**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**
**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**
**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**
**“ЛЭТИ” ИМ.В.И.УЛЬЯНОВА (ЛЕНИНА)»**
**КАФЕДРА МОЭВМ**

# ОТЧЕТ

**по лабораторно-практической работе № 9**
**«Модульное тестирование приложения»**
**по дисциплине «Объектно - ориентированное программирование на языке Java»**

Выполнил: Барченков П. А.

Факультет: КТИ

Группа: №3312

Подпись преподавателя: _____

Санкт-Петербург

2024

**Содержание**

## Цель работы

Знакомство с технологией модульного тестирования Java приложений с использованием системы JUnit.

## Перечень методов, которые тестируются в приложении

1. Метод добавления учителя addTeacher()

Метод добавляет нового учителя в таблицу, если все поля (ФИО учителя, Предмет, Классы) заполнены корректно. Если хотя бы одно поле пустое или содержит некорректные данные, запись не добавляется, и выводится сообщение об ошибке.

2. Метод редактирования учителя editTeacher()

Метод редактирует данные учителя в таблице по указанной строке. При вызове метода проверяется, что все поля заполнены корректно. Если данные валидны, соответствующая запись обновляется. В случае некорректного ввода выводится сообщение об ошибке.

3. Метод удаления учителя deleteTeacher()

Метод удаляет учителя из таблицы по индексу выбранной строки. Если индекс корректен и строка выбрана, запись удаляется. В противном случае выводится сообщение об ошибке.

4. Метод добавления ученика addStudent()

Метод добавляет нового ученика в таблицу, если все поля (ФИО ученика, Класс, Успеваемость) заполнены корректно. Если хотя бы одно поле пустое или содержит некорректные данные, запись не добавляется, и выводится сообщение об ошибке.

5. Метод редактирования ученика editStudent()

Метод редактирует данные ученика в таблице по указанной строке. При вызове метода проверяется, что все поля заполнены корректно. Если данные валидны, соответствующая запись обновляется. В случае некорректного ввода выводится сообщение об ошибке.

6. Метод удаления ученика deleteStudent()

Метод удаляет ученика из таблицы по индексу выбранной строки. Если индекс корректен и строка выбрана, запись удаляется. В противном случае выводится сообщение об ошибке.

7. Метод загрузки данных loadDataFromFile()

Метод загружает данные учителей и учеников из выбранного текстового файла, обновляя соответствующие таблицы в приложении. Если при загрузке данных возникает ошибка (например, некорректный формат файла или проблемы с доступом к файлу), выводится сообщение об ошибке, и данные не обновляются.

8. Метод сохранения данных saveDataToFile()

Метод сохраняет текущие данные учителей и учеников из таблиц в выбранный текстовый файл. Если при сохранении данных возникает ошибка (например, проблемы с доступом к файлу или некорректные данные), выводится сообщение об ошибке, и данные не сохраняются.

9. Метод поиска данных searchTable()

Метод фильтрует данные в текущей таблице (учителя или ученики) на основе выбранного критерия поиска и введенного значения. Если поле поиска пустое, выводится сообщение об ошибке. В противном случае таблица отображает только те записи, которые соответствуют заданным условиям поиска.

10. Метод сброса фильтрации resetTable()

Метод сбрасывает все активные фильтры и восстанавливает отображение всех записей в таблицах учителей и учеников. Также очищает поле поиска, возвращая интерфейс в исходное состояние.

