МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра «Информационно-вычислительные системы»

Курсовая работа
по дисциплине «Системное программное обеспечение»
на тему «Системный будильник»

ПГУ 1.090501.06

Специальность 09.05.01 «Применение и эксплуатация автоматизированных систем специального назначения»

Специализация №12 «Автоматизированные системы обработки информации и управления специального назначения»

Выполнил: студент гр. 18ВО1
_____Кравчук М.В.
Руководитель: к.т.н. доцент кафедры ИВС
_____Убиенных Г.В.

Работа защищена с оценкой_____
Преподаватели _____
_____
_____
Дата защиты _____

2021

Реферат

Пояснительная записка содержит 60 листов, 10 рисунков, 7 таблиц и 2 приложения.

Объектом исследования является системный будильник.

Цель работы — разработать приложение демонстрирующее работу системного будильника.

В результате проделанной работы разработано приложение-системный будильник.

При разработке приложения использовалась среда визуального программирования Visual Studio 2019 Community.

Приложение спроектировано с помощью CASE-средства StarUML.

					ПГУ 1.090501.06			
Изм.	Лист	№ докум.	Подп.	Дата				
Разраб.		Кравчук М.В.			Системный будильник	Лит.	Лист	Листов
Пров.		Убиенных Г.Ф.					3	60
						Гр. 18ВО1		
Н. контр.								
Утв.								

Содержание

Введение

Будильник в современном мире – незаменимая часть жизни современного человека.

Данная программа представляет собой системный будильник, который позволит даже неподготовленному пользователю ознакомиться с возможностями работы будильника, не затрачивая много времени и усилий./1/

Унифицированный язык моделирования (UML) является стандартным инструментом для создания «чертежей» программного обеспечения./2/ С помощью UML можно визуализировать, специфицировать, конструировать и документировать артефакты программных систем./3/

UML пригоден для моделирования любых систем: от информационных систем масштаба предприятия до распределенных Web-приложений и даже встроенных систем реального времени./4/ Это очень выразительный язык, позволяющий рассмотреть систему со всех точек зрения, имеющих отношение к ее разработке и последующему развертыванию./5/

1 Техническое задание

1.1 Основание для разработки

Программа разрабатывается на основании задания на курсовое проектирование, выданное к.т.н., доцентом кафедры ИВС Убиенных Г.Ф. и утвержденного зав. кафедрой ИВС к.т.н., доцентом Бобрышевой Г.В.

1.2 Назначение разработки

Разрабатываемое приложение должно быть предназначено для работы системного будильника со следующими функциями: добавление, редактирование, удаление напоминаний, вывод напоминания о скором наступлении события с соответствующим звуком будильника, вывод напоминания о завершении события.

1.3 Требования к программе

1.3.1 Требования к функциональным характеристикам

Разрабатываемая программа должна выполнять следующие функции:

- добавление, редактирование, удаление напоминаний;
- вывод напоминания о скором наступлении события;
- вывод напоминания о завершении события;
- воспроизведение звука будильника при напоминании.

Входные данные: информация о напоминании, дата события(день, месяц, год), время события(часы, минуты, секунды), время за которое требуется оповестить пользователя.

Выходные данные: напоминания о предстоящем событии.

1.3.2 Требования к надежности

Программа должна работать без перебоев, выдавать сообщения об ошибках при неверно заданных исходных данных, поддерживать диалоговый режим в рамках предоставляемых пользователю возможностей.

1.3.3 Требования к составу и параметрам технических средств

Программа должна быть предназначена для использования на локальном компьютере. Локальный компьютер должен иметь следующие технические характеристики:

- тактовая частота процессора - 2 ГГц;
- оперативная память - 512 Мбайт;
- свободное место на жестком диске - 30 Мбайт.

Локальный компьютер или сервер в сети должны иметь следующие технические характеристики:

- тактовая частота процессора - 2 ГГц;
- оперативная память - 1024 Мбайт;
- свободное место на жестком диске - 300 Мбайт.

1.3.4 Требования к информационной и программной совместимости

Программа должна быть предназначена для использования на локальном компьютере. На компьютере должна быть установлена программа SystemAlarmClock.exe.

Разработанное приложение должно работать под управлением операционной системы Microsoft Windows 7 и выше.

1.4 Требования к программной документации

Программная документация к разрабатываемому приложению должна содержать следующие разделы:

- техническое задание;
- проектирование объектно-ориентированной модели;
- проектирование приложения;
- описание программы;
- программа и методика испытаний;
- описание применения;
- текст программы.

1.5 Стадии и этапы разработки

Стадии и этапы разработки проекта приведены в таблице 1.

Таблица 1 — Стадии и этапы разработки проекта

N п/п	Стадии	Этапы	Срок выполнения	Исполнитель
1	2	3	4	5
1	Расчетная часть	Объектно-ориентированный анализ предметной области		
		Объектно-ориентированное проектирование системы		

Продолжение таблицы 1

N п/п	Стадии	Этапы	Срок выполнения	Исполнитель
1	2	3	4	5
1	Расчетная часть	Реализация и тестирование программы		
2	Графическая часть не предусмотрена			
3	Экспериментальная часть	Первичная версия системы		
		Вторая версия системы		

1.6 Порядок контроля и приемки

Для проверки выполнения программы всех ее функций в соответствии с требованиями технического задания необходимо разработать тестовые примеры. Следует протестировать действия программы на возможные реакции пользователя: запуск программы, нажатие кнопок. Приложение соответствует требованиям технического задания, если оно выполняет все заданные функции при различных наборах входных данных из тестовых примеров.

Приемка программы должна осуществляться при выполнении всех ее функций, а также при наличии полной документации на программу (пояснительная записка с приложениями: текстом программы и результатами тестовых примеров).

2 Проектирование приложения

2.1 Разработка модели предметной области

Системный будильник функционирует не только для того, чтобы будить людей по утрам. С развитием технологий возможности системного будильника, стало возможно значительно расширить, так же как и сферы применения будильника. В современном мире системный будильник предназначен для оказания помощи в деловой и личной жизни (напоминания о грядущих событиях до самого события).

Для возможности просмотра функций будильника разработана программная система «Системный будильник».

2.2 Разработка диаграммы вариантов использования

Диаграмма вариантов использования, разработанная с помощью CASE-средства StarUML представлена на рисунке 1.

