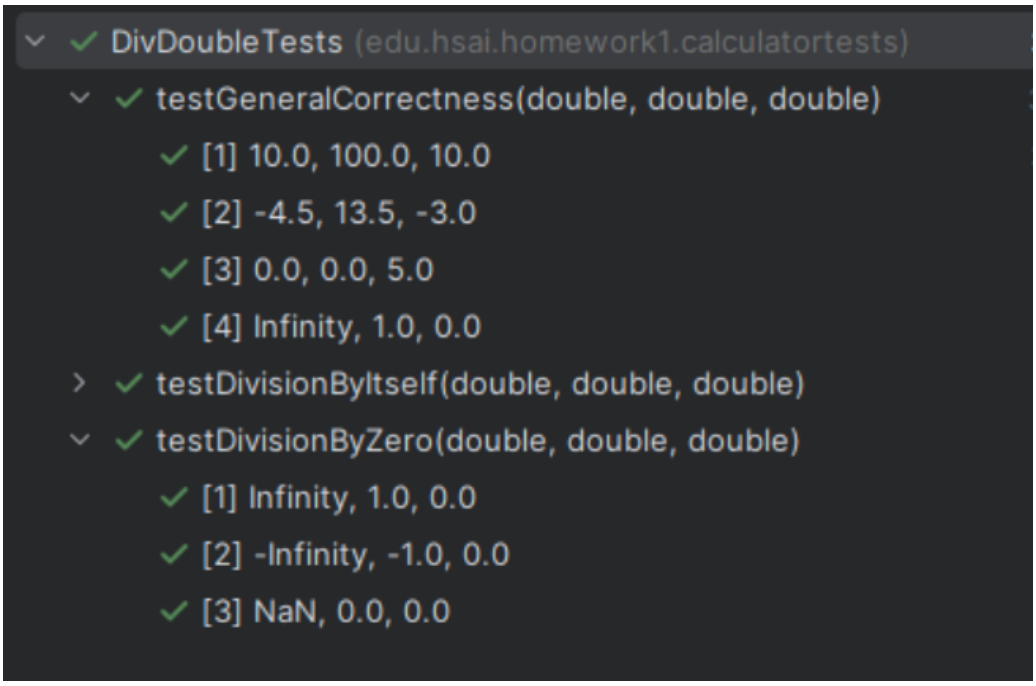

Рис. 4: Итоги теста DivDoubleTest

3.1.6 Результаты лабораторной работы: JUnit

В результате анализа юнит-тестов для библиотеки `calculator.jar` можно сделать следующие выводы:

1. Тестирование функции синуса (`TestSinus`):
   - Тесты корректности вычисления синуса подтвердили точные значения синуса для заданных углов, включая проверку нуля и значений, близких к π.
   - Тесты на симметричность и периодичность функции успешно подтвердили, что синус отрицательных углов соответствует ожиданиям, а значение функции остаётся неизменным после добавления 2π.
2. Тестирование функции сложения (`TestSumDouble`):
   - Тесты на сложение с граничными значениями показали, что функция корректно обрабатывает экстремально большие числа без переполнения, а результаты операций с нулём соответствуют математическим правилам.
3. Тестирование функции тангенса (`TanTests`):
   - В тесте на стремление тангенса к бесконечности при углах, близких к $\pi/2$, $3\pi/2$, результаты показали несоответствие ожиданиям: тесты не прошли, что может указывать на ошибки в обработке значений, близких к точкам неопределённости. Это требует дополнительного анализа и возможной корректировки функции.
4. Тестирование функции деления (`DivDoubleTests`):

- Тесты на деление подтвердили корректность возвращаемых значений в случаях деления на ноль, что соответствует математическим стандартам, возвращая `Infinity` или `NaN`.
- Общие тесты на деление и проверка деления числа само на себя также показали правильные и ожидаемые результаты.

