

КОМИТЕТ ПО ОБРАЗОВАНИЮ ПРАВИТЕЛЬСТВА САНКТ-ПЕТЕРБУРГА

Санкт-Петербургское государственное  
бюджетное профессиональное образовательное учреждение  
«Колледж информационных технологий»

## **О Т Ч Е Т**

**по учебной практике МДК 01.02 «Поддержка и тестирование  
программных модулей»**

**Специальность 09.02.07  
«Информационные системы и программирование»**

**Специализация:  
«Программист»**

Студент группы 493:

Лукьянов И. А.

Преподаватель: Полякова А.Н.

Санкт-Петербург 2023

## СОДЕРЖАНИЕ

1. ПРАКТИЧЕСКАЯ РАБОТА №1 .....	3
1.1. ЗАДАЧА .....	3
1.2. ХОД РАБОТЫ.....	5
2. ПРАКТИЧЕСКАЯ РАБОТА №2 .....	13
2.1. ЗАДАЧА .....	13
2.2. Ход работы .....	13

# **1. ПРАКТИЧЕСКАЯ РАБОТА №1**

## **1.1. ЗАДАЧА**

### **Разработки библиотеки классов**

Для того чтобы в производстве могли быстро и одинаково рассчитывать количество необходимого сырья для производства той или иной продукции, необходимо разработать библиотеку классов.

Данная библиотека будет подключаться к основному проекту и должна быть представлена в виде .dll/.jar файла или папки с файлом .ру.

Чтобы система правильно интегрировалась вам необходимо обязательно следовать правилам именования библиотек, классов и методов в них. В случае ошибок в рамках именования ваша работа не может быть проверена и ваш результат не будет зачтен. Классы и методы должны содержать модификатор public (если это реализуемо в рамках платформы), чтобы внешние приложения могли получить к ним доступ.

В качестве названия для библиотеки необходимо использовать: WSUniversalLib. Вам необходимо загрузить исходный код проекта с библиотекой в отдельный репозиторий с названием, совпадающим с названием проекта.

### **Класс расчета материалов**

Метод должен рассчитывать целое количество сырья, необходимого для производства определенного количества (count) продукции, учитывая возможный брак материалов. Для упрощения расчетов будем считать всю продукцию прямоугольного размера с известными значениями ширины (width) и длины (length).

Количество необходимого качественного сырья на одну единицу продукции рассчитывается как площадь продукции, умноженная на коэффициент типа продукции.

Коэффициенты типа продукции (product\_type):

Тип продукции 1 - 1.1,

Тип продукции 2 - 2.5,

Тип продукции 3 - 8.43.

При этом нужно учитывать процент брака материала в зависимости от его типа (material\_type):

Тип материала 1 - 0.3%,

Тип материала 2 - 0.12%.

При этом если в качестве параметров метода будут приходить несуществующие типы продукции/материалов или другие неподходящие данные, то метод должен вернуть -1.

Например, необходимо изготовить 15 единиц продукции 3 типа шириной 20 и длиной 45 из материала 1 типа. Количество качественного сырья (без учета брака) будет равно 113 805. Однако с учетом возможного брака материалов общее необходимое количество сырья должно быть увеличено до 114 147,442. Округлив полученное значение до ближайшего большего целого, получим 114 148 единиц необходимого сырья. Спецификация метода представлена в отдельном файле в ресурсах.

### **Разработка модульных тестов (Unit-tests)**

Для выполнения процедуры тестирования созданного вами метода библиотеки WSUniversalLib, возвращающего целое количество сырья для производства, вам необходимо создать отдельный проект модульных тестов.

В рамках проекта разработайте тесты, максимально полно покрывающие функционал метода. Ничего страшного, если ваш метод работает не совсем идеально и тесты могут быть не пройдены в связи с этим - в данном модуле это не так важно.

Обратите внимание, что имена тестов должны отражать их суть, т.е. вместо TestMethod1() тест следует назвать, например, GetQuantityForProduct\_NonExistentProductType() для тестирования случая передачи несуществующего типа продукции.

Необходимо разработать модульные тесты, которые на основании исходных данных можно условно разделить на 2 группы следующим образом: 10 методов низкой сложности и 5 методов высокой сложности.

## 1.2. ХОД РАБОТЫ

### 1.2.1 Метод для расчета количества сырья

Было разработано 10 модульных тестов для тестирования созданного метода библиотеки WUniversalLib.

На рисунке 1 изображен метод для расчета количества сырья.

```
Ссылка: 0
public int GetQuantityForProduct(int productType, int materialType, int count, float width, float length)
{
    if (count < 0 || width < 0 || length < 0) return -1;

    float koefType = 0;
    switch (productType)
    {
        case 1:
            koefType = 1.1f;
            break;
        case 2:
            koefType = 2.5f;
            break;
        case 3:
            koefType = 8.43f;
            break;
        default: return -1;
    }

    float brakProcent = 0;
    switch (materialType)
    {
        case 1:
            brakProcent = 0.3f;
            break;
        case 2:
            brakProcent = 0.12f;
            break;
        default: return -1;
    }

    float area = width * length;
    float needCount = count * area * koefType;
    float faultCount = needCount / (1 - (brakProcent / 100));
    return (int)(Math.Ceiling(faultCount));
}
```

Рисунок 1 – Код метода для расчета количества сырья

### 1.2.2. Модульные тесты

#### Тестирование правильности расчетов

Данный тест проверяет правильность расчетов в методе при корректных введенных данных. Код данного метода изображен на рисунке 2.

```
[TestMethod]
Ссылка: 0
public void GetQuantityForProduct_CorrectData()
{
    //Arrange
    Calculation cal = new Calculation();

    //Act
    int res = cal.GetQuantityForProduct(3, 1, 15, 20, 45);

    //Assert
    Assert.AreEqual(114148, res);
}
```

Рисунок 2 – Метод для тестирования правильности расчетов

### Тестирование несуществующего типа продукта

Данный тест проверяет метод на обработку ввода несуществующего типа продукта. Код данного метода изображен на рисунке 3.

```
[TestMethod]
Ссылка: 0
public void GetQuantityForProduct_CorrectData()
{
    //Arrange
    calculation cal = new calculation();

    //Act
    int res = cal.GetQuantityForProduct(3, 1, 15, 20, 45);

    //Assert
    Assert.AreEqual(114148, res);
}
```