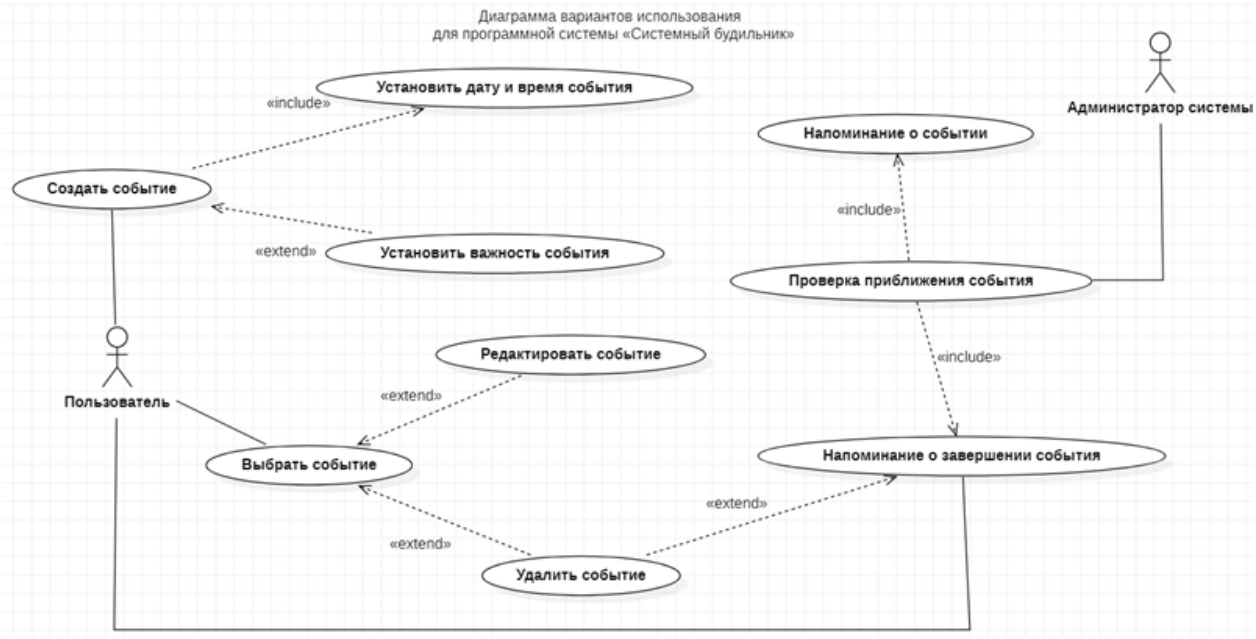

Рисунок 1 — Диаграмма вариантов использования для программной системы «Системный будильник»

Согласно диаграмме вариантов использования для программной системы «Системный будильник», изображенной на рисунке 1, действующее лицо — это пользователь, стрелки — связи между действующим лицом и вариантами использования, а варианты использования — это функции, выполняемые системой, а в частности: «Создать событие», «Удалить событие», «Редактировать событие», «Установка важности события», «Установить дату и время события», «Проверка приближения события», «Напоминание о событии» и «Напоминание о завершении события».

Описание вариантов использования:

Название: Создать событие.

Цель: Создать запись в базу данных о событии, с целью дальнейшего напоминания о нем.

Основные исполнители: Пользователь.

Основной сценарий:

1. Пользователь нажимает на кнопку «Добавить событие», расположенную на главной форме приложения;

2. Система открывает окно для создания события;

3. Выполнить вариант события «Установить дату и время события»;

4. Пользователь вводит информацию о событии в соответствующее текстовое поле и нажимает на кнопку, предназначенную для сохранения информации о событии.

5. Выполнить вариант события «Установить важность события»;

6. Система записывает событие в список событий и закрывает форму для создания события.

Название: «Установить дату и время события».

Цель: Внесение информации о дате и времени события, о котором требуется напомнить.

Основные исполнители: Пользователь.

Основной сценарий:

1. Пользователь вводит время и дату в поле, позволяющее выбор даты и времени;

2. Система записывает и проверяет введенную пользователем дату на корректность ввода.

Название: Установить важность события.

Цель: Установка времени, за которое требуется напомнить пользователю о предстоящем событии.

Основные исполнители: Пользователь.

Основной сценарий:

1. Система открывает окно для установки важности события;

2. Пользователь вводит время, за которое нужно напомнить о предстоящем событии.

Название: Выбрать событие.

Цель: Выбрать событие с целью совершения с ним дальнейших действий.

Основные исполнители: Пользователь.

Начальное состояние: Выполнен ВИ «Создать событие».

Основной сценарий:

1. Пользователь выбирает среди списка событий то, которое необходимо изменить или удалить и выделяет его;

2. Система отображает кнопки для удаления и редактирования информации о событии;

3. Если пользователю напоминание о событии больше не требуется, то выполнить вариант использования «Удалить событие»;

4. Если информация о событии устарела, то выполнить вариант использования «Редактировать событие»;

Название: Редактировать событие.

Цель: Редактирование информации о событии с целью поддержки её актуальности.

Основные исполнители: Пользователь.

Основной сценарий:

1. Пользователь нажимает кнопку, предназначенную для редактирования информации о событии;

2. Система открывает окно для редактирования события;

3. Пользователь вводит новые актуализированные данные о событии;

4. Пользователь нажимает кнопку, предназначенную для сохранения актуальной информации о событии;

5. Система записывает актуализированную информацию о предстоящем событии в список событий.

Название: Удалить событие.

Цель: Удаление события из списка событий.

Основные исполнители: Пользователь.

Основной сценарий:

1. Пользователь нажимает кнопку, предназначенную для удаления информации о событии;

2. Система выводит окно для подтверждения удаления;

3. Пользователь подтверждает удаление нажатием на кнопку «Ok»;

4. Система удаляет событие из списка событий.

Название: Проверка приближения события.

Цель: Отслеживание наступления всех событий, занесенных в список событий.

Основные исполнители: Администратор системы.

Начальное состояние: Выполнены ВИ «Создать событие».

Основной сценарий:

Каждую минуту выполнять:

1. Система читает в каждой записи списка событий поле даты и времени события;

2. Система читает в каждой записи списка событий поле времени, за которое нужно напомнить о предстоящем событии;

3. Система производит вычитание системного времени из времени напоминания;

4. Полученный результат система сравнивает с временем за которое нужно напомнить о предстоящем событии;

5. Если результат сравнения истинный, то выполнить вариант использования «Напоминание о событии»;

6. Система производит сравнение времени события с системным временем;

7. Если результат сравнения истинный, то выполнить вариант использования «Напоминание о завершении события»;

Название: Напоминание о событии.

Цель: Оказание помощи в деловой и личной жизни по средствам уведомления о предстоящем событии.

Основные исполнители: Администратор системы.

Основной сценарий:

1. Система выводит окно для сообщения пользователю о скором наступлении события;

2. Система выводит в данном окне информацию о событии и время события;

3. Система воспроизводит мелодию для обозначения скорого наступления события.