Отсутствие ошибок в тестах, за исключением указанных случаев с тангенсом, свидетельствует о том, что большинство функций библиотеки работают стабильно и надёжно. Ошибки в расчётах тангенса при специфических условиях подчёркивают важность дальнейшего тестирования и потенциальной оптимизации кода для обработки особых случаев.

3.2 Лабораторная работа: Selenium

В этой лабораторной работе необходимо реализовать два теста на Java с использованием Selenium WebDriver и фреймворка TestNG для проверки корректности отображения страниц сайта `https://jdi-testing.github.io/jdi-light/index.html` в браузере Safari. Тесты должны проверять, что элементы на страницах отображаются правильно и соответствуют ожиданиям, а также должны быть выполнены согласно требованиям Java Code Convention и запущены посредством TestNG suite xml.

#	Шаг	Данные	Ожидаемый результат
1	Open test site by URL	https://jdi-testing.github.io/jdi-light/index.html	Test site is opened
2	Assert Browser title	"Home Page"	Browser title equals "Home Page"
3	Perform login	username: Roman pass: Jdi1234	User is logged
4	Assert Username is loggined	"ROMAN IOVLEV"	Name is displayed and equals to expected result
5	Assert that there are 4 items on the header section are displayed and they have proper texts	"HOME", "CONTACT FORM", "SERVICE", "METALS & COLORS"	Menu buttons are displayed and have proper texts
6	Assert that there are 4 images on the Index Page and they are displayed	4 images	Images are displayed
7	Assert that there are 4 texts on the Index Page under icons and they have proper text	4 texts below of each image	Texts are displayed and equal to expected
8	Assert that there is the iframe with “Frame Button” exist		The iframe exists
9	Switch to the iframe and check that there is “Frame Button” in the iframe		The “Frame Button” exists
10	Switch to original window back		Driver has focus on the original window
11	Assert that there are 5 items in the Left Section are displayed and they have proper text	“Home”, “Contact form”, “Service”, “Metals & Colors”, “Elements packs”	Left section menu items are displayed and have proper text
12	Close Browser		Browser is closed

Рис. 5: Задача №1

#	Шаг	Данные	Ожидаемый результат
1	Open test site by URL	https://jdi-testing.github.io/jdi-light/index.html	Test site is opened
2	Assert Browser title	"Home Page"	Browser title equals "Home Page"
3	Perform login	username: Roman pass: Jdi1234	User is logged
4	Assert User name in the left-top side of screen that user is loggined	ROMAN IOVLEV	Name is displayed and equals to expected result
5	Open through the header menu Service -> Different Elements Page		Page is opened
6	Select checkboxes	Water, Wind	Elements are checked
7	Select radio	Selen	Element is checked
8	Select in dropdown	Yellow	Element is selected
9	Assert that • for each checkbox there is an individual log row and value is corresponded to the status of checkbox • for radio button there is a log row and value is corresponded to the status of radio button • for dropdown there is a log row and value is corresponded to the selected value.		Log rows are displayed and • checkbox name and its status are corresponding to selected • radio button name and it status is corresponding to selected • dropdown name and selected value is corresponding to selected
10	Close Browser		Browser is closed

Рис. 6: Задача №2

3.2.1 Класс SettingUpDriver

Класс `SettingUpDriver` в Java настраивает `WebDriver` для браузера Safari перед выполнением тестов и обеспечивает корректное завершение работы браузера после выполнения тестов. В методе `setup` открывается тестовый сайт и выполняется вход в систему, а метод `exit` закрывает браузер после завершения тестов.