Рисунок 3 – Метод для тестирования несуществующего типа продукта

### Тестирование несуществующего типа материала

Данный тест проверяет метод на обработку ввода несуществующего типа материала. Код данного метода изображен на рисунке 4.

```
[TestMethod]
Ссылка: 0
public void GetQuantityForProduct_NonExistentMaterialType()
{
    //Arrange
    calculation cal = new calculation();

    //Act
    int res = cal.GetQuantityForProduct(3, 1000, 15, 20, 45);

    //Assert
    Assert.AreEqual(-1, res);
}
```

Рисунок 4 – Метод для тестирования несуществующего типа материала

### Тестирование отрицательного количества продукции

Данный тест проверяет метод на обработку ввода отрицательного количества продукции. Код данного метода изображен на рисунке 5.

```
[TestMethod]
Ссылка: 0
public void GetQuantityForProduct_CountIsNegative()
{
    //Arrange
    calculation cal = new calculation();

    //Act
    int res = cal.GetQuantityForProduct(3, 1, -1, 20, 45);

    //Assert
    Assert.AreEqual(-1, res);
}
```

Рисунок 5 – Метод для тестирования отрицательного количества продукции

## Тестирование отрицательной ширины

Данный тест проверяет метод на обработку ввода отрицательной ширины. Код данного метода изображен на рисунке 6.

```
[TestMethod]
Ссылка: 0
public void GetQuantityForProduct_widthIsNegative()
{
    //Arrange
    Calculation cal = new Calculation();

    //Act
    int res = cal.GetQuantityForProduct(3, 1, 15, -1, 45);

    //Assert
    Assert.AreEqual(-1, res);
}
```

Рисунок 6 – Метод для тестирования отрицательной ширины

## Тестирование отрицательной длины

Данный тест проверяет метод на обработку ввода отрицательной длины. Код данного метода изображен на рисунке 7.

```
[TestMethod]
Ссылка: 0
public void GetQuantityForProduct_LengthIsNegative()
{
    //Arrange
    Calculation cal = new Calculation();

    //Act
    int res = cal.GetQuantityForProduct(3, 1, 15, 20, -1);

    //Assert
    Assert.AreEqual(-1, res);
}
```

Рисунок 7 – Метод для тестирования отрицательной длины

## Тестирование округления к наименьшему целому

Данный тест проверяет метод на округление количества с учетом погрешности брака к меньшему целому. Код данного метода изображен на рисунке 8.

```
[TestMethod]
Ссылка: 0
public void GetQuantityForProduct_ValueIsRoundedDown()
{
    //Arrange
    Calculation cal = new Calculation();

    //Act
    int res = cal.GetQuantityForProduct(3, 1, 15, 20, 45);
    float koefType = 8.43f;
    float faultProc = 0.3f;

    float resCountWithFaultProc = (15 * (20 * 45) * koefType) / (1 - (faultProc / 100));
    int valueRoundedDown = (int)Math.Ceiling(resCountWithFaultProc);

    //Assert
    Assert.AreEqual(valueRoundedDown, res);
}
```

Рисунок 8 – Метод для тестирования округления к наименьшему целому

## Тестирование нулевого количества

Данный тест проверяет метод на обработку ввода нулевого количества. Код данного метода изображен на рисунке 9.

```
[TestMethod]
Ссылка: 0
public void GetQuantityForProduct_CountIsNull()
{
    //Arrange
    Calculation cal = new Calculation();

    //Act
    int res = cal.GetQuantityForProduct(3, 1, 0, 20, 45);

    //Assert
    Assert.AreEqual(-1, res);
}
```

Рисунок 9 – Метод для тестирования нулевого количества

## Тестирование нулевой ширины

Данный тест проверяет метод на обработку ввода нулевой ширины. Код данного метода изображен на рисунке 10.

```
[TestMethod]
public void GetQuantityForProduct_widthIsNull()
{
    //Arrange
    Calculation cal = new Calculation();

    //Act
    int res = cal.GetQuantityForProduct(3, 1, 15, 0, 45);

    //Assert
    Assert.AreEqual(-1, res);
}
```

Рисунок 10 – Метод для тестирования нулевой ширины

## Тестирование нулевой длины

Данный тест проверяет метод на обработку ввода нулевой длины. Код данного метода изображен на рисунке 11.

```
[TestMethod]
Ссылка: 0
public void GetQuantityForProduct_LengthIsNull()
{
    //Arrange
    Calculation cal = new Calculation();

    //Act
    int res = cal.GetQuantityForProduct(3, 1, 15, 20, 0);

    //Assert
    Assert.AreEqual(-1, res);
}
```

Рисунок 11 – Метод для тестирования нулевой количества



## Успешность методов тестирования

На рисунке 12 показан результат выполнения всех методов тестирования.

Тестирование	Длительн...	Признаки	Сообщение об ошибке
WSUniversalLibUnitTest (10)	299 мс		
WSUniversalLibUnitTest (10)	299 мс		
WSUniversalLibUnitTest (10)	299 мс		
GetQuantityForProduct_CorrectData	16 мс		
GetQuantityForProduct_CountIsNegative	< 1 мс		
GetQuantityForProduct_CountIsNull	280 мс		Сбой Assert.AreEqual. Ожидается: <-1>. Фактически: <0>.
GetQuantityForProduct_LengthIsNegative	< 1 мс		
GetQuantityForProduct_LengthIsNull	2 мс		Сбой Assert.AreEqual. Ожидается: <-1>. Фактически: <0>.
GetQuantityForProduct_NonExistentMateri...	< 1 мс		
GetQuantityForProduct_NonExistentProdu...	< 1 мс		
GetQuantityForProduct_ValuelsRoundedD...	< 1 мс		
GetQuantityForProduct_WidthIsNegative	< 1 мс		
GetQuantityForProduct_WidthIsNull	1 мс		Сбой Assert.AreEqual. Ожидается: <-1>. Фактически: <0>.