## Скриншоты, иллюстрирующие выполнение тестов

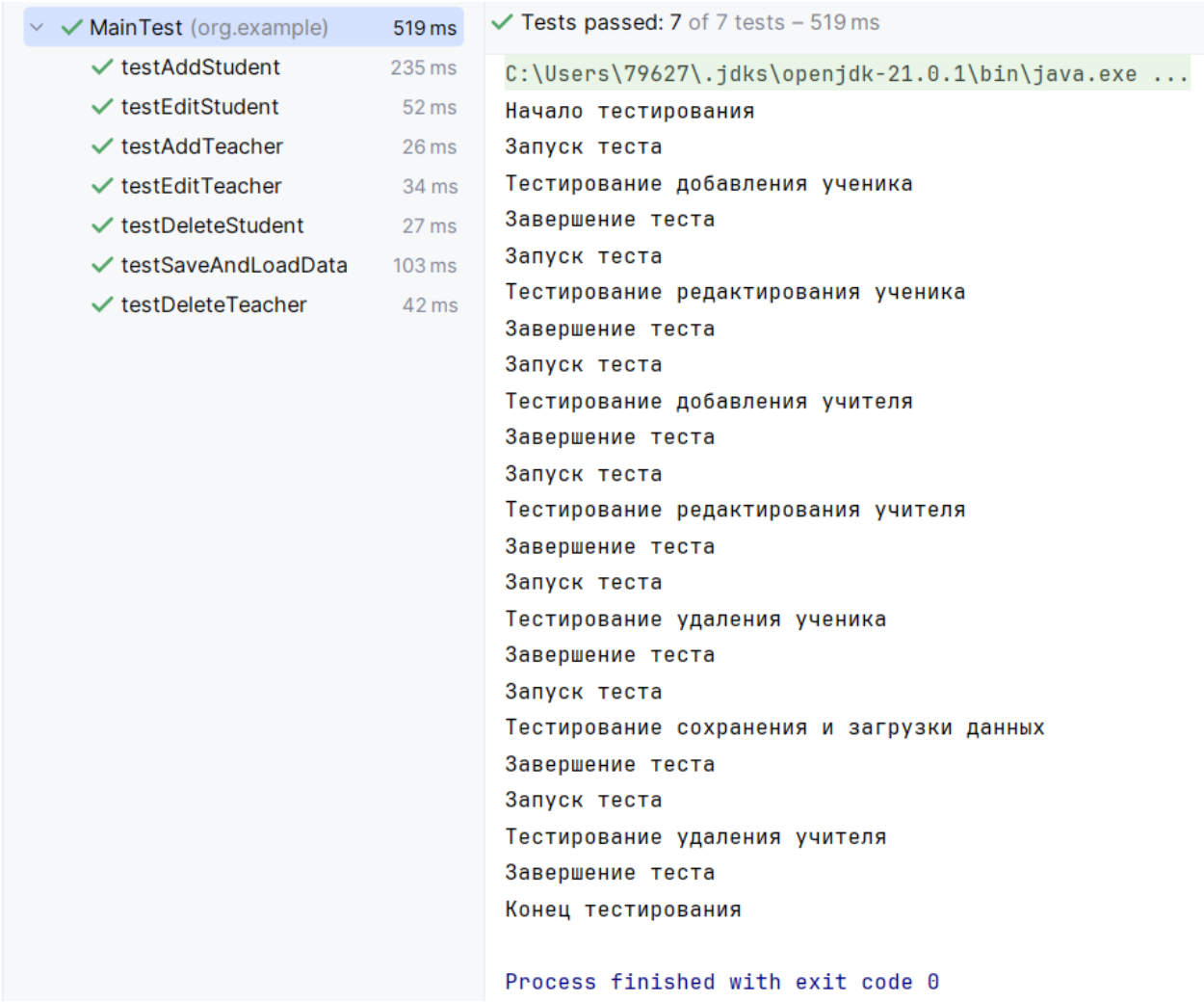


Рисунок 1 – Результат тестирования.

