Министерство образования и науки РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

"Тверской государственный технический университет"

(ФГБОУ ВО "ТвГТУ")

Кафедра "Программное обеспечение"

Дисциплина: "Теоретическая информатика"

Отчёт по лабораторной работе №1

Выполнили: студенты группы Б.ПИН.РИС 24.06

Полосков А.Д. Дубков С.В. Иванов М.А. Ткаченко А.А

Проверила: Лисничук Арина Бахытжановна

Тверь 2025

Оглавление

[**ВВЕДЕНИЕ** 3](#_Toc210257247)

[**Операция №1: добавление** 3](#_Toc210257248)

[**Операция №2: перенос задач** 4](#_Toc210257249)

[**Операция №3: фильтрация по дедлайну** 5](#_Toc210257250)

[**Реализация приложения "Планировщик задач"** 6](#_Toc210257251)

[**Диаграмма Классов** 9](#_Toc210257252)

[**Тесты** 9](#_Toc210257253)

[**Ссылка на git** 14](#_Toc210257254)

# **ВВЕДЕНИЕ**

**Актуальность:**

* Данный проект направлен на разработку ядра логики для планировщика задач средней сложности. Основная цель — реализовать ключевые функции управления задачами, гарантируя при этом целостность данных через строгую валидацию.

**Цели работы:**

* Освоить методику формального описания операций через предусловия (Pre) и постусловия (Post) с последующей демонстрацией их корректности непосредственно в пользовательском интерфейсе.
* Сформировать навык четкого определения спецификаций операций в терминах предусловий (Pre) и постусловий (Post) и реализовать механизм визуализации их выполнения в интерфейсе системы.

**Задача:**

* Реализовать WPF-приложение «Планировщик задач», включающее в себя 3 операции: добавление, перенос задач, фильтрация по дедлайну. Добавить визуализацию предусловий и постусловий в интерфейсе.

# **Операция №1: добавление**

**Контракт операции**

**Pre (предусловие):**

1. Задача не null
2. Название задачи не пустое и не состоит только из пробелов
3. Дедлайн задачи находится в будущем (позже текущего времени)
4. Приоритет задачи находится в диапазоне от 1 до 5

**Post (постусловие):**

1. Задача добавлена в коллекцию планировщика
2. Коллекция содержит добавленную задачу
3. Размер коллекции увеличился на 1

**Эффекты/исключения:**  
Если предусловие не выполнено, выбрасывается исключение ArgumentException и операция прерывается с сообщением.

**Граничные примеры**

**Валидный вход:**

1. **Вход:**
   * Название: "Важная встреча"
   * Дедлайн: DateTime.Now.AddDays(1)
   * Приоритет: 3
2. **Ожидаемый результат:** Задача успешно добавлена в планировщик
3. **Индикаторы:** Pre — зелёный, Post — зелёный

**Невалидный вход:**

1. **Вход:**
   * Название: "" (пустая строка)
   * Дедлайн: DateTime.Now.AddDays(1)
   * Приоритет: 3
2. **Ожидаемое поведение:** Операция не выполняется, выводится сообщение об ошибке "Название задачи не может быть пустым"
3. **Индикаторы:** Pre — красный, Post — красный

# **Операция №2: перенос задач**

**Контракт операции**

**Pre (предусловие):**

1. Задача не null
2. Задача существует в коллекции планировщика
3. Новый дедлайн находится в будущем (позже текущего времени)
4. Новый дедлайн отличается от текущего дедлайна задачи

**Post (постусловие):**

1. Дедлайн задачи изменен на новый
2. Старый дедлайн не равен новому дедлайну
3. Задача остается в той же позиции в коллекции

**Эффекты/исключения:**  
Если предусловие не выполнено, выбрасывается исключение ArgumentException и операция прерывается с сообщением.

