**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра ИС**

Курсовая РАБОТА

**по дисциплине «Объектно-ориентированное программирование»**

Тема: Будильник событий

|  |  |  |
| --- | --- | --- |
| Студенты гр. 1374 |  | Хлебников А. Дюков Н. В. |
| Преподаватель |  | Егоров С.С. |

Санкт-Петербург

2024**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| Студенты Хлебников А., Дюков Н. В. | | |
| Группа 1374 | | |
| Тема работы: Будильник событий | | |
| Исходные данные:  Есть будильник, показывающий текущее время, который должен при включении режима слежения информировать пользователя о наступлении некоторых событий в течении периода их актуальности. Одновременно могут быть актуальны несколько событий. | | |
| Студенты |  | Хлебников А. Дюков Н. В. |
| Преподаватель |  | Егоров С.С. |

**Аннотация**

Есть будильник, показывающий текущее время, который должен при включении режима слежения информировать пользователя о наступлении некоторых событий в течении периода их актуальности. Одновременно могут быть актуальны несколько событий. Проект разработан на языке программирования C#, используется графический интерфейс, построенный с помощью WPF, а также языка XAML. Архитектура приложения разработана с учетом паттерна MVVM (Model-View-ViewModel), позволяющий разделить приложение на функциональные части.

**Summary**

There is an alarm clock showing the current time, which should inform the user about the occurrence of certain events during the period of their relevance when the tracking mode is turned on. Several events may be relevant at the same time. The project is developed in the C# programming language, using a graphical interface built using WPF, as well as the XAML language. The application architecture is designed taking into account the MVVM (Model-View-ViewModel) pattern, which allows you to divide the application into functional parts.

**содержание**

|  |  |  |
| --- | --- | --- |
|  | Введение | 5 |
| 1. | Постановка задачи | 6 |
| 1.1. | Описание предметной области | 6 |
| 1.2. | Синтаксический анализ задания | 7 |
| 2. | Диаграммы объектной модели | 8 |
| 2.1. | Модель «сущность-связь» предметной области | 8 |
| 2.2. | Диаграмма классов | 8 |
| 3. | Подсистема «Интерфейс» | 11 |
| 3.1. | Графическое представление интерфейсных окон | 11 |
| 3.1.1. | Основное окно | 11 |
| 4. | Подсистема «Модель» | 13 |
| 4.1. | Перечень событий, изменяющих состояние модели ПрО | 13 |
| 5. | Скриншоты контрольного примера работы реализованного приложения | 14 |
| 5.1. | Описание контрольного примера | 14 |
| 5.2. | Инструкция пользователя работы с программой | 15 |
| 6. | Выводы по курсовой работе | 18 |
|  | Список использованных источников | 19 |

**введение**

В данной курсовой работе рассматривается разработка программного обеспечения, представляющего собой будильник, способный отслеживать текущие события и уведомлять пользователя о наступлении и окончании этих событий в заданные временные интервалы. Система состоит из будильника, событий и режима слежения. Параметры системы задаются во внешнем файле, после чего формируется случайная последовательность событий. Цель работы - разработать гибкую и масштабируемую систему на основе принципов объектно-ориентированного программирования.

**1. Постановка задачи**

* 1. **Описание предметной области**

**Цель системы**:

Создать будильник, который отслеживает текущие события и уведомляет пользователя о наступлении и окончании этих событий в заданные интервалы времени.

**Основные компоненты:**

1. **Будильник:**
   * Показывает текущее время.
   * Включает режим слежения за событиями.
2. **События:**
   * Имеют текстовое название.
   * Имеют интервал времени актуальности (период, в который событие считается активным).
3. **Режим слежения:**
   * Загружает файл
   * Генерирует последовательность событий

**Параметры системы:**

* Задано *K*=5 типов событий.
* Параметры событий хранятся во внешнем файле.
* В начальном состоянии, после загрузки файла, создается случайная последовательность из *N*=20 событий.
  + Тип каждого события выбирается случайным образом из интервала [1, K].
  + Интервал между событиями также выбирается случайным образом в диапазоне [Т1, Т2], где Т1=2Т1=2 секунды, а Т2=5Т2=5 секунд.