## Исходные тексты классов тестов

```
package org.example;

import org.example.Main;
import org.example.*;
import org.junit.*;
import javax.swing.table.DefaultTableModel;
import java.io.File;

public class MainTest {

    private Main mainInstance;

    @BeforeClass
    public static void allTestsStarted() {
        System.out.println("Начало тестирования");
    }

    @AfterClass
    public static void allTestsFinished() {
        System.out.println("Конец тестирования");
    }

    @Before
    public void setUp() {
        System.out.println("Запуск теста");
```

```
        mainInstance = new Main();
        mainInstance.SchoolManagementSystem(); // Инициализируем GUI и данные

        // Инициализируем таблицы с тестовыми данными
        // Учителя
        String[] teacherColumns = {"ФИО учителя", "Предмет", "Классы"};
        DefaultTableModel teacherModel = new
DefaultTableModel(teacherColumns, 0);
        teacherModel.addRow(new Object[]{"Иванов Иван Иванович",
"Математика", "5А;6Б"});
        teacherModel.addRow(new Object[]{"Петрова Анна Сергеевна", "Русский
язык", "7В;8Г"});
        teacherModel.addRow(new Object[]{"Сидоров Петр Петрович", "История",
"9А;10Б"});
        mainInstance.teacherTableModel = teacherModel;
        mainInstance.teacherTable.setModel(teacherModel);
        mainInstance.teacherSorter = null; // Отключаем сортировщик
        mainInstance.teacherTable.setRowSorter(null);

        // Ученики
        String[] studentColumns = {"ФИО ученика", "Класс", "Успеваемость"};
        DefaultTableModel studentModel = new
DefaultTableModel(studentColumns, 0);
        studentModel.addRow(new Object[]{"Смирнов Алексей Иванович", "5А",
"Отлично"});
        studentModel.addRow(new Object[]{"Кузнецова Мария Петровна", "6Б",
"Хорошо"});
        studentModel.addRow(new Object[]{"Новиков Дмитрий Сергеевич", "7В",
"Удовлетворительно"});
        mainInstance.studentTableModel = studentModel;
        mainInstance.studentTable.setModel(studentModel);
        mainInstance.studentSorter = null; // Отключаем сортировщик
        mainInstance.studentTable.setRowSorter(null);
    }

    @After
    public void finishTest() {
        System.out.println("Завершение теста");
    }

    @Test
    public void testAddTeacher() {
        System.out.println("Тестирование добавления учителя");

        String[] newTeacher = {"Смирнов Алексей Викторович", "Физика",
"11А;12Б"};
        try {
            mainInstance.addTeacher(newTeacher[0], newTeacher[1],
newTeacher[2]);
        } catch (InvalidInputException e) {
            Assert.fail("Произошла ошибка при добавлении учителя: " +
e.getMessage());
        }

        Assert.assertEquals("Должно быть 4 учителя после добавления", 4,
mainInstance.teacherTableModel.getRowCount());
        Assert.assertEquals("ФИО нового учителя должно совпадать",
newTeacher[0], mainInstance.teacherTableModel.getValueAt(3, 0));
        Assert.assertEquals("Предмет нового учителя должен совпадать",
newTeacher[1], mainInstance.teacherTableModel.getValueAt(3, 1));
        Assert.assertEquals("Классы нового учителя должны совпадать",
newTeacher[2], mainInstance.teacherTableModel.getValueAt(3, 2));
    }

    @Test
    public void testEditTeacher() {
        System.out.println("Тестирование редактирования учителя");

        // Редактируем первого учителя
        String[] editedTeacher = {"Иванов Иван Иванович", "Физика", "5А;6Б"};
        try {
            mainInstance.editTeacher(0, editedTeacher[0], editedTeacher[1],
editedTeacher[2]);
        } catch (InvalidInputException e) {
            Assert.fail("Произошла ошибка при редактировании учителя: " +
e.getMessage());
        }
```

```
        Assert.assertEquals("ФИО учителя должно быть отредактировано",
editedTeacher[0], mainInstance.teacherTableModel.getValueAt(0, 0));
        Assert.assertEquals("Предмет учителя должен быть отредактирован",
editedTeacher[1], mainInstance.teacherTableModel.getValueAt(0, 1));
        Assert.assertEquals("Классы учителя должны быть отредактированы",
editedTeacher[2], mainInstance.teacherTableModel.getValueAt(0, 2));
    }

    @Test
    public void testDeleteTeacher() {
        System.out.println("Тестирование удаления учителя");

        // Удаляем второго учителя (индекс 1)
        try {
            mainInstance.deleteTeacher(1);
        } catch (InvalidInputException e) {
            Assert.fail("Произошла ошибка при удалении учителя: " +
e.getMessage());
        }

        Assert.assertEquals("Должно быть 2 учителя после удаления", 2,
mainInstance.teacherTableModel.getRowCount());
        Assert.assertEquals("Второй учитель должен быть Сидоров Петр
Петрович", "Сидоров Петр Петрович",
mainInstance.teacherTableModel.getValueAt(1, 0));
    }

    @Test
    public void testAddStudent() {
        System.out.println("Тестирование добавления ученика");

