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: Локтионов Т. И.

Факультет КТИ

Группа № 3311

Подпись преподавателя \_\_\_\_\_\_\_\_\_\_

Санкт-Петербург

2024 г

**Цель работы:**знакомство с технологией модульного тестирования Java приложений с использованием системы JUnit.

**Описание задания:**1. Создайте новый проект, который будет дублировать проект лабораторной работы № 3.

2. Проанализируйте классы приложения и определите, какие методы необходимо протестировать.

3. Напишите JUnit-тесты для выбранных методов.

4. Запустите тесты и снимите с экрана скриншоты, иллюстрирующие выполнение тестов.

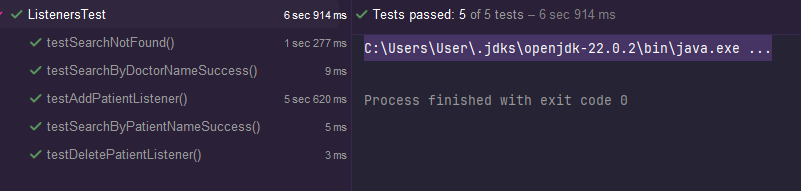
**Перечень методов:**

>>public static ActionListener**getAddPatientListener** ( ) {}

>> public static ActionListener**getDeletePatientListener** ( ) {}

>> public static ActionListener**getSearchListener** ( ) {}

**Скриншоты успешно пройденных тестов:**

**  
//** *последний слушатель был разбит на 3 отдельных теста*

**Ссылки:**

>>репозиторий: <https://github.com/iconLti/LTprojects/tree/master/OOP/Java%20projects/Hospital-lab09_mav>

>> google disk: <https://drive.google.com/drive/folders/1YRKOxpkXtTLnzTo8e_P50Sp0jCBNFT9a?usp=sharing>

**Исходный текст тестов:**

*import* org.junit.jupiter.api.BeforeEach;  
*import* org.junit.jupiter.api.Test;  
  
*import* javax.swing.\*;  
*import* javax.swing.table.DefaultTableModel;  
  
*import static* org.junit.jupiter.api.Assertions.\*;  
  
*public class* ListenersTest {  
  
 *private* DefaultTableModel tableModel;  
 *private* JTable dataTable;  
 *private* JTextField searchField;  
 *private* JComboBox<String> searchType;  
 *private* JFrame frame;  
  
 @BeforeEach  
 *public void* setUp() {  
 *// Инициализация таблицы перед каждым тестом* String[] columns = {"Имя пациента", "Болезнь", "Врач", "Специализация врача", "Дата приёма", "Статус"};  
 tableModel = *new* DefaultTableModel(columns, 0);  
 dataTable = *new* JTable(tableModel);  
  
 *// Добавляем тестовые данные* tableModel.addRow(*new* Object[]{"Иван Иванов", "ОРВИ", "Доктор А", "Терапевт", "01.10.2024", "Принят"});  
 tableModel.addRow(*new* Object[]{"Петр Петров", "Грипп", "Доктор Б", "Терапевт", "02.10.2024", "Ожидание"});  
  
 *// Инициализация других компонентов* searchField = *new* JTextField();  
 searchType = *new* JComboBox<>(*new* String[]{"Ключевому слову", "Имени пациента", "Имени врача"});  
 frame = *new* JFrame(); *// Фрейм для сообщений* }  
  
 @Test  
 *public void* testAddPatientListener() {  
 *// Эмуляция добавления пациента* Listeners.*getAddPatientListener*(tableModel).actionPerformed(*null*);  
 *assertEquals*(3, tableModel.getRowCount(), "Пациент должен быть добавлен в таблицу");  
 }  
  
 @Test  
 *public void* testDeletePatientListener() {  
 *// Добавляем тестовую строку* tableModel.addRow(*new* Object[]{"Иван Иванов", "ОРВИ", "Доктор А", "Терапевт", "01.10.2024", "Принят"});  
  
 *// Выбираем строку в таблице* dataTable.setRowSelectionInterval(0, 0);  
  
 *// Эмуляция удаления пациента* Listeners.*getDeletePatientListener*(tableModel, dataTable, *null*).actionPerformed(*null*);  
  
 *// Проверяем, что строка удалена  
 assertEquals*(2, tableModel.getRowCount(), "Пациент должен быть удален из таблицы");  
 }  
  
 @Test  
 *public void* testSearchByPatientNameSuccess() {  
 *// Устанавливаем критерии поиска* searchField.setText("Иван Иванов");  
 searchType.setSelectedIndex(1); *// Поиск по имени пациента  
  
 // Эмуляция поиска* Listeners.*getSearchListener*(dataTable, searchField, searchType, frame).actionPerformed(*null*);  
  
 *// Проверяем, что нужная строка выбрана  
 assertEquals*(0, dataTable.getSelectedRow(), "Пациент должен быть найден");  
 }  
  
 @Test  
 *public void* testSearchByDoctorNameSuccess() {  
 *// Устанавливаем критерии поиска* searchField.setText("Доктор Б");  
 searchType.setSelectedIndex(2); *// Поиск по имени врача  
  
 // Эмуляция поиска* Listeners.*getSearchListener*(dataTable, searchField, searchType, frame).actionPerformed(*null*);  
  
 *// Проверяем, что нужная строка выбрана  
 assertEquals*(1, dataTable.getSelectedRow(), "Пациент должен быть найден по имени врача");  
 }  
  
 @Test  
 *public void* testSearchNotFound() {  
 *// Устанавливаем критерии поиска* searchField.setText("Неизвестный пациент");  
 searchType.setSelectedIndex(1); *// Поиск по имени пациента  
  
 // Эмуляция поиска* Listeners.*getSearchListener*(dataTable, searchField, searchType, frame).actionPerformed(*null*);  
  
 *// Проверяем, что строка не выбрана, так как ничего не найдено  
 assertEquals*(-1, dataTable.getSelectedRow(), "Ни одна строка не должна быть выбрана");  
 }  
}