**Граничные примеры**

**Валидный вход:**

1. **Вход:**
   * Существующая задача из коллекции
   * Новый дедлайн: DateTime.Now.AddDays(3)
2. **Ожидаемый результат:** Дедлайн задачи изменен на новый
3. **Индикаторы:** Pre — зелёный, Post — зелёный

**Невалидный вход:**

1. **Вход:**
   * Задача: null
   * Новый дедлайн: DateTime.Now.AddDays(3)
2. **Ожидаемое поведение:** Операция не выполняется, выводится сообщение об ошибке "Задача не может быть null"
3. **Индикаторы:** Pre — красный, Post — красный

# **Операция №3: фильтрация по дедлайну**

**Контракт операции**

**Pre (предусловие):**

1. Критерий фильтрации по дедлайну активен (true)
2. В коллекции есть хотя бы одна задача для фильтрации

**Post (постусловие):**

1. Возвращается не-null коллекция задач
2. Все задачи в результате имеют дедлайн не позже сегодняшнего дня
3. Незавершенные задачи с просроченным дедлайном включены в результат
4. Результат содержит только задачи, соответствующие критерию фильтрации

**Эффекты/исключения:**  
Если предусловие не выполнено, выбрасывается исключение ArgumentException и операция прерывается с сообщением.

**Граничные примеры**

**Валидный вход:**

1. **Вход:**
   * Фильтр по дедлайну: true
   * В коллекции есть задачи с дедлайном на сегодня или ранее
2. **Ожидаемый результат:** Возвращается коллекция просроченных и сегодняшних задач
3. **Индикаторы:** Pre — зелёный, Post — зелёный

**Невалидный вход:**

1. **Вход:**
   * Фильтр по дедлайну: false
   * Фильтр по приоритету: false
2. **Ожидаемое поведение:** Операция не выполняется, выводится сообщение об ошибке "Должен быть выбран хотя бы один критерий фильтрации"
3. **Индикаторы:** Pre — красный, Post — красный

# **Реализация приложения "Планировщик задач"**

Приложение построено по паттерну MVVM (Model-View-ViewModel) с четким разделением на модель (бизнес-логика), представление (интерфейс) и модель представления (связь между бизнес-логикой и интерфейсом).

**Структура приложения:**

**1. ViewModelBase (ViewModel) — базовый класс**

Предоставляет единый механизм уведомления для всех наследников: реализует интерфейс INotifyPropertyChanged для автоматического обновления UI при изменении свойств.

**2. IOperationViewModel (ViewModel) — интерфейс контракта**

Определяет единый вывод контракта для всех операций, обеспечивая полиморфизм и единообразие интерфейса.

**3. MainViewModel (ViewModel) — главная ViewModel приложения**

Координирует работу всех операций и управляет состоянием главного окна:

* Автоматически уведомляет UI об изменениях коллекции задач
* При изменении выбранной операции обновляет панель с деталями операции
* Инициализирует список доступных операций планировщика
* Управляет состоянием выбранной задачи для операций

**4. RelayCommand (ViewModel) — реализация команд**

Делает возможной привязку методов к командам в XAML. Автоматически проверяет, может ли быть выполнена команда на основе текущего состояния.

**5. BoolToColorConverter (View) — конвертер для индикаторов**

Преобразует boolean-значения состояний Pre/Post в цвета для визуальных индикаторов (зеленый/красный).

**6. TaskSchedulerService (Model) — сервис планировщика задач**

Содержит бизнес-логику операций с задачами и реализует проверку Pre и Post условий с использованием Guard и Debug.Assert.

**7. OperationViewModels (ViewModel) — семейство классов операций**

Отвечают за:

* Управление состоянием конкретной операции планировщика
* Валидацию входных данных (Pre-conditions)
* Проверку результатов выполнения (Post-conditions)
* Координацию между UI и бизнес-логикой (уведомление интерфейса, обновление индикаторов)

**8. AddTaskOperationViewModel (ViewModel) — операция добавления задачи**

Реализует логику добавления новых задач в планировщик с проверкой предусловий и постусловий.

**9. MoveTaskOperationViewModel (ViewModel) — операция переноса задачи**

Реализует логику изменения дедлайна существующей задачи с валидацией временных параметров.

**10. FilterTasksOperationViewModel (ViewModel) — операция фильтрации задач**

Реализует логику фильтрации задач по критериям (дедлайн, приоритет) с проверкой условий.