Рисунок 12 – Результат выполнения методов тестирования

### 1.2.3. Тестирование «Test-Case»

#### Тестовый пример 1

На рисунке 13 показан первый тестовый пример.

Тестовый пример #1:

Тестовый пример #	1
Приоритет тестирования	Высокий
Заголовок/название теста	Возможность запуска
Краткое изложение теста	Возможность запуска проекта с правильными данными
Этапы теста	1. Ввести тип продукта: «3» 2. Ввести тип материала: «1» 3. Ввести количество: «15» 4. Ввести ширину: «20» 5. Ввести длину: «45» 6. Запустить метод
Тестовые данные	Тип продукта: «3», Тип материала: «1», Количество: «15», Ширина: «20», Длина: «45»
Ожидаемый результат	Проект запуститься и вернёт целое число.
Фактический результат	Проект запустился и вернул целое число.
Статус	Зачет
Предварительное условие	
Постусловие	
Примечания/комментарии	

Рисунок 13 – Первый тестовый пример

## Тестовый пример 2

На рисунке 14 показан второй тестовый пример.

Тестовый пример #2:

Тестовый пример #	2
Приоритет тестирования	Высокий
Заголовок/название теста	Проверка правильности расчетов в проекте
Краткое изложение теста	Правильность расчетов метода с корректными данными
Этапы теста	<ol style="list-style-type: none"><li>1. Ввести тип продукта: «3»</li><li>2. Ввести тип материала: «1»</li><li>3. Ввести количество: «15»</li><li>4. Ввести ширину: «20»</li><li>5. Ввести длину: «45»</li><li>6. Запустить метод</li></ol>
Тестовые данные	Тип продукта: «3», Тип материала: «1», Количество: «15», Ширина: «20», Длина: «45»
Ожидаемый результат	Проект запуститься и вернёт целое число: «114148»
Фактический результат	Проект запустился и вернул целое число: «114148»
Статус	Зачет
Предварительное условие	
Постусловие	
Примечания/комментарии	

Рисунок 14 – Второй тестовый пример

## Тестовый пример 3

На рисунке 15 показан третий тестовый пример.

Тестовый пример #3:

Тестовый пример #	3
Приоритет тестирования	Высокий
Заголовок/название теста	Проверка на несуществующий тип продукта
Краткое изложение теста	Проверка на контроль за существованием типа продукта
Этапы теста	<ol style="list-style-type: none"> <li>1. Ввести тип продукта: «1000»</li> <li>2. Ввести тип материала: «1»</li> <li>3. Ввести количество: «15»</li> <li>4. Ввести ширину: «20»</li> <li>5. Ввести длину: «45»</li> <li>6. Запустить метод</li> </ol>
Тестовые данные	Тип продукта: «1000», Тип материала: «1», Количество: «15», Ширина: «20», Длина: «45»
Ожидаемый результат	Проект запуститься и вернёт целое число: «-1»
Фактический результат	Проект запустился и вернул целое число: «-1»
Статус	Зачет
Предварительное условие	
Постусловие	
Примечания/комментарии	

Рисунок 15 – Третий тестовый пример

## Тестовый пример 4

На рисунке 16 показан четвертый тестовый пример.

Тестовый пример #4:

Тестовый пример #	4
Приоритет тестирования	Высокий
Заголовок/название теста	Проверка на отрицательные значения
Краткое изложение теста	Контроль на отрицательность введенных значений
Этапы теста	<ol style="list-style-type: none"> <li>1. Ввести тип продукта: «3»</li> <li>2. Ввести тип материала: «1»</li> <li>3. Ввести количество: «-1»</li> <li>4. Ввести ширину: «20»</li> <li>5. Ввести длину: «45»</li> <li>6. Запустить метод</li> </ol>
Тестовые данные	Тип продукта: «3», Тип материала: «1», Количество: «-1», Ширина: «20», Длина: «45»
Ожидаемый результат	Проект запуститься и вернёт целое число: «-1»
Фактический результат	Проект запустился и вернул целое число: «-1»
Статус	Зачет
Предварительное условие	
Постусловие	
Примечания/комментарии	

Рисунок 16 – Четвертый тестовый пример

## Тестовый пример 5

На рисунке 17 показан пятый тестовый пример.

Test case #5:

Test Case #	5
Приоритет тестирования	Высокий
Заголовок/название теста	Проверка на отрицательные значения
Краткое изложение теста	Контроль над нулевыми значениями
Этапы теста	<ol style="list-style-type: none"> <li>1. Ввести тип продукта: «3»</li> <li>2. Ввести тип материала: «1»</li> <li>3. Ввести количество: «-0»</li> <li>4. Ввести ширину: «20»</li> <li>5. Ввести длину: «45»</li> <li>6. Запустить метод</li> </ol>
Тестовые данные	Тип продукта: «3», Тип материала: «1», Количество: «0», Ширина: «20», Длина: «45»
Ожидаемый результат	Проект запуститься и вернёт целое число: «-1»
Фактический результат	Проект запустился и вернул целое число: «0»
Статус	Незачет
Предварительное условие	
Постусловие	
Примечания/комментарии	

Рисунок 17 – Пятый тестовый пример

## **2. ПРАКТИЧЕСКАЯ РАБОТА №2**

### **2.1. ЗАДАЧА**

Согласно своему варианту по МДК 01.01 выполнить интеграционное тестирование:

- 5 unit-tests (связь между модулями)
- 10 test-case
- провести тестирование по test-case

Необходимо описать свой программный продукт.

### **2.2. Ход работы**

#### **Описание программного продукта:**

Информационная система ООО «Синема» – прикладное программное обеспечение, использующееся для выдачи билетов на киносеансы, просмотра информации по киносеансу, а также предназначена для решения следующих задач:

- контроль документооборота;
- формирование необходимой информации по итогам выдачи билета клиенту (показываемый фильм, дата сеанса, время сеанса, зал и место, дата продажи билета, цена за билет).

