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

Иванов Максим, Полосков Андрей, Дубков Святослав

Проверила: Лисничук Арина Бахытжановна

Тверь 2025

Оглавление

[**ВВЕДЕНИЕ** 3](#_Toc209825791)

[**Операция №1: добавление** 3](#_Toc209825792)

[**Операция №2: перенос задач** 4](#_Toc209825793)

[**Операция №3: фильтрация по дедлайну** 5](#_Toc209825794)

[**Реализация приложения "Планировщик задач"** 6](#_Toc209825795)

# **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**