Название: Напоминание о завершении события.

Цель: Оповещение о завершении события с целью удаления события из

списка событий или редактирования информации о нём.

Основные исполнители: Администратор системы и Пользователь.

Основной сценарий:

1. Система выводит окно для сообщения пользователю о завершении события;

2. Система выводит в данном окне информацию о событии и о времени события, а также две кнопки: «Удалить» и «Отмена»;

3. Если пользователю напоминание о событии больше не требуется, то выполнить вариант использования «Удалить событие»;

4. Иначе, пользователь нажимает на кнопку «Отмена»;

5. Система закрывает окно для сообщения пользователю о завершении события.

2.3 Разработка диаграммы классов

Описание использующихся классов в варианте использования «Создать событие» представлено в таблице 2. На рисунке 2 представлена диаграмма использующихся классов в варианте использования «Создать событие» в программе «Системный будильник», а также отображены организованные между ними связи.

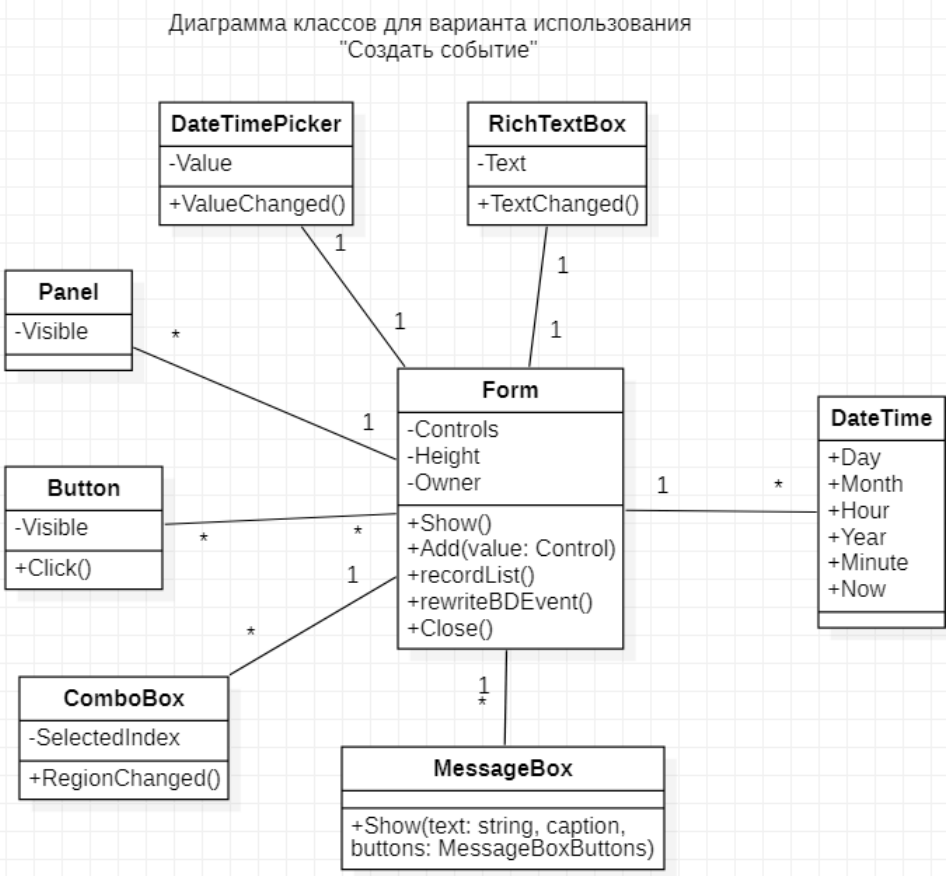

Рисунок 2 — Диаграмма классов варианта использования «Создать событие» программной системы «Системный будильник»

В результате анализа задания и диаграммы варианта использования «Создать событие» приложения «Системный будильник» используются следующие стандартные классы, которые связаны с классом Form ассоциативной связью: Button, Panel, DateTimePicker, RichTextBox, DateTime, ComboBox, MessageBox.

Описание классов приведено в таблице 2.

Таблица 2 — Описание использующихся в варианте использования «Создать событие» классов

Класс	Описание класса
Form	Форма для работы с приложением «Системный будильник»
Button	Используется для выполнения команд пользователя
Panel	Класс-контейнер для других компонентов
DateTimePicker	Используется для выбора даты и времени
RichTextBox	Класс, используемый как многострочный редактор
DateTime	Класс для работы с датой и временем
ComboBox	Используется для выбора времени до события
MessageBox	Используется для предупреждения пользователя

Класс Form (2 экземпляра класса) содержит поля:

1. Controls — управление элементами формы;
2. Height — длинна формы;
3. Owner — ссылка на экземпляр формы.

Класс Form содержит методы:

1. Show() - создание формы;
2. Add(in value:Control) – добавить элемент на форму;
3. recordList() – запись нового события;
4. rewriteBDEvent() – перезапись базы данных «События»;
5. Close() – закрытие формы.

Класс Button (множество экземпляров класса) содержит поле:

1. Visible – определяет видимость компонента во время выполнения.

Класс Button содержит метод:

1. Click() - нажатие на кнопку.

Класс Panel (множество экземпляров класса) содержит поле:

1. Visible – определяет видимость компонента во время выполнения.

Класс DateTimePicker (1 экземпляр класса) содержит поле:

1. Value – значение даты и времени.

Класс DateTimePicker содержит метод:

1. ValueChanged() – обработчик события изменения значения.

Класс RichTextBox (1 экземпляр класса) содержит поле:

1. Text – текст на элементе.

Класс RichTextBox содержит метод:

1. TextChanged() – обработчик события изменения текста.

Класс DateTime (множество экземпляров класса) содержит поля:

1. Year – год;
2. Month — месяц;
3. Day – день;
4. Hour – час;
5. Minute — минута.
6. Now – системное время.

Класс ComboBox (множество экземпляров класса) содержит поля:

1. SelectedIndex — номер выбранного элемента.

Класс ComboBox содержит метод:

1. RegionChanged() — обработчик события изменения значения.

Класс MessageBox (множество экземпляров класса) содержит метод:

1. Show(text:string, caption, buttons:MessageBoxButtons) — отображает окно сообщения, содержащее заданный текст, заголовок и кнопки.

Описание использующихся классов в варианте использования «Редактировать событие» представлено в таблице 3. На рисунке 3 представлена диаграмма использующихся классов в варианте использования «Редактировать событие» в программе «Системный будильник», а также отображены организованные между ними связи.