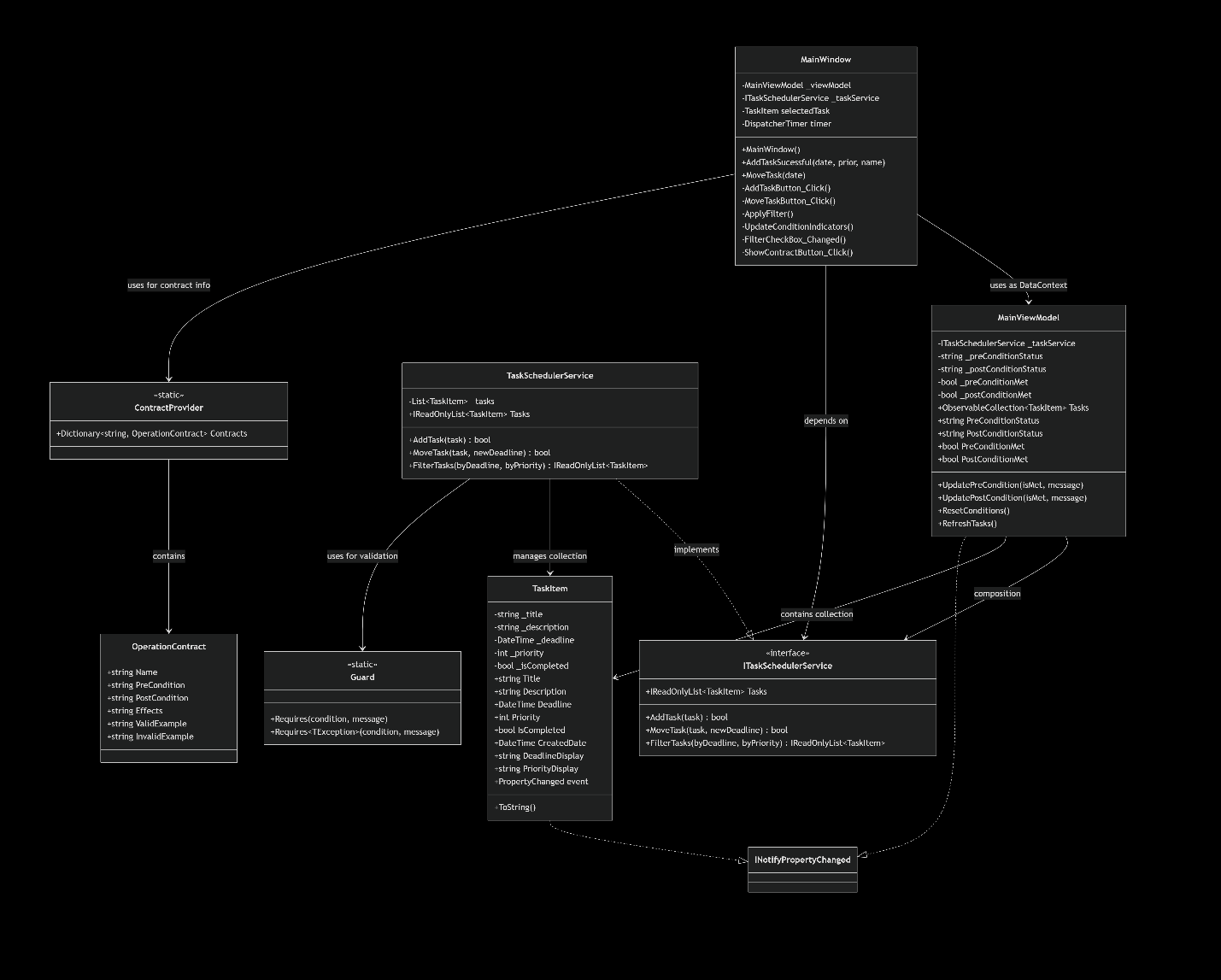
**Взаимодействие компонентов:**

**Модель (Model):** TaskSchedulerService содержит чистую бизнес-логику с строгими контрактами  
**Представление (View):** MainWindow.xaml обеспечивает визуальный интерфейс с индикаторами состояний  
**Модель представления (ViewModel):** Координирует данные между Model и View, управляет состоянием UI

Каждая операция имеет собственную ViewModel, что обеспечивает:

* Изолированное тестирование операций
* Четкое разделение ответственности
* Легкое добавление новых операций
* Единообразный интерфейс для всех операций

# **Диаграмма Классов**

[](https://imgbox.com/QVPKnS0o)

