МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное
учреждение высшего образования

«Национальный исследовательский университет
«Московский институт электронной техники»

Лабораторная работа № 6 по дисциплине
«Конструирование программного обеспечения»

«Unit-тестирование»

Подготовили:

Студенты группы ПИН-36

Бойков И.И.
Бозюкова Л.С.
Карпухин Д.И.
Силантьев М.В.

Москва, 2024

Для проведения модульного тестирования реализуем минимально жизнеспособный продукт на Java с использованием JavaFX в среде IntelliJ IDEA:

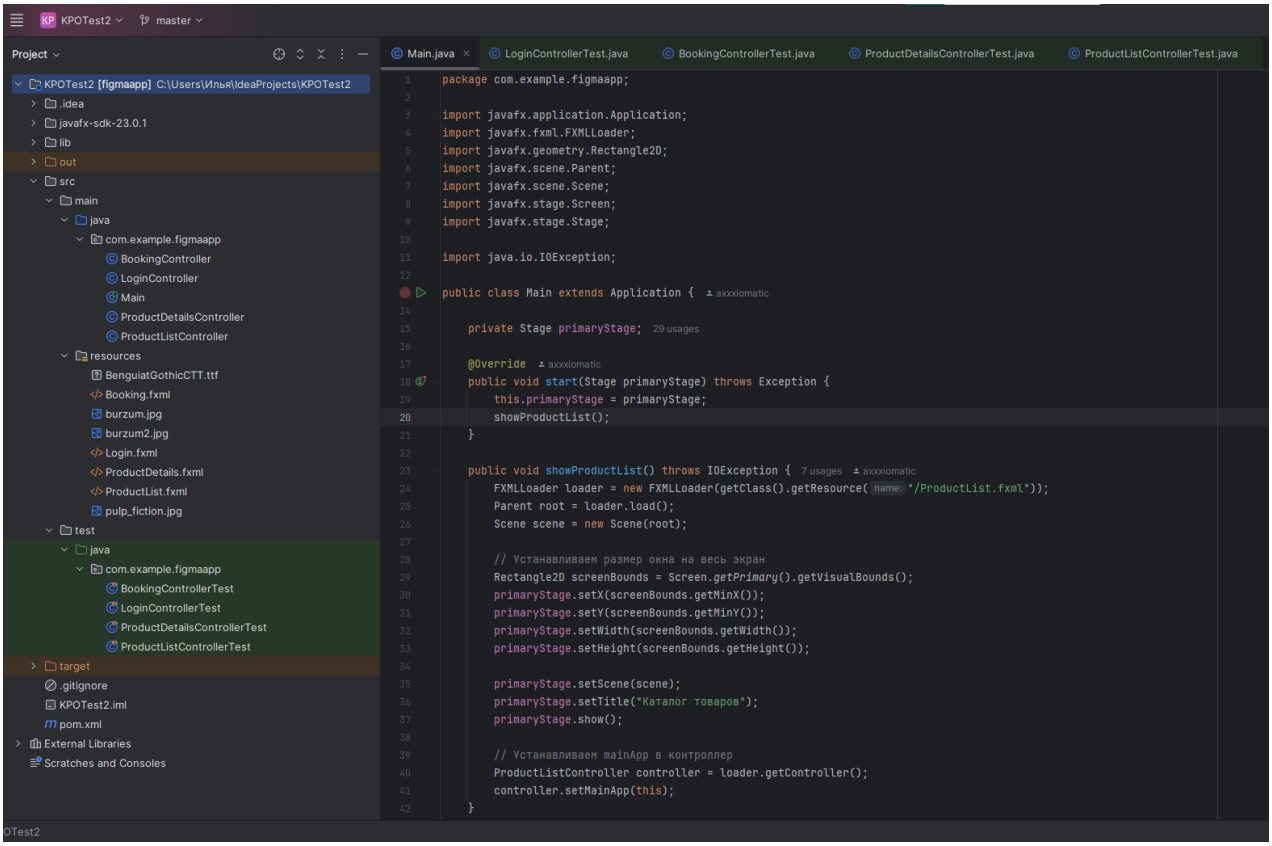

Рисунок 1. Программный код и структура проекта в IDE IntelliJ IDEA

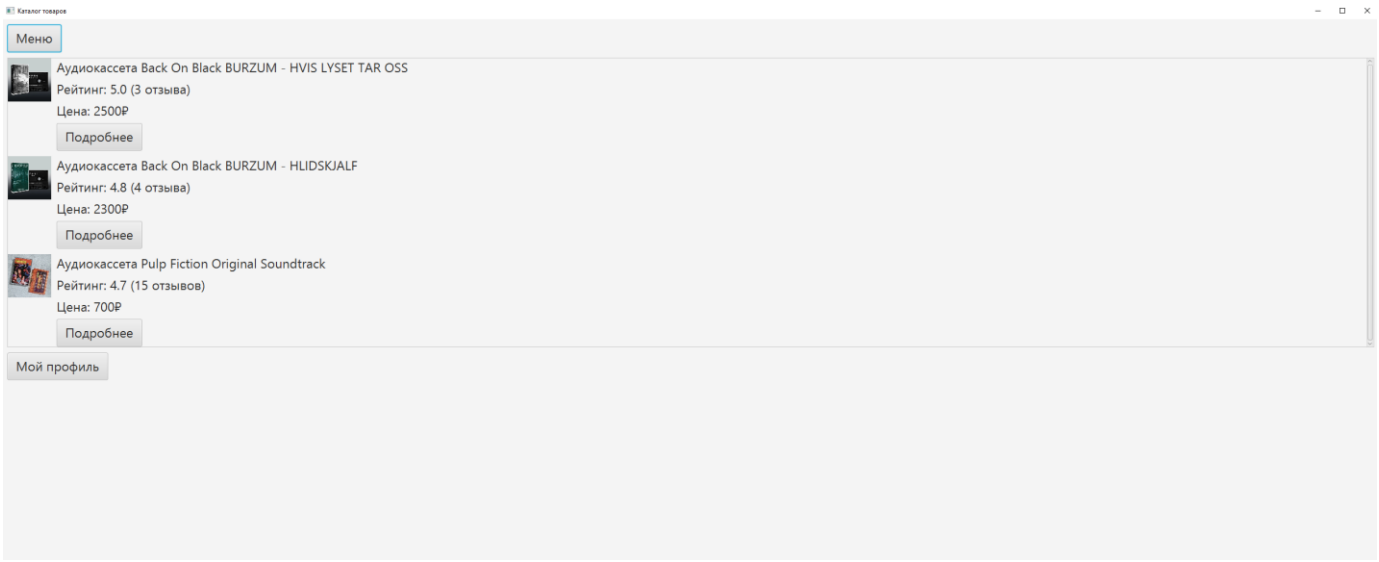

Рисунок 2. Внешний вид графического интерфейса прототипа проекта

Получен рабочий и тестируемый прототип разрабатываемого продукта. Теперь напишем и попробуем запустить несколько модульных тестов для прототипа проекта, используя JUnit:

```java
import org.junit.jupiter.api.Test;
import org.mockito.Mockito;
import org.testfx.framework.junit5.ApplicationTest;

import static org.mockito.Mockito.*;

public class BookingControllerTest extends ApplicationTest {  axxxiomatic

    private BookingController controller;  6 usages
    private Main mainApp;  3 usages

    @BeforeEach  axxxiomatic
    public void setUp() {
        controller = new BookingController();
        mainApp = Mockito.mock(Main.class);
        controller.setMainApp(mainApp);
    }

    @Test  axxxiomatic
    public void testHandleBookAction() throws Exception {
        TextField fromField = new TextField( s: "10:00");
        TextField toField = new TextField( s: "12:00");
        TextField additionalInfoField = new TextField( s: "Additional Info");

        controller.fromField = fromField;
        controller.toField = toField;
        controller.additionalInfoField = additionalInfoField;

        controller.handleBookAction(new ActionEvent());

        verify(mainApp, times( wantedNumberOfInvocations: 1)).showProductList();
    }
}
```