**1.2. Синтаксический анализ задания:**

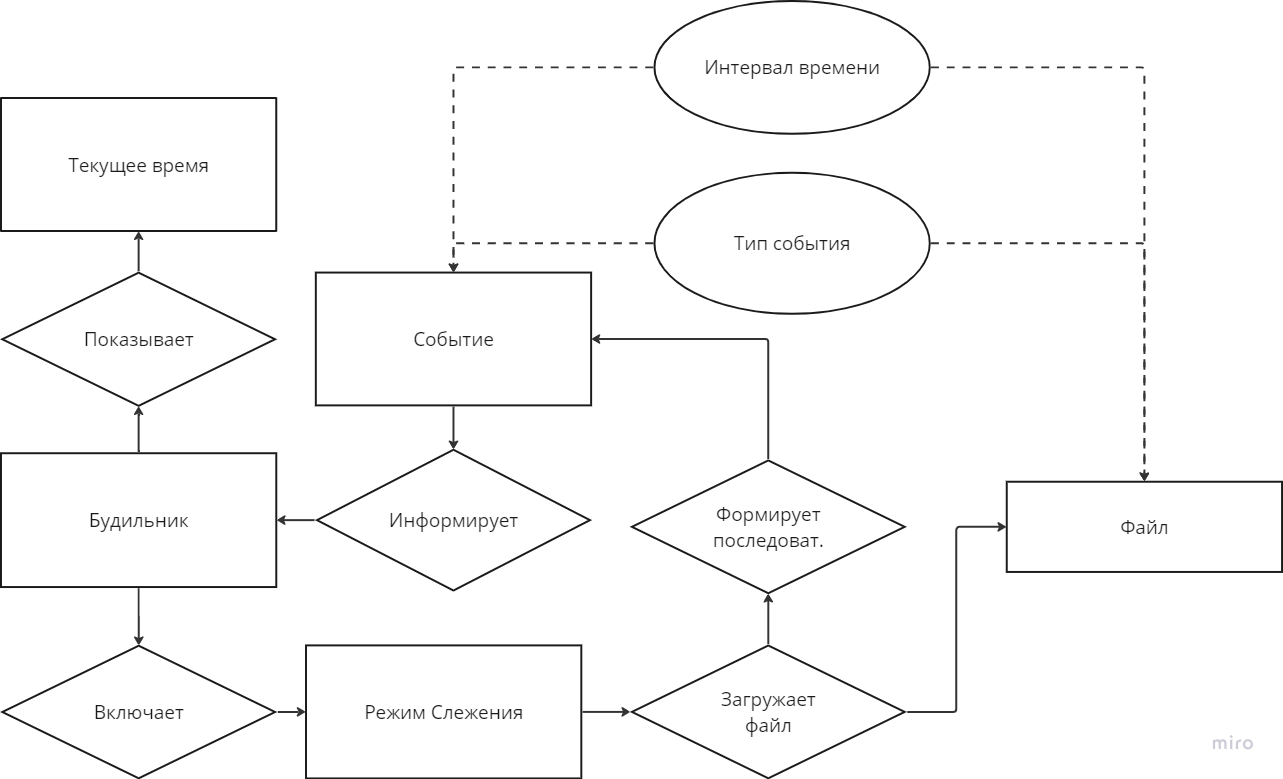
Есть будильник, показывающий текущее время, который должен при включении режима слежения информировать пользователя о наступлении некоторых событий в течении периода их актуальности. Одновременно могут быть актуальны несколько событий. Задано K (5) типов событий, каждый из которых характеризуется текстовым названием и интервалом времени его актуальности. Указанные параметры должны задаваться в виде внешнего файла. В начальном состоянии после загрузки этого файла должна формироваться случайная последовательность из N (20) событий, тип и интервал между предыдущим выбирается случайно по равномерному распределению на интервалах [1,K] и [Т1,T2] (2-5 сек.) соответственно. Для первого события интервал — это время от момента запуска отслеживания событий будильником, этому моменту приписываются нулевой отсчет. Поскольку изначально заданы только интервалы, в момент пуска для сформированной последовательности должен быть произведен расчет истинного времени наступления и окончания периода актуальности событий. По этим временам будильник в реальном времени должен информировать об их наступлении.