Основными целями внедрения системы являются:

- упрощение контроля выдачи билетов;
- полный контроль над статусом киносеанса;
- минимизация человеческого фактора при ведении документов;
- снижение рутинной работы при работе с документами.

Исходная информация в систему поступает из запросов клиента. Входными данными являются следующая информация:

- сведения о фильме;
- сведения о дате киносеанса;
- сведения о времени киносеанса;
- сведения о месте в зале (ряд и номер места).

Рассмотрим определение прецедентов (вариантов использования). Система требуется, прежде всего, следующим заинтересованным лицам:

- оператор кассового отделения;
- гость;
- менеджер;
- системный администратор.

Функционал каждой роли:

Системный администратор обладает возможностью просмотра, редактирования, добавления, удаления данных. Ко всему прочему системный администратор имеет возможность создавать пользователей для системы, удалять их или же изменять их учётные данные.

Гость может только просматривать данные.

Система должна функционировать в многопользовательском режиме, поэтому каждый пользователь кроме гостя должен иметь свой пароль доступа в систему.

Проект предназначен для кинотеатра, занимающегося выдачей билетов на киносеансы.

Внедрение данного проекта поможет автоматизировать процесс выдачи билетов, способствуя снижению рутинной работы с ведением документооборота и минимизации случаев человеческого фактора.

## Unit-тесты

Все пройденные тесты представлены на рисунке 18

Тестирование	Длит...	Признаки	Сообще...	Сводка по группе
▲ ✓ UnitTestProject (5)	4,8 с			UnitTestProject
▲ ✓ UnitTestProject (5)	4,8 с			Тесты в группе: 5
▲ ✓ UnitTest1 (5)	4,8 с			🕒 Общая длительность: 4,8 с
✓ Auth_GoToFormSeance_IsTrue	4,3 с			Результаты
✓ FormScreenshot_ShowAllScreenshotInFilm_IsTrue	99 мс			✓ 5 пройден
✓ OnFilm_GoToFormFilm_FilmAreEqual	15 мс			
✓ OnHallView_ShowAllPlacesInHall_IsTrue	329 мс			
✓ OnSeance_GoToFormSeance_SeanceAreEqual	8 мс			

Рисунок 18 – Пройденные тесты

Первый тест – связь между модулем авторизации и переход к форме списка сеансов. Данный тест представлен на рисунке 19.

```
//Тестирование, что переход происходит на правильную форму
[TestMethod]
| Ссылки: 0
public void Auth_GoToFormSeance_IsTrue()
{
    //Arrange
    FormAuth formAuth = new FormAuth();

    //Act
    Form formRes = formAuth.Auth("123", "123");
    string res = formRes.GetType().Name;

    //Assert
    Assert.IsTrue(res == "FormSeance");
}
```

Рисунок 19 – Тестирование связи модуля авторизации и модуля сеансов

Второй тест – проверка на то, что показывается форма выбранного сеанса, связь между модулем списка сеансов и модуля сеанса. Данный тест представлен на рисунке 20.

```
//Тестирование того, что сеанс переданся корректно
[TestMethod]
| Ссылки: 0
public void OnSeance_GoToFormSeance_SeanceAreEqual()
{
    //Arrange
    int seanceId = 1;

    //Act
    FormOrderSeance formRes = FormSeance.OnSeance(seanceId);
    Seance res = formRes.seance;

    //Assert
    Assert.AreEqual(seanceId, res.SeanceId);
}
```

Рисунок 20 – Тестирование связи модуля списка сеансов и модуля сеанса

Третий тест – проверка на то, что показывается выбранный фильм, связь между модулем списка сеансов и модуля фильма. Данный тест представлен на рисунке 21.

```
//Тестирование того, что фильм переданся корректно
[TestMethod]
| Ссылки: 0
public void OnFilm_GoToFormFilm_FilmAreEqual()
{
    //Arrange
    int filmId = 1;

    //Act
    FormFilm formRes = FormSeance.OnFilm(filmId);
    Film res = formRes.film;

    //Assert
    Assert.AreEqual(filmId, res.FilmId);
}
```

Рисунок 21 – Тестирование связи модуля списка сеансов и модуля фильма

Четвертый тест – связь между модулем выбора сеанса и отображения списка мест в данном сеансе. Данный тест представлен на рисунке 22.

```
//Тестирование того, что данные о зале переданы и
//все места зала отображены
[TestMethod]
[Ссылка: 0]
public void OnHallView_ShowAllPlacesInHall_IsTrue()
{
    //Arrange
    DB DB = new DB();
    int seanceId = 1;
    Seance seance = DB.Seance.Where(x => x.SeanceId == seanceId).FirstOrDefault();
    seance.HallId = 1;
    DB.SaveChanges();
    List<Place> places = DB.Place.Where(x => x.HallId == seance.HallId).ToList();

    int countPlaces = 0;

    //Act
    FormOrderSeance formOrderSeance = new FormOrderSeance(seanceId);
    DataGridView placesview = formOrderSeance.OnHallView();
    foreach (Place place in places)
    {
        for (int i = 0; i < placesview.Rows.Count; i++)
        {
            for (int j = 0; j < placesview.Columns.Count; j++)
            {
                if (place.PlaceId == (int)placesview[j, i].Tag)
                    countPlaces++;
            }
        }
    }

    //Assert
    Assert.AreEqual(places.Count, countPlaces);
}
```

Рисунок 22 – Тестирования связи модуля списка сеансов и модуля списка мест

Пятый тест – связь между модулем выбора фильма и отображения скриншотов фильма. Данный тест представлен на рисунке 23.

```
//Тестирование что данные о фильме переданы и все скриншоты отображены
[TestMethod]
[Ссылка: 0]
public void FormScreenshot_ShowAllScreenshotInFilm_IsTrue()
{
    //Arrange
    DB DB = new DB();
    int filmId = 2;
    List<Screenshot> screenshots = DB.Screenshot.Where(x => x.FilmId == filmId).ToList();

    int countScreenshots = 0;

    //Act
    FormScreenshot formScreenshot = new FormScreenshot(filmId);
    List<Screenshot> res = formScreenshot.binaryScreenshots;
    foreach (Screenshot item in screenshots)
    {
        if (res.FindIndex(x => x.ScreenshotId == item.ScreenshotId) != -1)
            countScreenshots++;
    }

    //Assert
    Assert.AreEqual(screenshots.Count, countScreenshots);
}
```

