**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра ВТ**

отчет

**по лабораторной работе №8**

**по дисциплине «Объектно-ориентированное программирование»**

Тема: **Модульное тестирование приложения**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 3315 |  | Красавцев Н.В. |
| Преподаватель |  | Гречухин М.Н. |

Санкт-Петербург

2024

**Цель работы.**

Знакомство с технологией модульного тестирования в Java с использованием библиотеки JUnit.

**Описание задания.**

Ознакомиться с технологией модульного тестирования Java-приложений с использованием системы JUnit. Написать JUnit-тесты для выбранных методов из предыдущих лабораторных работ.

**Код методов выбранных для теста.**

void checkName(String input) throws MyException, NullPointerException{

if (input.length() == 0) throw new NullPointerException("Вы не ввели имя");

if (!input.matches("[а-яА-ЯёЁ ]+")) throw new MyException();

}

void checkSurname2(String input) throws MyException, NullPointerException{

if (input.length() == 0) throw new NullPointerException("Вы не ввели отчество");

if (!input.matches("[а-яА-ЯёЁ ]+")) throw new MyException();

}

**Код тестов.**

@Test

public void testCheckName\_validName() {

ServiceList serviceList = new ServiceList();

try {

serviceList.checkName("Иван");

} catch (Exception e) {

fail("Exception should not be thrown for a valid name.");

}

}

@Test

public void testCheckName\_invalidName\_empty() {

ServiceList serviceList = new ServiceList();

Exception exception = assertThrows(NullPointerException.class, () -> {

serviceList.checkName("");

});

assertEquals("Вы не ввели имя", exception.getMessage());

}

@Test

public void testCheckName\_invalidName\_nonCyrillic() {

ServiceList serviceList = new ServiceList();

Exception exception = assertThrows(MyException.class, () -> {

serviceList.checkName("John");

});

assertNotNull(exception);

}

// Test checkSurname2() method

@Test

public void testCheckSurname2\_validPatronymic() {

ServiceList serviceList = new ServiceList();

try {

serviceList.checkSurname2("Александрович");

} catch (Exception e) {

fail("Exception should not be thrown for a valid patronymic.");

}

}

@Test

public void testCheckSurname2\_invalidPatronymic\_empty() {

ServiceList serviceList = new ServiceList();

Exception exception = assertThrows(NullPointerException.class, () -> {

serviceList.checkSurname2("");

});

assertEquals("Вы не ввели отчество", exception.getMessage());

}

@Test

public void testCheckSurname2\_invalidPatronymic\_nonCyrillic() {

ServiceList serviceList = new ServiceList();

Exception exception = assertThrows(MyException.class, () -> {

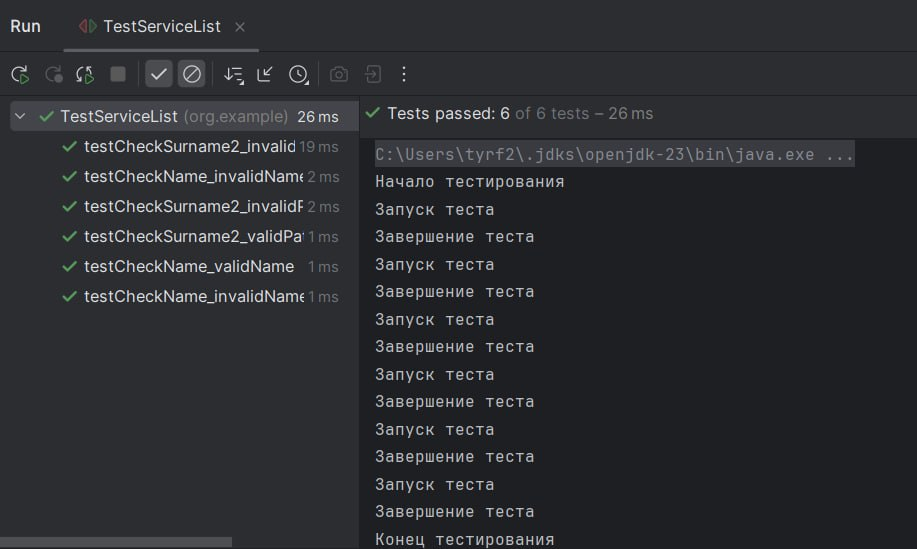
serviceList.checkSurname2("Johnson");

});

assertNotNull(exception);

}

**Работа класса.**



Javadoc.

@Test

/\*\*

\*

\* It is a constructor.

\*

\*/

public void testCheckName\_validName() {

ServiceList serviceList = new ServiceList();

try {

serviceList.checkName("Иван");

} catch (Exception e) {

fail("Exception should not be thrown for a valid name.");

}

}

@Test

/\*\*

\*

\* It is a constructor.

\*

\*/

public void testCheckName\_invalidName\_empty() {

ServiceList serviceList = new ServiceList();

Exception exception = assertThrows(NullPointerException.class, () -> {

serviceList.checkName("");

});

assertEquals("Вы не ввели имя", exception.getMessage());

}

@Test

/\*\*

\*

\* It is a constructor.

\*

\*/

public void testCheckName\_invalidName\_nonCyrillic() {

ServiceList serviceList = new ServiceList();

Exception exception = assertThrows(MyException.class, () -> {

serviceList.checkName("John");

});

assertNotNull(exception);

}

// Test checkSurname2() method

@Test

/\*\*

\*

\* It is a constructor.

\*

\*/

public void testCheckSurname2\_validPatronymic() {

ServiceList serviceList = new ServiceList();

try {

serviceList.checkSurname2("Александрович");

} catch (Exception e) {

fail("Exception should not be thrown for a valid patronymic.");

}

}

@Test

/\*\*

\*

\* It is a constructor.

\*

\*/

public void testCheckSurname2\_invalidPatronymic\_empty() {

ServiceList serviceList = new ServiceList();

Exception exception = assertThrows(NullPointerException.class, () -> {

serviceList.checkSurname2("");

});

assertEquals("Вы не ввели отчество", exception.getMessage());

}

@Test

/\*\*

\*

\* It is a constructor.

\*

\*/

public void testCheckSurname2\_invalidPatronymic\_nonCyrillic() {

ServiceList serviceList = new ServiceList();

Exception exception = assertThrows(MyException.class, () -> {

serviceList.checkSurname2("Johnson");

});

assertNotNull(exception);

} \*

\*/

public static void allTestsFinished() {

System.out.println("Конец тестирования");

}

@Before // Фиксируем запуск теста

/\*\*

\*

\* Test started

\*

\*/

public void testStarted() {

System.out.println("Запуск теста");

}

@After // Фиксируем завершение теста

/\*\*

\*

\* Test finished

\*

\*/

public void testFinished() {

System.out.println("Завершение теста");

}

}

**Выводы.**

В процессе задания мы изучили, как использовать систему JUnit для тестирования Java-приложений по модулям. Мы освоили различные методы и правила написания тестов в этой системе, а также создали проверки для определенных функций в нашем проекте курсовой работы.