Таблица синтаксического анализа

|  |  |
| --- | --- |
| **Лексема** | **Элемент объектной модели** |
| Будильник | Класс |
| Показывающий | Метод будильника |
| Текущее время | Класс |
| Включение режима | Метод будильника |
| Режим слежения | Класс |
| Информировать | Метод будильника |
| Событие | Класс |
| Интервал времени | Характеристика события |
| Тип события | Характеристика события |
| Внешний файл | - |
| Загрузка файла | Метод режима слежения |
| Формирование случайной последовательности | Метод режима слежения |

**2. Диаграммы объектной модели**

**2.1. Модель «сущность-связь» предметной области**

****

**2.2. Диаграмма классов**

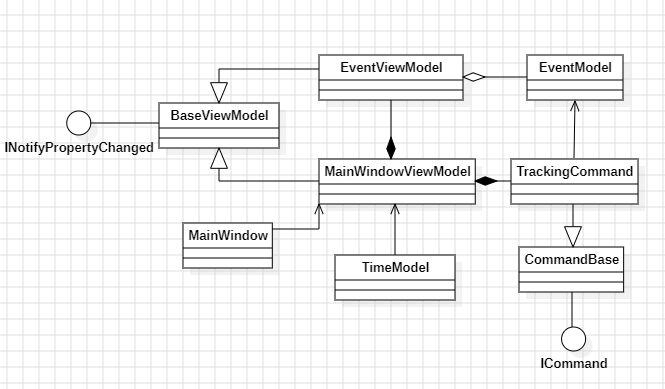
****

Таблица соответствия.

|  |  |  |
| --- | --- | --- |
| **Предметная область** | **Модель «сущность-связь»** | **Диаграмма классов** |
| Будильник | Будильник | MainWindowViewModel |
| Текущее время | Текущее время | TimeModel |
| Режим слежения | Режим слежения | TrackingCommand |
| Событие | Событие | EventModel |
| Файл | Файл | - |

Таблица отношений между классами

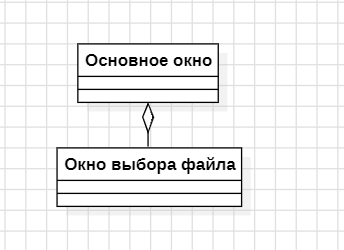
|  |  |  |
| --- | --- | --- |
| **Класс с исходящей стрелкой** | **Класс со входящей стрелкой** | **Вид отношения** |
| MainWindow | MainWindowViewModel | Ассоциация |
| MainWindowViewModel | BaseViewModel | Наследование |
| TimeModel | MainWindowViewModel | Ассоциация |
| TrackingCommand | MainWindowViewModel | Композиция |
| TrackingCommand | EventModel | Ассоциация |
| TrackingCommand | CommandBase | Наследование |
| EventModel | EventViewModel | Агрегация |
| CommandBase | ICommand | Реализация |
| EventViewModel | BaseViewModel | Наследование |
| EventViewModel | MainWindowViewModel | Композиция |
| BaseViewModel | INotifyPropertyChanged | Реализация |

Таблица атрибутов классов подсистемы "Модель"