Рисунок 23 – Тестирования связи модуля фильма и модуля скриншотов



## Test-case

Тестовый пример #1:

Тестовый пример #	1
Приоритет тестирования	Высокий
Заголовок/название теста	Возможность запуска приложения
Краткое изложение теста	Возможность запуска установленного приложения
Этапы теста	1. Открыть файл с именем «Cinema.exe»
Тестовые данные	
Ожидаемый результат	Приложение запустилось и отобразило форму авторизации
Фактический результат	Приложение запустилось и отобразило форму авторизации
Статус	Зачет
Предварительное условие	Операционная система Windows 10 64-bit, установленное приложение ООО «Синема»
Постусловие	
Примечания/комментарии	

Тестовый пример #2:

Тестовый пример #	2
Приоритет тестирования	Высокий
Заголовок/название теста	Возможность подключения к базе данных
Краткое изложение теста	Возможность первого подключения к базе данных
Этапы теста	1. Запустить приложение 2. Нажать кнопку «Войти»
Тестовые данные	
Ожидаемый результат	Появится сообщение о неверном логине или пароле
Фактический результат	Появилось сообщение о неверном логине или пароле
Статус	Зачет
Предварительное условие	Установленное приложение ООО «Синема»
Постусловие	
Примечания/комментарии	В случае, если подключение бы не удалось, то появилось бы сообщение об ошибке, или же о безуспешной попытке соединения с базой данных

Тестовый пример #3:

Тестовый пример #	3
Приоритет тестирования	Высокий
Заголовок/название теста	Возможность авторизации
Краткое изложение теста	Прохождение этапа авторизации с корректными данными
Этапы теста	1. Ввести логин: «123» 2. Ввести пароль: «123»
Тестовые данные	Логин: «123», Пароль: «123»
Ожидаемый результат	Появится сообщение о успешном входе, затем откроется форма списка сеансов
Фактический результат	Появилось сообщение о успешном входе, затем открылась форма списка сеансов
Статус	Зачет
Предварительное условие	Установленное и запущенное приложение ООО «Синема»
Постусловие	В базу данных добавилась запись о входе пользователя «123»
Примечания/комментарии	

Тестовый пример #4:

Тестовый пример #	4
Приоритет тестирования	Высокий
Заголовок/название теста	Получение списка сеансов
Краткое изложение теста	Проверка на автоматическое формирование списка киносеансов за выбранную дату
Этапы теста	1. Ввести логин: «123» 2. Ввести пароль: «123» 3. Выбрать дату показа: «27.08.2023»
Тестовые данные	Логин: «123», Пароль: «123», Дата показа: «27.08.2023»
Ожидаемый результат	На форме киносеансов отобразятся киносеансы за выбранную дату
Фактический результат	На форме киносеансов отобразились киносеансы за выбранную дату
Статус	Зачет
Предварительное условие	Установленное и запущенное приложение ООО «Синема»
Постусловие	В базу данных добавилась запись о входе пользователя «123»
Примечания/комментарии	

## Test case #5:

Test Case #	5
Приоритет тестирования	Высокий
Заголовок/название теста	Открытие формы зала киносеанса
Краткое изложение теста	Проверка на правильное отображение мест в зале выбранного киносеанса
Этапы теста	<ol style="list-style-type: none"> <li>1. Ввести логин: «123»</li> <li>2. Ввести пароль: «123»</li> <li>3. Выбрать дату показа: «27.08.2023»</li> <li>4. Выбрать фильм «Король Лев» со временем: «16:00»</li> </ol>
Тестовые данные	Логин: «123», Пароль: «123», Дата показа: «27.08.2023», Сеанс: «Король Лев – 16:00»
Ожидаемый результат	Откроется форма зала со списком мест
Фактический результат	Открылась форма зала со списком мест
Статус	Зачет
Предварительное условие	Установленное и запущенное приложение ООО «Синема»
Постусловие	В базу данных добавилась запись о входе пользователя «123»
Примечания/комментарии	

## Test case #6:

Test Case #	6
Приоритет тестирования	Высокий
Заголовок/название теста	Получение списка занятых мест киносеанса
Краткое изложение теста	При отображении мест будут недоступны места, которые уже заняты
Этапы теста	<ol style="list-style-type: none"> <li>1. Ввести логин: «123»</li> <li>2. Ввести пароль: «123»</li> <li>3. Выбрать дату показа: «27.08.2023»</li> <li>4. Выбрать фильм «Король Лев» со временем: «16:00»</li> </ol>
Тестовые данные	Логин: «123», Пароль: «123», Дата показа: «27.08.2023», Сеанс: «Король Лев – 16:00»
Ожидаемый результат	Во втором ряду первое место будет занято
Фактический результат	Во втором ряду первое место будет занято
Статус	Зачет
Предварительное условие	Установленное и запущенное приложение ООО «Синема», в таблице «Билеты» должна содержаться запись о сеансе «Король Лев – 16:00» с первым местом во втором ряду

Постусловие	В базу данных добавилась запись о входе пользователя «123»
Примечания/комментарии	

Test case #7:

Test Case #	7
Приоритет тестирования	Высокий
Заголовок/название теста	Создание билета
Краткое изложение теста	Проверка на формирования печатного вида билета на киносеанс после его покупки
Этапы теста	<ol style="list-style-type: none"> <li>1. Ввести логин: «123»</li> <li>2. Ввести пароль: «123»</li> <li>3. Выбрать дату показа: «27.08.2023»</li> <li>4. Выбрать фильм «Король Лев» со временем: «16:00»</li> <li>5. Выбрать первое место в первом ряду</li> <li>6. Нажать кнопку «Купить»</li> <li>7. Перейти в папку проекта</li> <li>8. Открыть файл с именем: «Сеанс-Король Лев, Время-16_00, Билет-Ряд1Место1.pdf»</li> </ol>
Тестовые данные	Логин: «123», Пароль: «123», Дата показа: «27.08.2023», Сеанс: «Король Лев – 16_00»
Ожидаемый результат	Билет сформируется и будет корректно отображаться
Фактический результат	Билет сформировался, однако имя файла было задано некорректно
Статус	Незачет
Предварительное условие	Установленное и запущенное приложение ООО «Синема», в таблице «Билеты» не должна содержаться запись о сеансе «Король Лев – 16:00» с первым местом в первом ряду
Постусловие	В базу данных добавилась запись о входе пользователя «123», был создан файл в папке проекта «Сеанс-Король Лев, Время-16_00, Билет-Ряд1Место1.pdf»
Примечания/комментарии	