Листинг 6: SettingUpDriver

```
public class SettingUpDriver {
        protected static WebDriver driver;

        @BeforeTest
        public static void setup() {
                System.setProperty("webdriver.chrome.driver",
                        "src\\test\\resources\\chromedriver.exe");
                System.setProperty("webdriver.http.factory", "jdk-http-
                    client");

                driver = new ChromeDriver();
                driver.manage().window().maximize();

                // 1. Open test site by URL
                driver.navigate().to(
                  "https://jdi-testing.github.io/jdi-light/index.html");

                // 2. Perform login
                driver.findElement(By.cssSelector(
                        "html>body>header>div>nav>ul.uui-navigation.navbar-
                           nav.navbar-right>li>a>span"))
                                        .click();
                driver.findElement(By.id("name")).sendKeys("Roman");
                driver.findElement(By.id("password")).sendKeys("Jdi1234");
                driver.findElement(By.id("login-button")).click();
        }

        @AfterTest
        public static void exit() {
                // 3. Close the browser
                driver.close();
        }
}
```

3.2.2 Класс Test1

Класс `Test1` расширяет `SettingUpDriver` и проверяет корректность отображения различных элементов на главной странице сайта. Он открывает тестовый сайт, выполняет логин, и с помощью утверждений (`SoftAssert`) проверяет заголовок страницы, наличие и текст пунктов навигации, отображение и текст иконок, присутствие `iframe` и элементов в боковом меню.

Листинг 7: Test1

```
public class Test1 extends SettingUpDriver {

    @Test
    public void Test1() {
        SoftAssert softAssert = new SoftAssert();

        // 1. Check the page title
        softAssert.assertEquals(driver.getTitle(), "Home Page");

        // 2. Check the logged-in user name
        WebElement loggedInUser = driver.findElement(By.id("user-name"));
        softAssert.assertEquals(loggedInUser.getText(), "ROMAN IOVLEV");

        // 3. Check navigation items in the header
        WebElement headerNav = driver.findElement(By.cssSelector(
```

```
            "ul.uui-navigation.navbar-nav.m-18"));
        List<WebElement> navItems = headerNav.findElements(
            By.xpath("./child::*"));

        // 4. Verify count and visibility of navigation items
        int expectedNavItemsCount = 4;
        softAssert.assertEquals(navItems.size(), expectedNavItemsCount);
        for (WebElement item : navItems) {
            softAssert.assertTrue(item.isDisplayed());
        }

        // 5. Verify texts of navigation items
        List<String> expectedNavTexts = List.of(
            "HOME", "CONTACT FORM", "SERVICE", "METALS & COLORS");
        softAssert.assertEquals(
            navItems.stream().map(WebElement::getText).toList(),
            expectedNavTexts);

        // 6. Check display of icons and their texts
        List<WebElement> indexPageImages =
            driver.findElements(By.className("benefit-icon"));
        softAssert.assertEquals(indexPageImages.size(), 4);
        indexPageImages.forEach(img ->
            softAssert.assertTrue(img.isDisplayed()));

        List<WebElement> indexPageTexts =
            driver.findElements(By.className("benefit-txt"));
        softAssert.assertEquals(indexPageTexts.size(), 4);
        indexPageTexts.forEach(txt ->
            softAssert.assertTrue(txt.isDisplayed()));

        List<String> expectedTextContents = List.of(
            "One", "Two", "Three", "Four");
        softAssert.assertEquals(
            indexPageTexts.stream().map(WebElement::getText).toList(),
            expectedTextContents);

        // 7. Check presence of iframe and its content
        WebElement iframeElement =
            driver.findElement(By.tagName("iframe"));
        softAssert.assertEquals(
            iframeElement.getAttribute("src"),
            "https://jdi-testing.github.io/jdi-light/frame-button.html");

        // Switch to iframe and verify the button
        driver.switchTo().frame("iframe");
        WebElement frameButton =
            driver.findElement(By.id("frame-button"));
        softAssert.assertEquals(
            frameButton.getAttribute("value"),
            "Frame Button");

        // Return to the main content
        driver.switchTo().defaultContent();

        // 8. Check sidebar menu items
        WebElement sidebarMenu = driver.findElement(By.cssSelector(
            "ul.sidebar-menu.left"));
        List<WebElement> sidebarItems =
            sidebarMenu.findElements(By.xpath("./child::*"));
        sidebarItems.forEach(item ->
```

```
77            softAssert.assertTrue(item.isDisplayed()));
78
79          List<String> expectedSidebarTexts = List.of(
80            "Home", "Contact form", "Service",
81            "Metals & Colors", "Elements packs");
82          softAssert.assertEquals(
83            sidebarItems.stream().map(WebElement::getText).toList(),
84            expectedSidebarTexts);
85
86          // Assert all verifications
87          softAssert.assertAll();
88      }
89  }
```