        String[] newStudent = {"Иванова Елена Сергеевна", "8Г", "Отлично"};
        try {
            mainInstance.addStudent(newStudent[0], newStudent[1],
newStudent[2]);
        } catch (InvalidInputException e) {
            Assert.fail("Произошла ошибка при добавлении ученика: " +
e.getMessage());
        }

        Assert.assertEquals("Должно быть 4 ученика после добавления", 4,
mainInstance.studentTableModel.getRowCount());
        Assert.assertEquals("ФИО нового ученика должно совпадать",
newStudent[0], mainInstance.studentTableModel.getValueAt(3, 0));
        Assert.assertEquals("Класс нового ученика должен совпадать",
newStudent[1], mainInstance.studentTableModel.getValueAt(3, 1));
        Assert.assertEquals("Успеваемость нового ученика должна совпадать",
newStudent[2], mainInstance.studentTableModel.getValueAt(3, 2));
    }

    @Test
    public void testEditStudent() {
        System.out.println("Тестирование редактирования ученика");

        // Редактируем первого ученика
        String[] editedStudent = {"Смирнов Алексей Иванович", "5А",
"Хорошо"};
        try {
            mainInstance.editStudent(0, editedStudent[0], editedStudent[1],
editedStudent[2]);
        } catch (InvalidInputException e) {
            Assert.fail("Произошла ошибка при редактировании ученика: " +
e.getMessage());
        }

        Assert.assertEquals("ФИО ученика должно быть отредактировано",
editedStudent[0], mainInstance.studentTableModel.getValueAt(0, 0));
        Assert.assertEquals("Класс ученика должен быть отредактирован",
editedStudent[1], mainInstance.studentTableModel.getValueAt(0, 1));
        Assert.assertEquals("Успеваемость ученика должна быть
отредактирована", editedStudent[2],
mainInstance.studentTableModel.getValueAt(0, 2));
    }

    @Test
    public void testDeleteStudent() {
        System.out.println("Тестирование удаления ученика");
```

```
        // Удаляем третьего ученика (индекс 2)
        try {
            mainInstance.deleteStudent(2);
        } catch (InvalidInputException e) {
            Assert.fail("Произошла ошибка при удалении ученика: " +
e.getMessage());
        }

        Assert.assertEquals("Должно быть 2 ученика после удаления", 2,
mainInstance.studentTableModel.getRowCount());
        Assert.assertEquals("Второй ученик должен быть Кузнецова Мария
Петровна", "Кузнецова Мария Петровна",
mainInstance.studentTableModel.getValueAt(1, 0));
    }

    @Test
    public void testSaveAndLoadData() {
        System.out.println("Тестирование сохранения и загрузки данных");

        try {
            // Создаем временный файл
            File tempFile = File.createTempFile("testData", ".txt");
            tempFile.deleteOnExit();

            // Сохраняем текущие данные
            mainInstance.saveDataToFile(tempFile);

            // Изменяем данные
            mainInstance.addTeacher("Смирнов Алексей Викторович", "Физика",
"11А;12Б");
            mainInstance.addStudent("Иванова Елена Сергеевна", "8Г",
"Отлично");

            Assert.assertEquals("Должно быть 4 учителя перед загрузкой", 4,
mainInstance.teacherTableModel.getRowCount());
            Assert.assertEquals("Должно быть 4 ученика перед загрузкой", 4,
mainInstance.studentTableModel.getRowCount());

            // Загружаем данные из файла
            mainInstance.loadDataFromFile(tempFile);

            // Проверяем, что данные вернулись к исходному состоянию
            Assert.assertEquals("Должно быть 3 учителя после загрузки", 3,
mainInstance.teacherTableModel.getRowCount());
            Assert.assertEquals("Должен быть 3 ученик после загрузки", 3,
mainInstance.studentTableModel.getRowCount());

            // Проверка первого учителя
            Assert.assertEquals("ФИО первого учителя должно совпадать",
"Иванов Иван Иванович", mainInstance.teacherTableModel.getValueAt(0, 0));
            Assert.assertEquals("Предмет первого учителя должен совпадать",
"Математика", mainInstance.teacherTableModel.getValueAt(0, 1));
            Assert.assertEquals("Классы первого учителя должны совпадать",
"5А;6Б", mainInstance.teacherTableModel.getValueAt(0, 2));

            // Проверка первого ученика
            Assert.assertEquals("ФИО первого ученика должно совпадать",
"Смирнов Алексей Иванович", mainInstance.studentTableModel.getValueAt(0, 0));
            Assert.assertEquals("Класс первого ученика должен совпадать",
"5А", mainInstance.studentTableModel.getValueAt(0, 1));
            Assert.assertEquals("Успеваемость первого ученика должна
совпадать", "Отлично", mainInstance.studentTableModel.getValueAt(0, 2));

        } catch (Exception e) {
            Assert.fail("Произошла ошибка при тестировании сохранения и
загрузки данных: " + e.getMessage());
        }
    }
}
```

## Приложение

Репозиторий: https://github.com/PlatonBarchenkov/OOP_lab_09.git