# **Тесты**

1. **Модуль GuardTests - Тесты утилиты проверки условий**

Что проверяет Guard

Валидация предусловий в коде

Выброс исключений при невыполнении условий

*using Lab1\_TaskScheduler.Utils;*

*using System;*

*using Xunit;*

*namespace Lab1\_TaskScheduler.Tests*

*{*

*public class GuardTests*

*{*

*[Fact]*

*public void Requires\_WithTrueCondition\_ShouldNotThrowException()*

*{*

*// Arrange*

*var condition = true;*

*// Act & Assert*

*var exception = Record.Exception(() => Guard.Requires(condition, "Сообщение"));*

*Assert.Null(exception);*

*}*

*[Fact]*

*public void Requires\_WithFalseCondition\_ShouldThrowArgumentException()*

*{*

*// Arrange*

*var condition = false;*

*var errorMessage = "Тестовое сообщение об ошибке";*

*// Act & Assert*

*var exception = Assert.Throws<ArgumentException>(() =>*

*Guard.Requires(condition, errorMessage));*

*Assert.Contains(errorMessage, exception.Message);*

*Assert.Contains("Pre-condition failed", exception.Message);*

*}*

*[Fact]*

*public void Requires\_WithCustomException\_ShouldThrowSpecifiedException()*

*{*

*// Arrange*

*var condition = false;*

*// Act & Assert*

*var exception = Assert.Throws<InvalidOperationException>(() =>*

*Guard.Requires<InvalidOperationException>(condition, "Кастомная ошибка"));*

*Assert.Contains("Кастомная ошибка", exception.Message);*

*}*

*}*

*}*

*Конкретные проверки*

*// Проверяет, что при ВЕРНОМ условии НЕТ исключения*

*Requires\_WithTrueCondition\_ShouldNotThrowException()*

*// Проверяет, что при НЕВЕРНОМ условии ВЫБРАСЫВАЕТСЯ ArgumentException*

*// с правильным сообщением об ошибке*

*Requires\_WithFalseCondition\_ShouldThrowArgumentException()*

*// Проверяет возможность выброса КАСТОМНОГО исключения*

*// (в данном случае InvalidOperationException)*

*Requires\_WithCustomException\_ShouldThrowSpecifiedException()*

**2.Модуль TaskItemTests - Тесты модели задачи**

Что проверяет TaskItem:

Корректность работы механизма уведомлений (INotifyPropertyChanged)

Правильность форматирования данных для отображения

Строковое представление объекта

*using Lab1\_TaskScheduler.Models;*

*using System;*

*using System.ComponentModel;*

*using Xunit;*

*namespace Lab1\_TaskScheduler.Tests*

*{*

*public class TaskItemTests*

*{*

*[Fact]*

*public void TaskItem\_PropertyChanged\_ShouldBeRaisedWhenPropertyChanges()*

*{*

*// Arrange*

*var task = new TaskItem();*

*var propertyChangedRaised = false;*

*string changedPropertyName = "";*

*task.PropertyChanged += (sender, e) =>*

*{*

*propertyChangedRaised = true;*

*changedPropertyName = e.PropertyName;*

*};*

*// Act*

*task.Title = "Новое название";*

*// Assert*

*Assert.True(propertyChangedRaised);*

*Assert.Equal(nameof(TaskItem.Title), changedPropertyName);*

*}*

*[Fact]*

*public void TaskItem\_DeadlineDisplay\_ShouldReturnFormattedString()*

*{*

*// Arrange*

*var deadline = new DateTime(2024, 1, 15, 14, 30, 0);*

*var task = new TaskItem { Deadline = deadline };*

*// Act*

*var display = task.DeadlineDisplay;*

*// Assert*

*Assert.Equal("15.01.2024 14:30", display);*

*}*

*[Fact]*

*public void TaskItem\_PriorityDisplay\_ShouldReturnFormattedString()*

*{*

*// Arrange*

*var task = new TaskItem { Priority = 4 };*

*// Act*

*var display = task.PriorityDisplay;*

*// Assert*

*Assert.Equal("Приоритет: 4", display);*

*}*

*[Fact]*

*public void TaskItem\_ToString\_ShouldReturnCorrectFormat()*

*{*

*// Arrange*

*var task = new TaskItem*

*{*

*Title = "Тестовая задача",*

*Priority = 3,*

*Deadline = new DateTime(2024, 1, 15, 10, 0, 0)*

*};*

*// Act*

*var result = task.ToString();*

*// Assert*

*Assert.Equal("Тестовая задача (Приоритет: 3, Дедлайн: 15.01.2024 10:00)", result);*