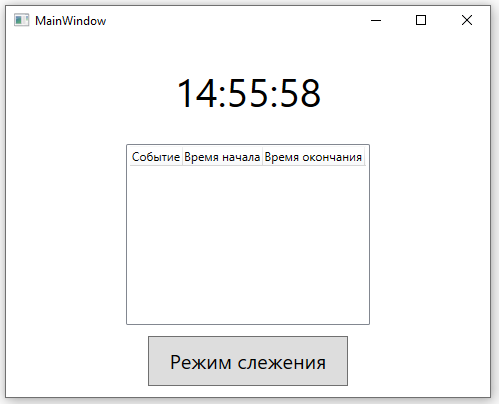
|  |  |  |
| --- | --- | --- |
| **Наименование**  **атрибута** | **Идентификатор**  **: тип** | **Назначение** |
| **Класс: MainWindowViewModel (Будильник)** | | |
| Список событий | \_eventList: ObservableCollection<EventViewModel> | Список, хранящий в себе события |
| Отображение списка событий | EventListView:  ICollectionView | Список, использующийся в визуальном отображении |
| Текущее время | \_time: TimeModel | Переменная, хранящая в себе текущее время |
| Привязка кнопки | Tracking: ICommand | Привязывает кнопку к функции |
| Визуальное отображение времени | Time: TimeModel | Визуально отображает текущее время |
| **Класс: TimeModel (Текущее время)** | | |
| Текущее время | DateTime: \_currentTime | Текущее время |
| **Класс: TrackingCommand (Режим слежения)** | | |
| Обращение к ViewModel | \_mainWindowViewModel:  MainWindowViewModel | Используется для связи кнопки и ViewModel |
| Путь к файлу | \_fileName: string | Хранит путь к файлу |
| Список типов событий | \_eventTypes: List<EventModel> | Список, хранящий в себе типы событий |
| **Класс: EventModel (Событие)** | | |
| Название события | EventName: string | Название события |
| Время начала | StartTime: string | Время начала события |
| Промежуток времени | Duration: TimeSpan | Промежуток времени существования события |
| Время окончания | StopTime: string | Время окончания события |
| Индикатор | IsRunning: bool | Индикатор наступления или окончания события |

**3. Подсистема «Интерфейс»**

**3.1 Графическое представление интерфейсных окон**

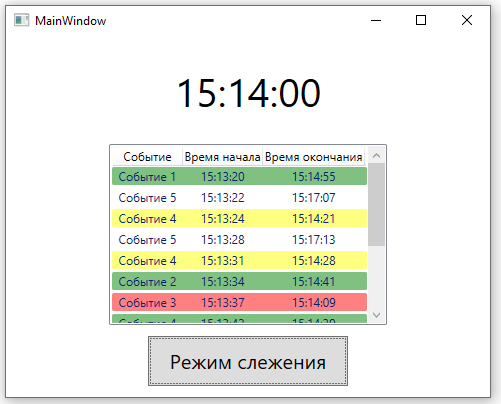
****

**3.1.1 Основное окно**

****

Система имеет только одно окно, в начальном состоянии, отображающее текущее время, а также кнопку «Режим слежения».

**3.1.2 Окно отображения состояния объектов ПрО**

****

**4. ПОДСИСТЕМА «МОДЕЛЬ»**

**4.1. Перечень событий, изменяющих состояние модели ПрО**

|  |  |  |
| --- | --- | --- |
| **Событие** | **Условие** | **Описание** |
| Начальное | - | Начальное состояние |
| Режим «Слежение» | Нажата кнопка | Переводит программу в режим слежение |
| Цвет события сменился на «Зеленый» | До конца события осталась 1 минута | Через 1 минуту событие будет окончено |
| Цвет события сменился на «Желтый» | До конца события осталась 30 секунд | Через 30 секунд событие будет окончено |
| Цвет события сменился на «Красный» | До конца события осталась 10 секунд | Через 10 секунд событие будет окончено |