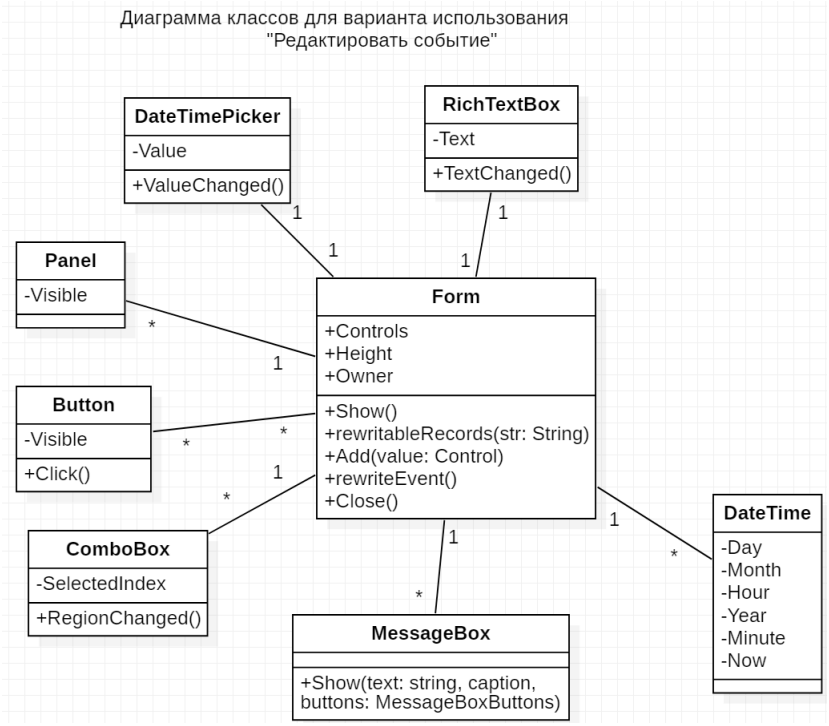

Рисунок 3 — Диаграмма классов варианта использования «Редактировать событие» программной системы «Системный будильник»

В результате анализа задания и диаграммы варианта использования «Редактировать событие» приложения «Системный будильник» используются следующие стандартные классы, которые связаны с классом Form ассоциативной связью: Button, Panel, DateTimePicker, RichTextBox, DateTime, ComboBox, MessageBox.

Описание классов приведено в таблице 3.

Таблица 3 — Описание использующихся в варианте использования «Редактировать событие» классов

Класс	Описание класса
Form	Форма для работы с приложением «Системный будильник»
Button	Используется для выполнения команд пользователя
Panel	Класс-контейнер для других компонентов
DateTimePicker	Используется для выбора даты и времени
RichTextBox	Класс, используемый как многострочный редактор
DateTime	Класс для работы с датой и временем
ComboBox	Используется для выбора времени до события
MessageBox	Используется для предупреждения пользователя

Класс Form (2 экземпляра класса) содержит поля:

1. Controls — управление элементами формы;
2. Height — длинна формы;
3. Owner — ссылка на экземпляр формы.

Класс Form содержит методы:

1. Show() - создание формы;
2. rewritableRecords() - отправляет данные для изменения;
3. Add(in value:Control) – добавить элемент на форму;
4. rewriteEvent() – редактирование события в списке событий;
5. Close() – закрытие формы.

Класс Button (множество экземпляров класса) содержит поле:

1. Visible – определяет видимость компонента во время выполнения.

Класс Button содержит метод:

1. Click() - нажатие на кнопку.

Класс Panel (множество экземпляров класса) содержит поле:

1. Visible – определяет видимость компонента во время выполнения.

Класс DateTimePicker (1 экземпляр класса) содержит поле:

1. Value – значение даты и времени.

Класс DateTimePicker содержит метод:

1. ValueChanged() – обработчик события изменения значения.

Класс RichTextBox (1 экземпляр класса) содержит поле:

1. Text – текст на элементе.

Класс RichTextBox содержит метод:

1. TextChanged() – обработчик события изменения текста.

Класс DateTime (множество экземпляров класса) содержит поля:

1. Year – год;
2. Month — месяц;
3. Day – день;
4. Hour – час;
5. Minute — минута.
6. Now – системное время.

Класс ComboBox (множество экземпляров класса) содержит поля:

1. SelectedIndex — номер выбранного элемента.

Класс ComboBox содержит метод:

1. RegionChanged() — обработчик события изменения значения.

Класс MessageBox (множество экземпляров класса) содержит метод:

1. Show(text:string, caption, buttons:MessageBoxButtons) — отображает окно сообщения, содержащее заданный текст, заголовок и кнопки.

Описание использующихся классов в варианте использования «Проверка приближения события» представлено в таблице 4. На рисунке 4 представлена диаграмма использующихся классов в варианте использования «Проверка приближения события» в программе «Системный будильник», а также отображены организованные между ними связи.