Test case #8:

Test Case #	8
Приоритет тестирования	Высокий
Заголовок/название теста	Отображение информации о фильме
Краткое изложение теста	Проверка на отображение информации о фильме при его выборе в списке
Этапы теста	<ol style="list-style-type: none"> <li>1. Ввести логин: «123»</li> <li>2. Ввести пароль: «123»</li> <li>3. Выбрать дату показа: «27.08.2023»</li> <li>4. Выбрать фильм «Король Лев»</li> <li>5. Нажать на обложку фильма</li> </ol>
Тестовые данные	Логин: «123», Пароль: «123», Дата показа: «27.08.2023», Сеанс: «Король Лев – 16:00»
Ожидаемый результат	Откроется форма фильма с информацией о фильме «Король Лев»
Фактический результат	Открылась форма фильма с информацией о фильме «Король Лев»
Статус	Зачет
Предварительное условие	Установленное и запущенное приложение ООО «Синема»
Постусловие	В базу данных добавилась запись о входе пользователя «123»
Примечания/комментарии	

Test case #9:

Test Case #	9
Приоритет тестирования	Высокий
Заголовок/название теста	Отображение скриншотов фильма
Краткое изложение теста	Проверка на отображение скриншотов фильма
Этапы теста	<ol style="list-style-type: none"> <li>1. Ввести логин: «123»</li> <li>2. Ввести пароль: «123»</li> <li>3. Выбрать дату показа: «27.08.2023»</li> <li>4. Выбрать фильм «Король Лев»</li> <li>5. Нажать на обложку фильма</li> <li>6. Нажать на кнопку «Скриншоты»</li> </ol>
Тестовые данные	Логин: «123», Пароль: «123», Дата показа: «27.08.2023», Сеанс: «Король Лев – 16:00»
Ожидаемый результат	Откроется форма со скриншотами фильма «Король Лев»
Фактический результат	Открылась форма со скриншотами фильма «Король Лев»
Статус	Зачет

Предварительное условие	Установленное и запущенное приложение ООО «Синема»
Постусловие	В базу данных добавилась запись о входе пользователя «123»
Примечания/комментарии	

Test case #10:

Test Case #	10
Приоритет тестирования	Высокий
Заголовок/название теста	Отображение списка пользователей
Краткое изложение теста	Проверка на корректное отображение списка пользователей
Этапы теста	<ol style="list-style-type: none"> <li>1. Ввести логин: «321»</li> <li>2. Ввести пароль: «321»</li> <li>3. Нажать на кнопку «Управление данными»</li> <li>4. Нажать на кнопку «Пользователи»</li> </ol>
Тестовые данные	Логин: «321», Пароль: «321»
Ожидаемый результат	Откроется форма со списком пользователей системы
Фактический результат	Открылась форма со списком пользователей системы
Статус	Зачет
Предварительное условие	Установленное и запущенное приложение ООО «Синема»
Постусловие	В базу данных добавилась запись о входе пользователя «321»
Примечания/комментарии	

### Прохождения тестового сценария

Для прохождения был выбран тестовый пример №7: Создание билета, так как охватывал большую часть модулей программы.

Первым шагом необходимо запустить приложение (рис. 24)

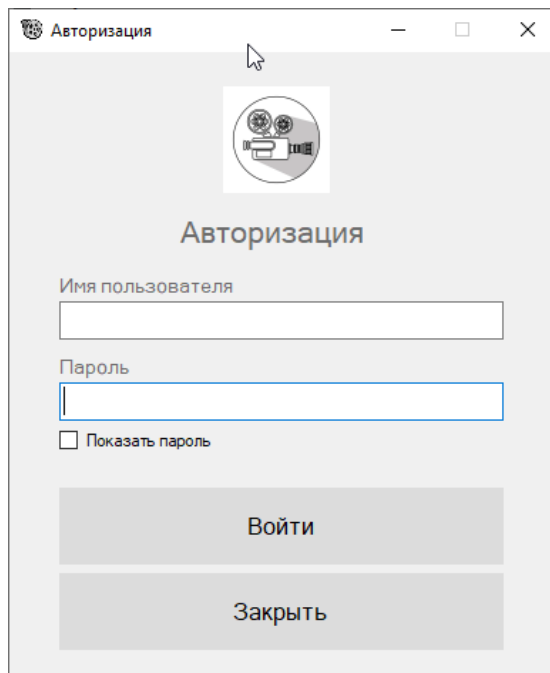


Рисунок 24 – Запущенное приложение

Далее необходимо ввести логин: «123», и пароль: «123». Затем нажать кнопку «Войти» (рис. 25).

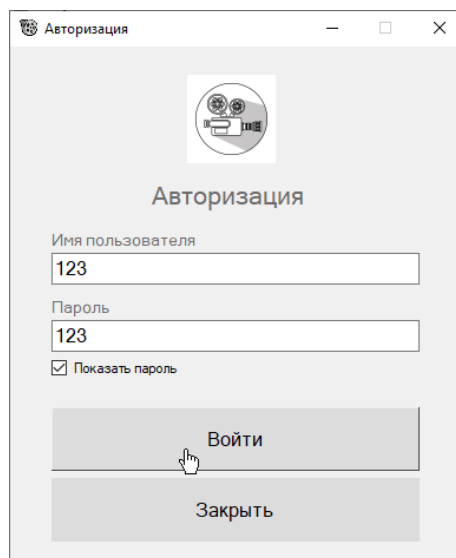


Рисунок 25 – Данные для авторизации

Появилось сообщение о успешной авторизации (рис. 26).