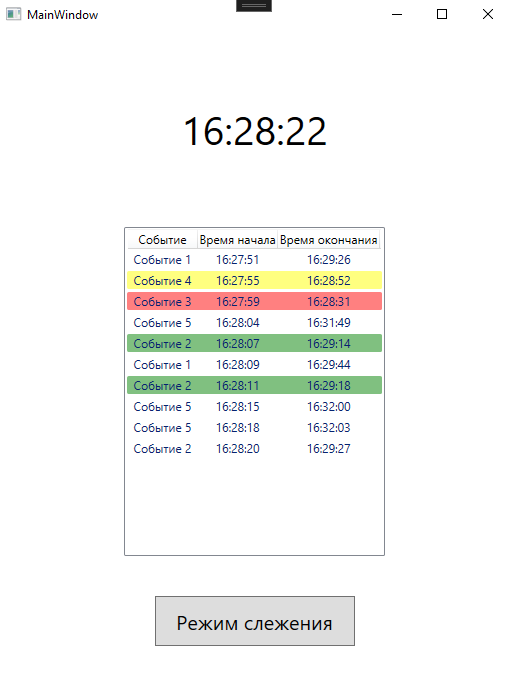
**5. Скриншоты контрольного примера работы реализованного приложения**

**5.1. Описание контрольного примера**

В данном примере будет использоваться файл с описаниями типов событий следующего вида:

### 

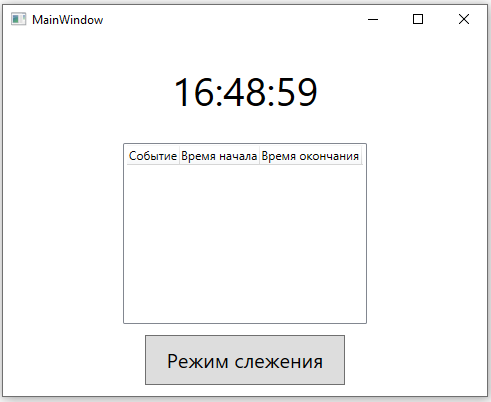
В данном случае сгенерированы вот такие события:



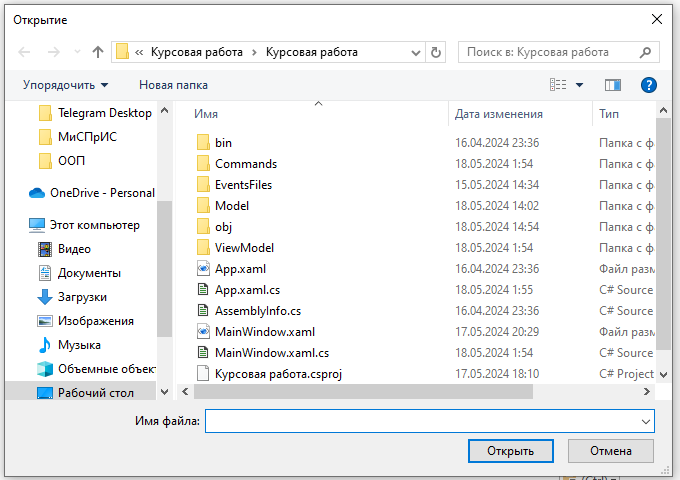
Как можно заметить, события имеют разный цвет, в зависимости от оставшегося количества времени до окончания этого события.

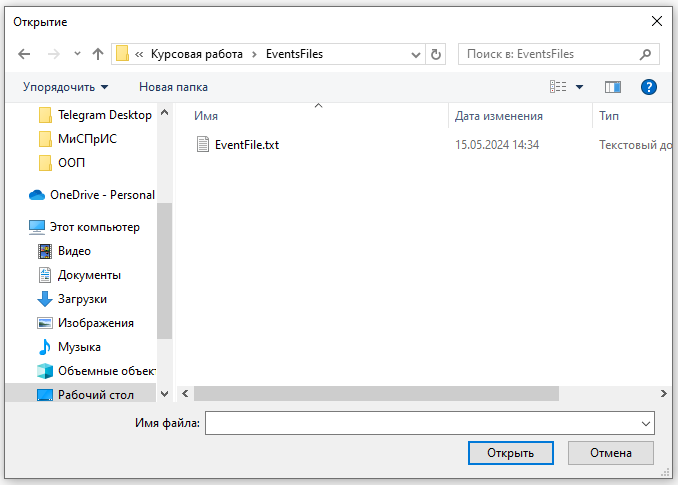
**5.2. Инструкция пользователя работы с программой**