3.2.3 Класс Test2

Тестовый класс `Test2` с использованием Selenium и TestNG проверяет корректность отображения и работы элементов на сайте. Тесты проверяют заголовок страницы, процесс входа в систему, выбор чекбоксов, радиокнопок и значений в выпадающем списке, а также корректность записей в логах действий.

Листинг 8: Test2

```
1   public class Test2 extends SettingUpDriver {
2
3           @Test
4           public void testTitle() {
5                   // 1. Check page title
6                   driver.get("https://jdi-testing.github.io/jdi-light/index.
                        html");
7                   assertEquals(driver.getTitle(), "Home Page");
8           }
9
10          @Test
11          public void testLogin() {
12                  // 2. Perform user login
13                  driver.findElement(By.id("user-icon")).click();
14                  driver.findElement(By.id("name")).sendKeys("Roman");
15                  driver.findElement(By.id("password")).sendKeys("Jdi1234");
16                  driver.findElement(By.id("login-button")).click();
17                  assertEquals(
18                          driver.findElement(By.id("user-name")).getText(),
19                          "ROMAN IOVLEV");
20          }
21
22          @Test
23          public void testElement() {
24                  // 3. Navigate to the page with checkboxes, radio buttons,
                        and dropdowns
25                  driver.findElement(By.cssSelector(
26                          "body>header>div>nav>ul.uui-navigation.navbar-nav.m
                                -18>li>a>span"))
27                          .click();
28                  driver.findElement(By.xpath(
29                          "/html/body/header/div/nav/ul[1]/li[3]/a")).click();
30
31                  // 4. Select checkboxes "Water" and "Wind"
32                  List<WebElement> checkboxes =
33                          driver.findElements(By.className("label-checkbox"));
34                  for (WebElement checkbox : checkboxes) {
```

```
35                          if (checkbox.getText().equals("Water")
36                           || checkbox.getText().equals("Wind")) {
37                                  checkbox.click();
38                          }
39                  }
40
41                  // 5. Select radio button "Selen"
42                  List<WebElement> radios =
43                           driver.findElements(By.className("label-radio"));
44                  for (WebElement radio : radios) {
45                          if (radio.getText().equals("Selen")) {
46                                  radio.click();
47                          }
48                  }
49
50                  // 6. Select option "Yellow" from the dropdown
51                  List<WebElement> options =
52                           driver.findElements(By.tagName("option"));
53                  for (WebElement option : options) {
54                          if (option.getText().equals("Yellow")) {
55                                  option.click();
56                          }
57                  }
58
59                  // 7. Verify log entries
60                  final int logIndexStart = 9;
61                  String logsText = driver.findElement(By.cssSelector(
62                           "ul.panel-body-list.logs")).getText();
63                  String[] logLinesArray = logsText.split("\n");
64                  List<String> logLines = Arrays.stream(logLinesArray)
65                    .map(log -> log.substring(logIndexStart)).toList();
66
67                  // Expected log entries
68                  List<String> expectedLogEntries = List.of(
69                          "Colors: value changed to Yellow",
70                    "Metal: value changed to Selen",
71                    "Wind: state changed to true",
72                    "Water: state changed to true"
73                  );
74
75                  // Check logs
76                  assertEquals(logLines, expectedLogEntries);
77          }
78  }
```

3.2.4 Результаты лабораторной работы №2

В лабораторной работе №2 успешное прохождение всех тестов с использованием Selenium WebDriver и TestNG для проверки сайта гарантировалось несколькими ключевыми аспектами. Вопервых, настройка WebDriver в классе `SettingUpDriver` была выполнена корректно, обеспечивая стабильное взаимодействие с Safari. Использование точных CSS и XPath селекторов позволило эффективно взаимодействовать с элементами на странице. Мягкие утверждения (Soft Asserts) в `Test1` давали возможность выполнения нескольких проверок в рамках одного теста, улучшая детализацию результатов без прерывания выполнения при возникновении ошибок. Тесты охватывали широкий спектр элементов страницы, включая заголовки, элементы навигации и `iframe`, что увеличивало покрытие функционала. Управление сессией браузера, гарантирующее закрытие браузера после тестирования, предотвратило влияние предыдущих тестов на последующие результаты.