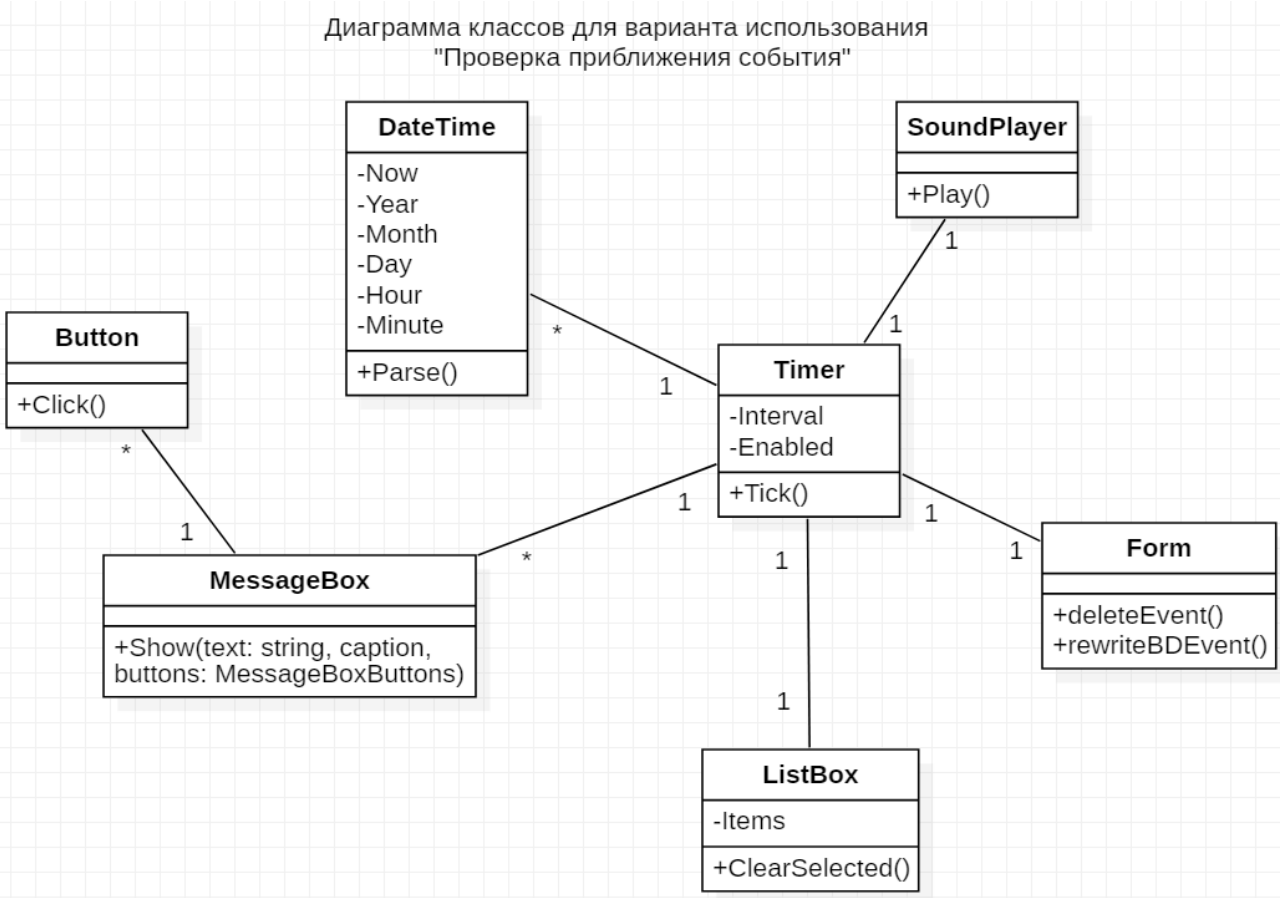

Рисунок 4 — Диаграмма классов варианта использования «Проверка приближения события» программной системы «Системный будильник»

В результате анализа задания и диаграммы варианта использования «Проверка приближения события» приложения «Системный будильник» используются следующие стандартные классы, которые связаны с классом Timer ассоциативной связью: Form, SoundPlayer, DateTime, ListBox, MessageBox. Так же с классом MessageBox ассоциативной связью связан класс Button.

Описание классов приведено в таблице 4.

Таблица 4 — Описание использующихся в варианте использования «Редактировать событие» классов

Класс	Описание класса
Timer	Используется для отслеживания приближения время события
Form	Форма для работы с приложением «Системный будильник»
Button	Используется для выполнения команд пользователя
DateTime	Класс для работы с датой и временем
MessageBox	Используется для оповещения пользователя
SoundPlayer	Используется для музыкального сопровождения напоминания
ListBox	Используется для отображения списка событий

Класс Timer (1 экземпляр класса) содержит поля:

1. Interval – интервал времени в миллисекундах, по истечении которого возникает событие Tick;

2. Enabled – признак активности таймера.

Класс Form содержит метод:

1. Tick() - по истечению времени Interval, производит проверку приближения события.

Класс Form (1 экземпляр класса) содержит методы:

1. deleteEvent() - удаление события их списка событий;

2. rewriteBDEvent() – перезапись базы данных «События».

Класс Button (1 экземпляр класса) содержит метод:

1. Click() - нажатие на кнопку.

Класс DateTime (множество экземпляров класса) содержит поля:

1. Year – год;

2. Month — месяц;

3. Day – день;

4. Hour – час;

5. Minute — минута.

6. Now – системное время.

Класс DateTime содержит метод:

1. Parse() - преобразует строковое представление даты и времени в его эквивалент DateTime.

Класс MessageBox (множество экземпляров класса) содержит метод:

1. Show(text:string, caption, buttons:MessageBoxButtons) — отображает окно сообщения, содержащее заданный текст, заголовок и кнопки.

Класс SoundPlayer (1 экземпляр класса) содержит метод:

1. Play() - воспроизвести музыкальное сопровождение напоминания.

Класс ListBox (1 экземпляр класса) содержит поле:

1. Items – элементы списка.

Класс ListBox содержит метод:

1. ClearSelected() – снимает выделение со всех строк.

2.4 Разработка диаграмм последовательности

Диаграмма последовательности для варианта использования «Создать событие» приведена на рисунке 5.