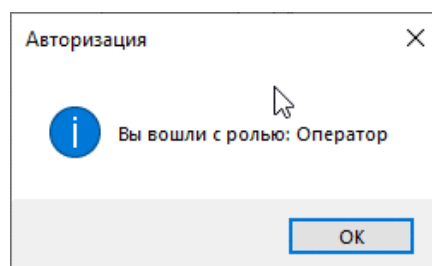


Рисунок 26 – Сообщение об успешной авторизации

В появившемся списке сеансов необходимо выбрать дату: «27.08.2023» и сеанс: «Король Лев – 16:00» (рис. 27).

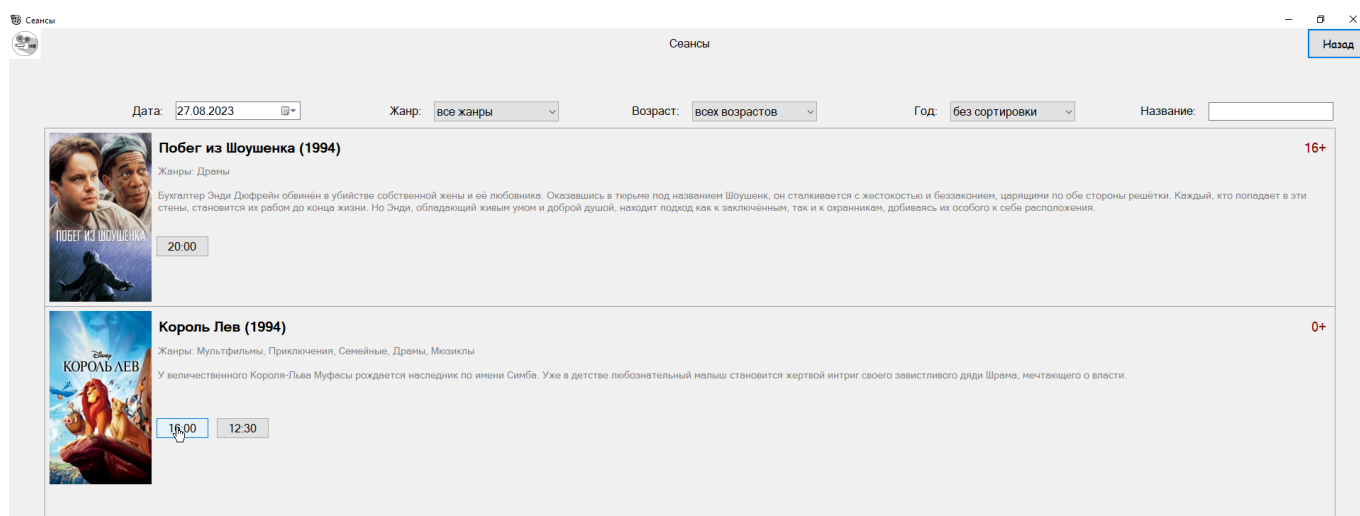


Рисунок 27 – Выбор сеанса

В появившемся окне сеанса необходимо выбрать первое место в первом ряду (рис. 28):

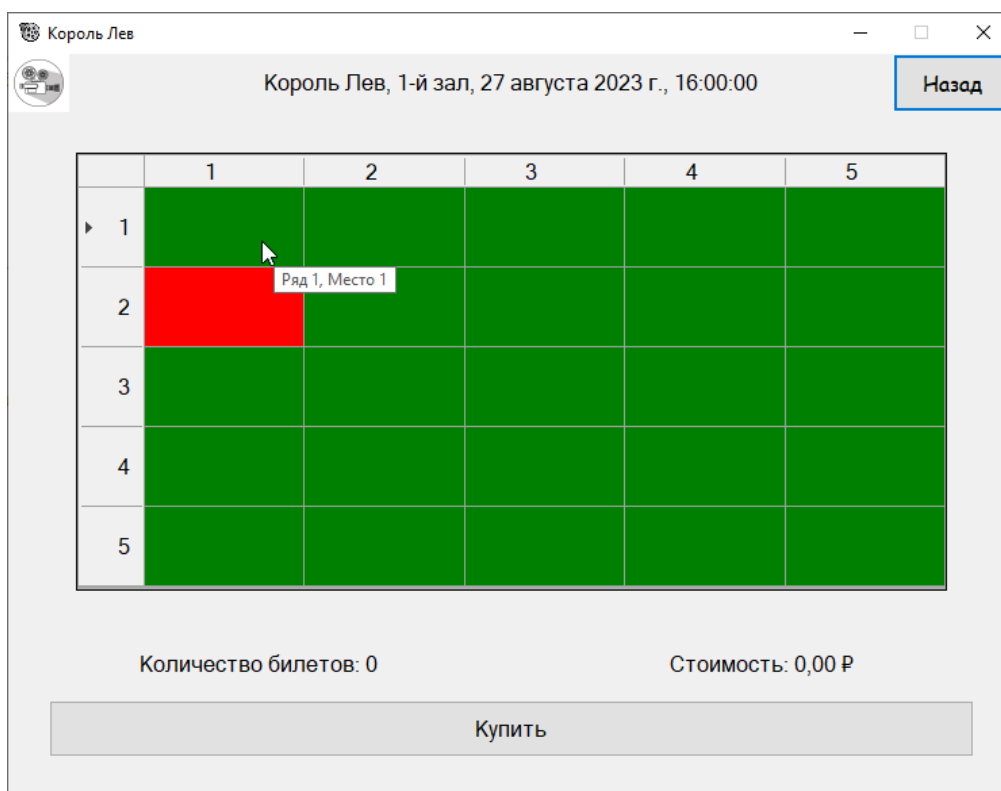


Рисунок 28 – Выбор места

После выбора необходимо нажать кнопку «Купить» (рис. 29).