*}*

*}*

*}*

*Конкретные проверки:*

*// Проверяет, что при изменении свойства Title*

*// корректно вызывается событие PropertyChanged*

*TaskItem\_PropertyChanged\_ShouldBeRaisedWhenPropertyChanges()*

*// Проверяет форматирование даты в формате "dd.MM.yyyy HH:mm"*

*TaskItem\_DeadlineDisplay\_ShouldReturnFormattedString()*

*// Проверяет форматирование приоритета в виде "Приоритет: X"*

*TaskItem\_PriorityDisplay\_ShouldReturnFormattedString()*

*// Проверяет полное строковое представление задачи в формате:*

*// "Название (Приоритет: X, Дедлайн: дата)"*

*TaskItem\_ToString\_ShouldReturnCorrectFormat()*

**3. Модуль TaskSchedulerServiceTests - Тесты сервиса планировщика**

Что проверяет TaskSchedulerService:

Логику добавления задач

Валидацию бизнес-правил

Операции с задачами

*using Lab1\_TaskScheduler.Models;*

*using Lab1\_TaskScheduler.Services;*

*using System;*

*using Xunit;*

*namespace Lab1\_TaskScheduler.Tests*

*{*

*public class TaskSchedulerServiceTests*

*{*

*private readonly TaskSchedulerService \_service;*

*private readonly TaskItem \_validTask;*

*public TaskSchedulerServiceTests()*

*{*

*\_service = new TaskSchedulerService();*

*\_validTask = new TaskItem*

*{*

*Title = "Тестовая задача",*

*Description = "Описание тестовой задачи",*

*Deadline = DateTime.Now.AddDays(1),*

*Priority = 3*

*};*

*}*

*[Fact]*

*public void AddTask\_WithValidTask\_ShouldAddTaskToCollection()*

*{*

*// Arrange*

*var initialCount = \_service.Tasks.Count;*

*// Act*

*var result = \_service.AddTask(\_validTask);*

*// Assert*

*Assert.True(result);*

*Assert.Equal(initialCount + 1, \_service.Tasks.Count);*

*Assert.Contains(\_validTask, \_service.Tasks);*

*}*

*[Fact]*

*public void AddTask\_WithInvalidDeadline\_ShouldThrowArgumentException()*

*{*

*// Arrange*

*var invalidTask = new TaskItem*

*{*

*Title = "Невалидная задача",*

*Description = "Задача с прошедшим дедлайном",*

*Deadline = DateTime.Now.AddDays(-1),*

*Priority = 3*

*};*

*// Act & Assert*

*var exception = Assert.Throws<ArgumentException>(() => \_service.AddTask(invalidTask));*

*Assert.Contains("Дедлайн должен быть в будущем", exception.Message);*

*}*

*[Fact]*

*public void MoveTask\_WithValidParameters\_ShouldUpdateDeadline()*

*{*

*// Arrange*

*\_service.AddTask(\_validTask);*

*var newDeadline = DateTime.Now.AddDays(5);*

*// Act*

*var result = \_service.MoveTask(\_validTask, newDeadline);*

*// Assert*

*Assert.True(result);*

*Assert.Equal(newDeadline, \_validTask.Deadline);*

*}*

*}*

*}*

*Конкретные проверки:*

*// Проверяет, что валидная задача добавляется в коллекцию*

*// и размер коллекции увеличивается на 1*

*AddTask\_WithValidTask\_ShouldAddTaskToCollection()*

*// Проверяет, что задача с ПРОШЕДШИМ дедлайном*

*// НЕ может быть добавлена (выбрасывает исключение)*

*AddTask\_WithInvalidDeadline\_ShouldThrowArgumentException()*

*// Проверяет операцию переноса дедлайна задачи*

*// на новую дату*

*MoveTask\_WithValidParameters\_ShouldUpdateDeadline()*

# **Ссылка на git**

<https://github.com/LiraLays/Lab1TaskLisichka>