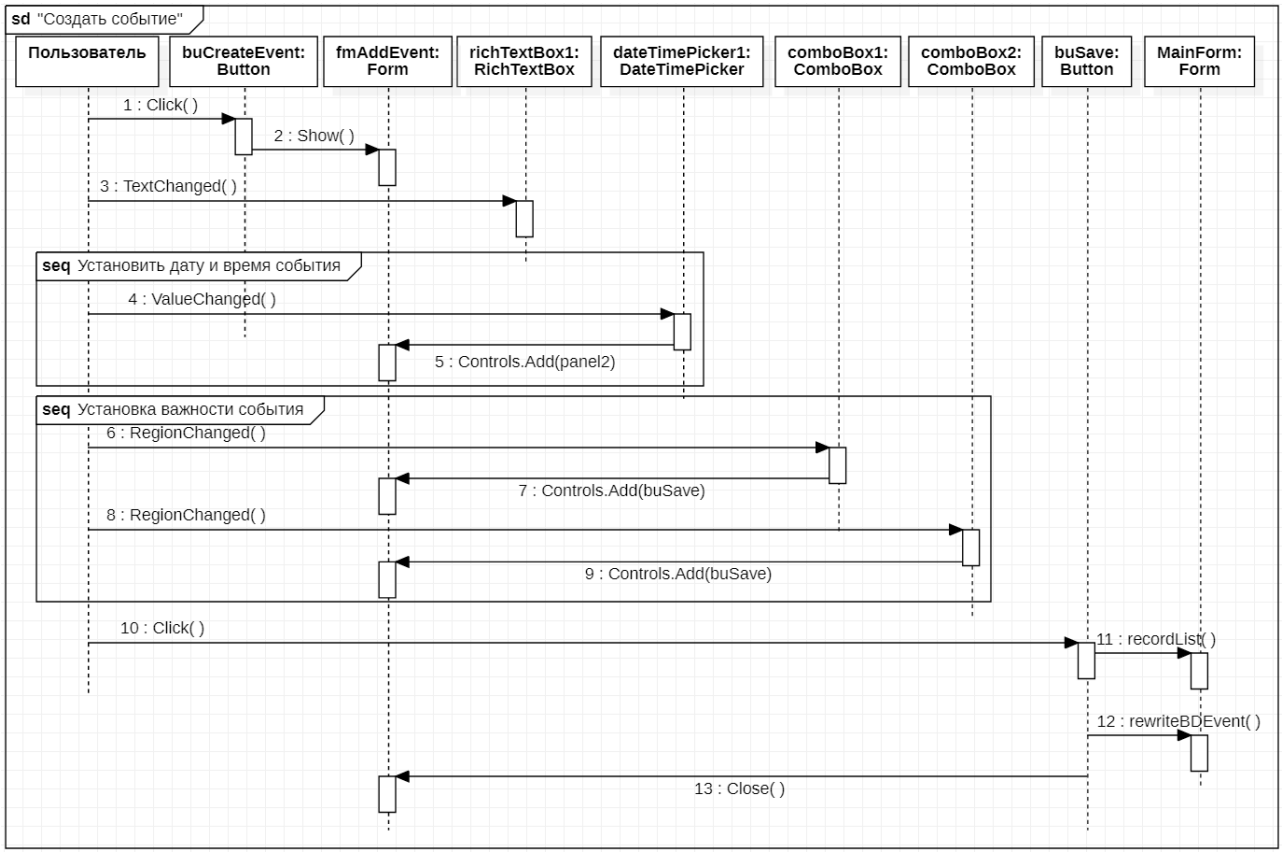

Рисунок 5 - Диаграмма последовательности для варианта использования «Создать событие»

В данной диаграмме изображена последовательность действий при создании события.

Основной сценарий:

1.[1] Пользователь нажимает на кнопку «Добавить событие», расположенную на главной форме приложения;

2.[2] Система открывает окно для создания события;

3.[3] Пользователь вводит информацию о событии в соответствующее текстовое поле;

4.[4-5] Выполнить вариант события «Установить дату и время события»;

5.[6-9] Выполнить вариант события «Установить важность события»;

6.[10] Пользователь нажимает на кнопку, предназначенную для сохранения информации о событии;

7.[11-13] Система записывает событие в список событий и закрывает форму для создания события.

Диаграмма последовательности для варианта использования «Редактировать событие» приведена на рисунке 6.

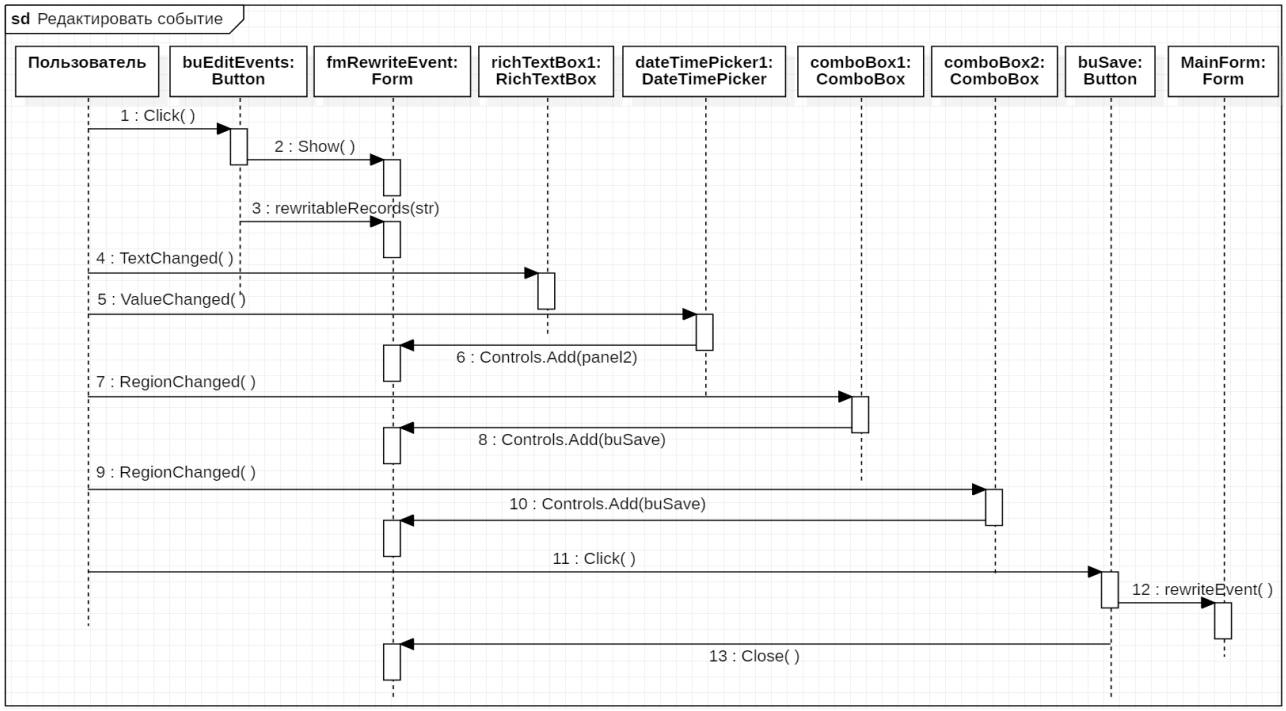

Рисунок 6 - Диаграмма последовательности для варианта использования «Редактировать событие»

В данной диаграмме изображена последовательность действий при редактировании события.

Основной сценарий:

1.[1] Пользователь нажимает кнопку, предназначенную для редактирования информации о событии;

2.[2-3] Система открывает окно для редактирования события, занося информацию о событии в соответствующие элементы;

3.[4-10] Пользователь вводит новые актуализированные данные о событии;

4.[11] Пользователь нажимает кнопку, предназначенную для сохранения актуальной информации о событии;

5.[12-13] Система записывает актуализированную информацию о предстоящем событии в список событий.