Рисунок 3. Структура и программный код модульного теста

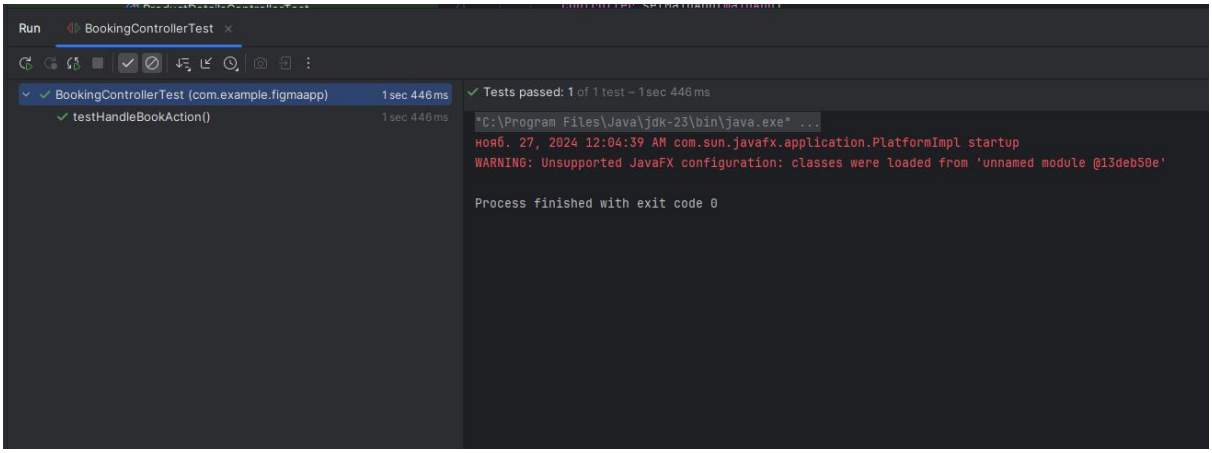

Рисунок 4. Результат выполнения модульного теста

Как видим, тест был успешно завершён, несмотря на небольшие различия в настройке конфигурации JavaFX. Аналогичным образом реализуем модульные тесты протестируем остальные классы.

Проведём анализ покрытия кода модульными тестами. Анализировать будем средствами нейронной сети Deepseek Coder, ей были предоставлены все классы и тесты в формате исходного кода и выдано задание провести анализ покрытия кода в соответствии с текстом лабораторной работы. Рассмотрим полученные результаты:

1. Main.java

Класс Main содержит методы для отображения различных экранов приложения. Тесты для этого класса не предоставлены, поэтому покрытие кода тестами для Main будет нулевым.

2. LoginController.java

Класс LoginController содержит метод handleLoginAction, который обрабатывает событие нажатия кнопки входа.

Тест: *LoginControllerTest.testHandleLoginAction*

- **Покрытие операторов:** 100% (все строки метода handleLoginAction выполнены)
- **Покрытие условий:** 50% (проверка на пустоту полей выполнена, но не проверяется случай, когда поля не пусты)
- **Покрытие путей:** 50% (проверяется только один путь – когда поля пусты)
- **Покрытие функций:** 100% (метод handleLoginAction выполнен)
- **Покрытие вход/выход:** 100% (вызов showProductList выполнен)
- **Покрытие комбинаций:** 50% (проверяется только одна комбинация – пустые поля)

3. BookingController.java

Класс BookingController содержит метод handleBookAction, который обрабатывает событие нажатия кнопки бронирования.

Тест: *BookingControllerTest.testHandleBookAction*

- **Покрытие операторов:** 100% (все строки метода handleBookAction выполнены)
- **Покрытие условий:** 50% (проверка на пустоту полей выполнена, но не проверяется случай, когда поля не пусты)
- **Покрытие путей:** 50% (проверяется только один путь – когда поля пусты)
- **Покрытие функций:** 100% (метод handleBookAction выполнен)
- **Покрытие вход/выход:** 100% (вызов showProductList выполнен)
- **Покрытие комбинаций:** 50% (проверяется только одна комбинация – пустые поля)

4. ProductDetailsController.java

Класс ProductDetailsController содержит два метода: handleBookingAction и handleBackToListAction.

Тест: *ProductDetailsControllerTest.testHandleBookingAction*

- **Покрытие операторов:** 100% (все строки метода handleBookingAction выполнены)
- **Покрытие условий:** 100% (нет условий)
- **Покрытие путей:** 100% (нет условий)
- **Покрытие функций:** 100% (метод handleBookingAction выполнен)
- **Покрытие вход/выход:** 100% (вызов showBooking выполнен)
- **Покрытие комбинаций:** 100% (нет условий)

Тест: *ProductDetailsControllerTest.testHandleBackToListAction*

- **Покрытие операторов:** 100% (все строки метода handleBackToListAction выполнены)
- **Покрытие условий:** 100% (нет условий)
- **Покрытие путей:** 100% (нет условий)
- **Покрытие функций:** 100% (метод handleBackToListAction выполнен)
- **Покрытие вход/выход:** 100% (вызов showProductList выполнен)
- **Покрытие комбинаций:** 100% (нет условий)

5. ProductListController.java

Класс ProductListController содержит метод handleProductDetailsAction, который обрабатывает событие нажатия кнопки для просмотра деталей продукта.