При запуске программы открывается основное окно:

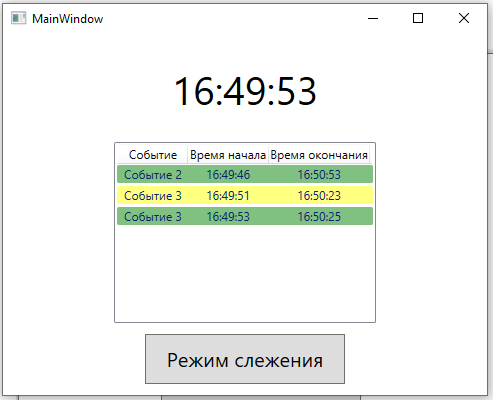


При нажатии кнопки «Режим слежения» открывается окно выбора файла:

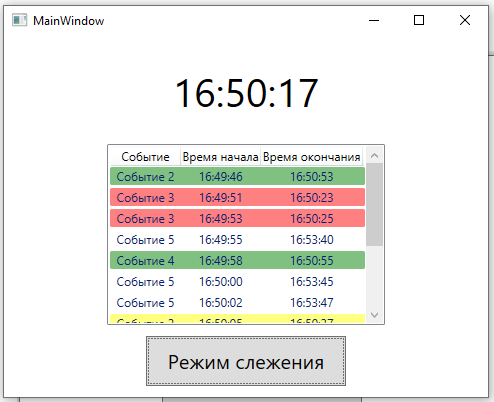


В папке «EventFiles» есть файл который можно использовать, но так же можно выбрать любой .txt файл имеющий вид «Название, Время»: 

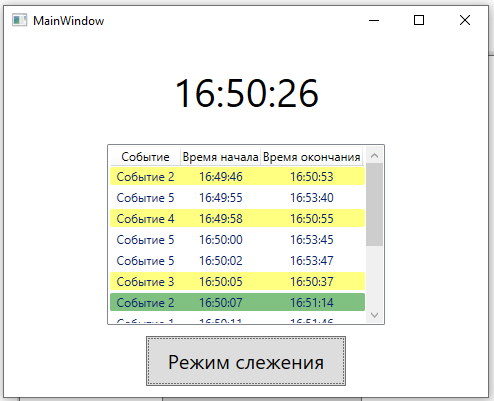
При выборе файла программа начинает генерировать события, сгенерированные события, у которых наступило время начала, отображаются в таблице:



При приближении времени окончания события, оно изменит свой цвет:



При наступлении времени окончания события, оно будет удалено с таблицы:



Для закрытия программы, нажмите X в правом верхнем углу.

**6. Выводы по курсовой работе**

В результате выполнения курсовой работы была разработана система **«Будильник с событиями»** с использованием методик объектно-ориентированного программирования, а также языка программирования C#. Система работает в точности с заявленными, заданием, условиями, а также протестированная на контрольном примере.

**список использованных источников**

1. Интернет-ресурс // METANIT.COM. URL: <https://metanit.com/sharp/tutorial/> (дата обращения: 11.05.2024)
2. Интернет-ресурс // METANIT.COM. URL: <https://metanit.com/sharp/wpf/> (дата обращения: 17.05.2024)
3. Марк Дж. Прайс «C# 10 и .NET 6. Современная кросс-платформенная разработка»
4. Ральф Джонсон «Паттерны объектно-ориентированного проектирования»