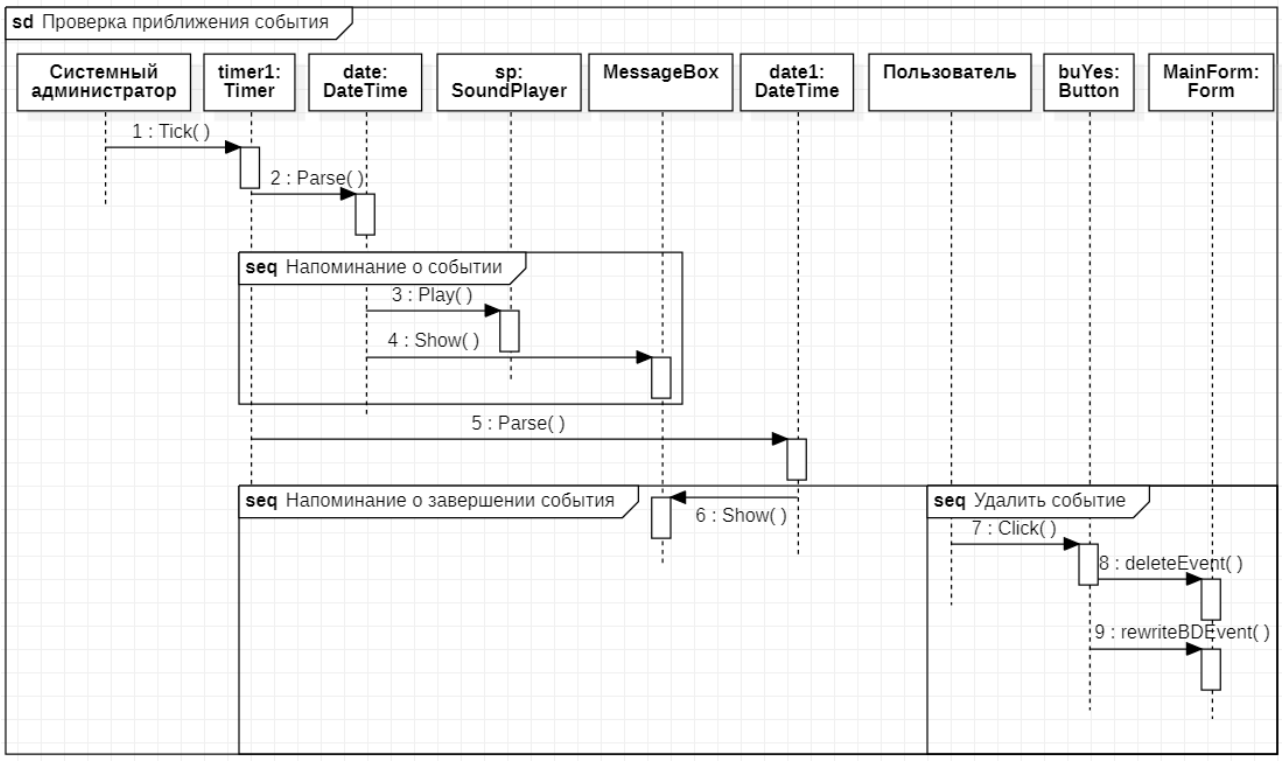

Рисунок 7 - Диаграмма последовательности для варианта использования «Проверка приближения события»

В данной диаграмме изображена последовательность действий при проверке приближения события.

Основной сценарий:

[1]Системный администратор каждую минуту запускает проверку приближения события, которая включает в себя, следующие действия:

1.[2 и 5] Система читает в каждой записи списка событий поле времени

события и напоминания о нём;

2. Система производит сравнение времени события с системным временем;

3.[3-4] Если результат сравнения истинный, то выполнить вариант использования «Напоминание о событии»;

4. Система производит сравнение времени напоминания с системным временем;

5.[6-9] Если результат сравнения истинный, то выполнить вариант использования «Напоминание о завершении события».

2.5 Разработка диаграммы состояний

Диаграмма состояний для объекта timer1 представлена на рисунке 8.

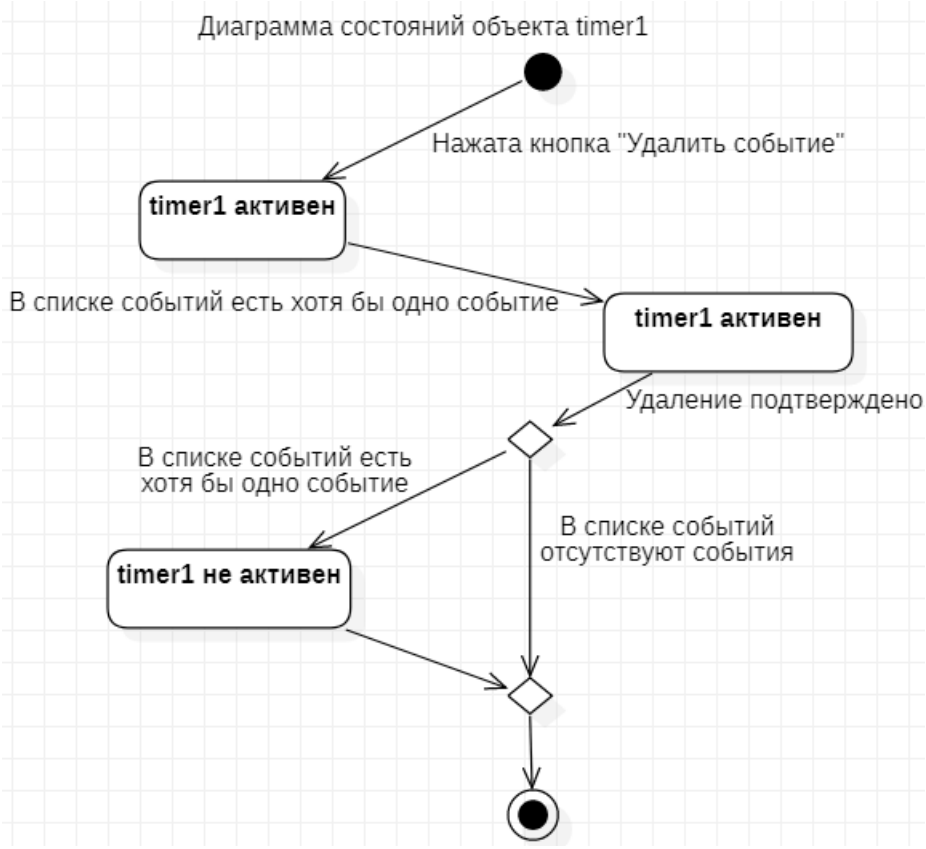

Рисунок 8 — Диаграмма состояний объекта timer1

Объект имеет 2 состояния: активен и неактивен

1) После запуска приложения timer1 активен.

2) После нажатия кнопки «Удалить событие», свойство enabled объекта timer не изменяется.

3) Завершение удаления осуществляется подтверждением процедуры удаления события. Свойство enabled объекта timer1 изменяется на false в случае если в списке событий отсутствует, хотя бы одно событие, после чего объект timer1 становится не активным.

2.6 Разработка диаграммы модулей

Диаграмма компонентов (модулей) — элемент языка моделирования UML, статическая структурная диаграмма, которая показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами.