Тест: *ProductListControllerTest.testHandleProductDetailsAction*

- **Покрытие операторов:** 100% (все строки метода handleProductDetailsAction выполнены)
- **Покрытие условий:** 100% (нет условий)
- **Покрытие путей:** 100% (нет условий)
- **Покрытие функций:** 100% (метод handleProductDetailsAction выполнен)
- **Покрытие вход/выход:** 100% (вызов showProductDetails выполнен)
- **Покрытие комбинаций:** 100% (нет условий)

Результаты анализа:

- **Main:** 0% покрытие (нет тестов)
- **LoginController:**
  - **Покрытие операторов:** 100%
  - **Покрытие условий:** 50%
  - **Покрытие путей:** 50%
  - **Покрытие функций:** 100%
  - **Покрытие вход/выход:** 100%
  - **Покрытие комбинаций:** 50%

- **BookingController:**
  - **Покрытие операторов:** 100%
  - **Покрытие условий:** 50%
  - **Покрытие путей:** 50%
  - **Покрытие функций:** 100%
  - **Покрытие вход/выход:** 100%
  - **Покрытие комбинаций:** 50%

- **ProductDetailsController:**
  - **Покрытие операторов:** 100%
  - **Покрытие условий:** 100%
  - **Покрытие путей:** 100%
  - **Покрытие функций:** 100%
  - **Покрытие вход/выход:** 100%
  - **Покрытие комбинаций:** 100%

- **ProductListController:**
  - **Покрытие операторов:** 100%
  - **Покрытие условий:** 100%
  - **Покрытие путей:** 100%
  - **Покрытие функций:** 100%
  - **Покрытие вход/выход:** 100%
  - **Покрытие комбинаций:** 100%

Получим среднюю величину покрытия кода тестами:

1. Покрытие операторов

- **Main:** 0%
- **LoginController:** 100%
- **BookingController:** 100%
- **ProductDetailsController:** 100%
- **ProductListController:** 100%

Среднее покрытие операторов: $\frac{0+100+100+100+100}{5} = \frac{400}{5} = 80\%$

2. Покрытие условий

- **Main:** 0%
- **LoginController:** 50%
- **BookingController:** 50%
- **ProductDetailsController:** 100%
- **ProductListController:** 100%

Среднее покрытие условий: $\frac{0+50+50+100+100}{5} = \frac{300}{5} = 60\%$

3. Покрытие путей

- **Main:** 0%
- **LoginController:** 50%
- **BookingController:** 50%
- **ProductDetailsController:** 100%
- **ProductListController:** 100%

Среднее покрытие путей: $\frac{0+50+50+100+100}{5} = \frac{300}{5} = 60\%$

4. Покрытие функций

- **Main:** 0%
- **LoginController:** 100%
- **BookingController:** 100%
- **ProductDetailsController:** 100%
- **ProductListController:** 100%

Среднее покрытие функций: $\frac{0+100+100+100+100}{5} = \frac{400}{5} = 80\%$

5. Покрытие вход/выход

- **Main:** 0%
- **LoginController:** 100%
- **BookingController:** 100%
- **ProductDetailsController:** 100%
- **ProductListController:** 100%

Среднее покрытие вход/выход: $\frac{0+100+100+100+100}{5} = \frac{400}{5} = 80\%$

6. Покрытие комбинаций

- **Main:** 0%
- **LoginController:** 50%
- **BookingController:** 50%
- **ProductDetailsController:** 100%
- **ProductListController:** 100%

Среднее покрытие комбинаций: $\frac{0+50+50+100+100}{5} = \frac{300}{5} = 60\%$

Среднее значение по всем типам покрытия: $\frac{80+60+60+80+80+60}{6} = \frac{420}{6} = 70\%$

Таким образом, в среднем покрытие кода тестами составляет **70%**.