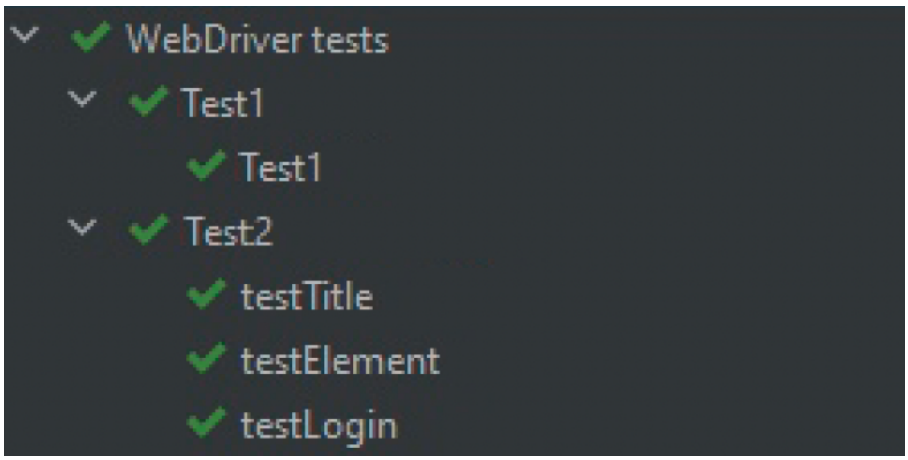

Рис. 7: Результаты тестов WebDriver

Так же так как проект был собран с помощью Gradle, можно в папке build.reports, открыть файл index.html, в котором будет графический результат:

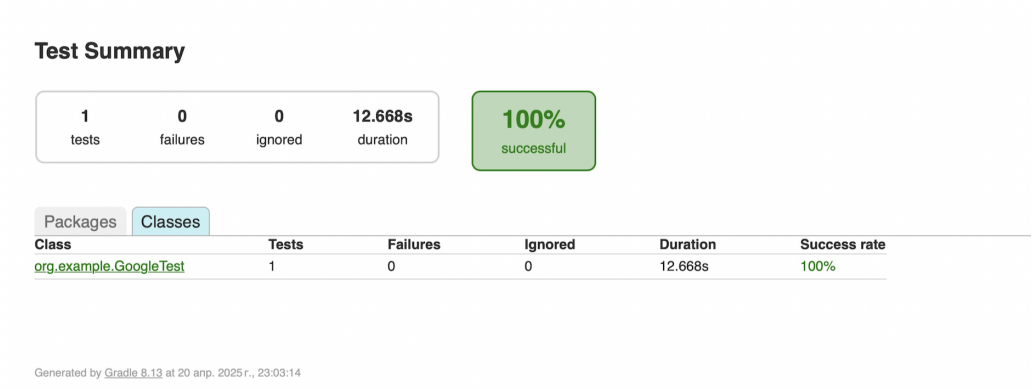

Рис. 8: Графический результат

Заключение

В ходе выполнения лабораторных работ были изучены некоторые аспекты автоматизации тестирования программного обеспечения. Основное внимание было уделено созданию и написанию юнит-тестов с использованием JUnit, организации и выполнению тестов с помощью Selenium WebDriver и TestNG. Эти навыки и инструменты повышают эффективность и качество процесса тестирования программного обеспечения.

Список источников

1. Курс лекций по атоматизированному тестированию. Спасов Г.Е.