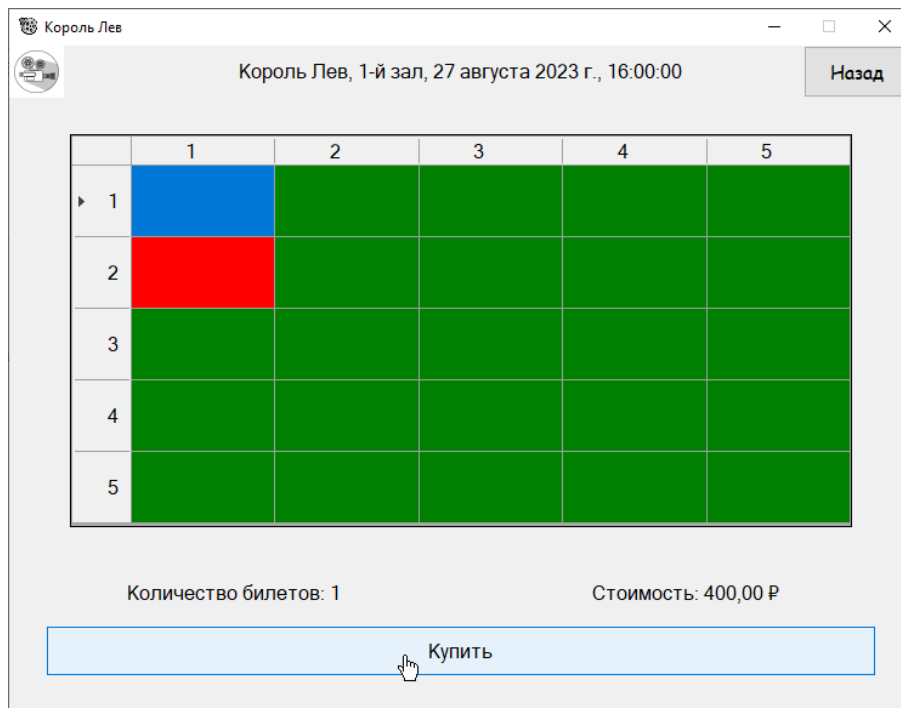


Рисунок 29 – Покупка билета

Далее необходимо перейти в папку проекта (рис. 30)

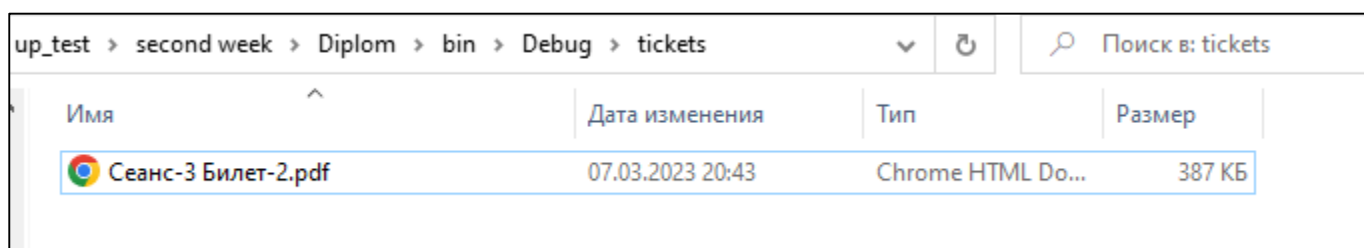


Рисунок 30 – Папка проекта

Как мы видим, файл не именуется «Сеанс-Король Лев, Время-16\_00, Билет-Ряд1Место1.pdf», необходимо внести исправление в проект.

### Исправление ошибки

Путь для сохранения файла выглядит следующим образом:

```
Application.StartupPath + "\\tickets\\" + "Сеанс-" + seance.SeanceId + " Билет-" + ticket.TicketId + ".pdf"
```

Нам необходимо включить в него название фильма, время, а также место и ряд билета, таким образом, новый путь будет выглядеть так:

```
Application.StartupPath + "\\tickets\\" + "Сеанс-" + seance.Film.FilmName + ", Время-" + seancetime + ", Билет-Ряд" + ticket.Place.PlaceRow + "Место" + ticket.Place.PlaceNumber + ".pdf"
```

На рисунке 31 изображен сохранённый файл с исправленным именем.


up_test > second week > Diplom > bin > Debug > tickets			Поиск в: tickets
Имя	Дата изменения	Тип	
 Сеанс-Король Лев, Время-16_00, Билет-Ряд1Место1.pdf	07.03.2023 21:02	Chrome HTML Do...	

Рисунок 31 – Исправленное имя файла

Тестовый пример №7 после исправления:

Test case #7:

Test Case #	7
Приоритет тестирования	Высокий
Заголовок/название теста	Создание билета
Краткое изложение теста	Проверка на формирования печатного вида билета на киносеанс после его покупки
Этапы теста	<ol style="list-style-type: none"> <li>1. Ввести логин: «123»</li> <li>2. Ввести пароль: «123»</li> <li>3. Выбрать дату показа: «27.08.2023»</li> <li>4. Выбрать фильм «Король Лев» со временем: «16:00»</li> <li>5. Выбрать первое место в первом ряду</li> <li>6. Нажать кнопку «Купить»</li> <li>7. Перейти в папку проекта</li> <li>8. Открыть файл с именем: «Сеанс-Король Лев, Время-16_00, Билет-Ряд1Место1.pdf»</li> </ol>
Тестовые данные	Логин: «123», Пароль: «123», Дата показа: «27.08.2023», Сеанс: «Король Лев – 16_00»
Ожидаемый результат	Билет сформируется и будет корректно отображаться
Фактический результат	Билет сформировался и корректно отобразился
Статус	Зачет
Предварительное условие	Установленное и запущенное приложение ООО «Синема», в таблице «Билеты» не должна содержаться запись о сеансе «Король Лев – 16:00» с первым местом в первом ряду
Постусловие	В базу данных добавилась запись о входе пользователя «123», был создан файл в папке проекта «Сеанс-Король Лев, Время-16_00, Билет-Ряд1Место1.pdf»
Примечания/комментарии	

Ссылка на github: [https://github.com/Kazel321/test\\_up](https://github.com/Kazel321/test_up)