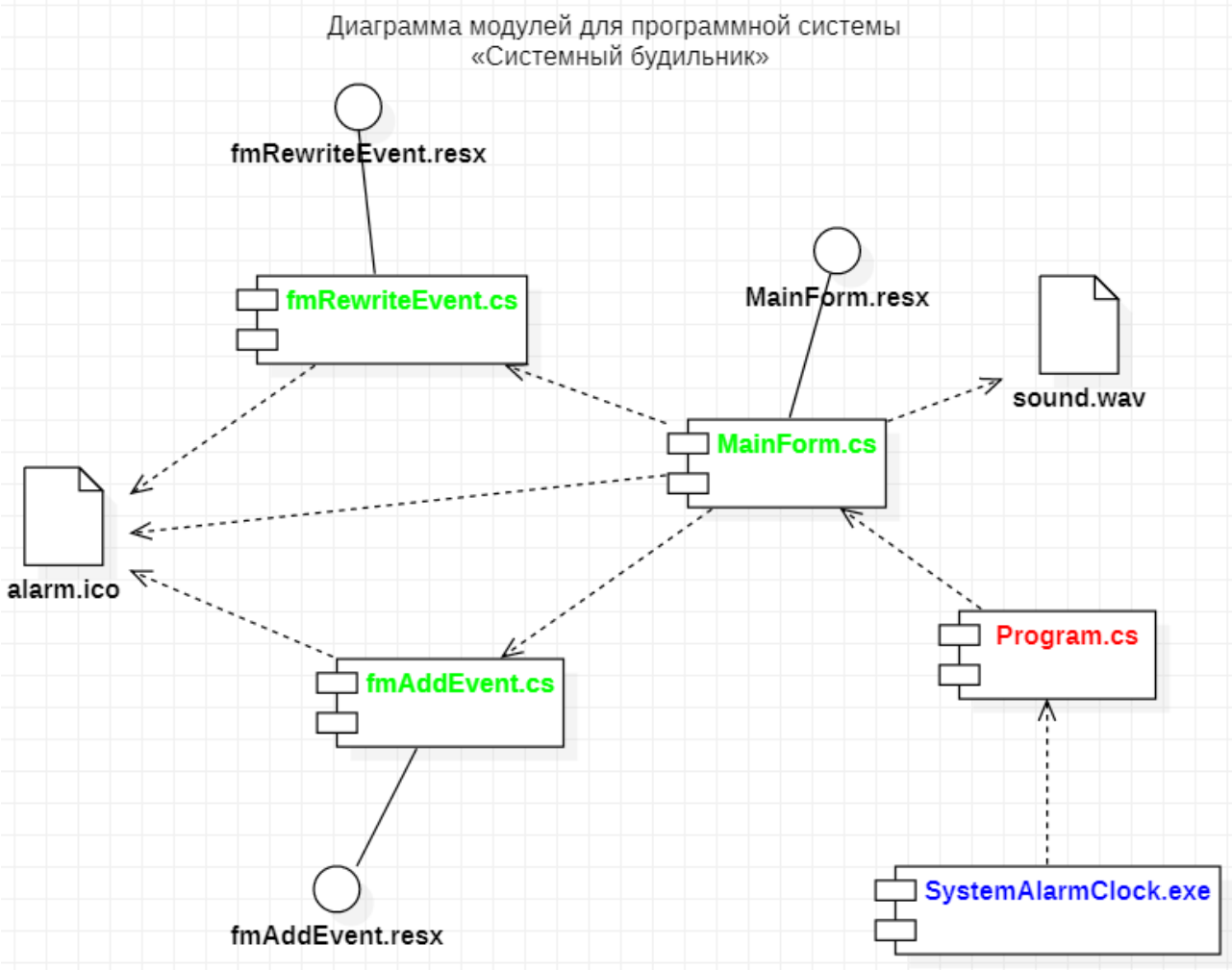

Рисунок 9 — Диаграмма модулей для программной системы «Системный будильник»

2.7 Разработка диаграммы развертывания

Диаграмма развертывания – это тип UML-диаграммы, которая показывает архитектуру исполнения системы, включая такие узлы, как аппаратные или программные среды исполнения, а также промежуточное программное обеспечение, соединяющее их.

Диаграммы развертывания обычно используются для визуализации физического аппаратного и программного обеспечения системы. Используя его, вы можете понять, как система будет физически развернута на аппаратном обеспечении.

Диаграммы развертывания помогают моделировать аппаратную топологию системы по сравнению с другими типами UML-диаграмм, которые в основном описывают логические компоненты системы.

Рисунок 10 - Диаграмма развертывания для программной системы «Системный будильник»

Модуль представленный на диаграмме развертывания для программной системы «Системный будильник» (Рисунок 10), реализующий многопользовательский интерфейс, является продолжением поддержки программного продукта "Системный будильник".

3 Описание программы

3.1 Общие сведения

Разработанная программа имеет наименование «Системный будильник», исполняемый файл называется SystemAlarmClock.exe. Программа написана на языке программирования Visual C# в среде визуального программирования Visual Studio 2019 Community. Текст программы приведен в приложении А.

3.2 Функциональное назначение

Программа предназначена для работы с системным будильником.

3.3 Описание логической структуры

Программа является событийно управляемой. Логическая структура программы является многомодульной.

При запуске программы открывается главное окно программы.

На главной форме размещены элементы: 4 Button, 1 Timer, 1 ListBox, 1 Label.

Компонент buCreateEvent отвечает за добавление события.

Компонент buEditEvents отвечает за изменение события .

Компонент buDeleteEvent отвечает за удаление события.

Компонент buClose отвечает за выход из программы.

Компонент timer1 отвечает за сравнение времени событий с текущим временем.

Компонент listBox1 отвечает за список записей событий.

Компонент label1 отвечает за пометку «Список событий».

Описание созданных обработчиков событий приложения приведено в таблице 5.

Таблица 5 — Описание обработчиков событий модуля MainForm

Название программного модуля	Наименование формы	Имя процедуры	Примечание
1	2	3	4
MainForm	MainForm	MainForm()	Конструктор класса MainForm
		void Form1_Load(object sender, EventArgs e)	Обработчик события открытия главной формы приложения
		void buCreateEvent_Click(object sender, EventArgs e)	Обработчик события клика на кнопку «Добавить событие»
		void button2_Click(object sender, EventArgs e)	Обработчик события клика на кнопку «Удаить событие»
		void button3_Click(object sender, EventArgs e)	Обработчик события клика на кнопку «Выход»
		void button4_Click(object sender, EventArgs e)	Обработчик события клика на кнопку «Редактировать событие»
		void timer1_Tick(object sender, EventArgs e)	Обработчик события работы таймера
		void listBox1_MouseClick(object sender, MouseEventArgs e)	Обработчик события клика на событие в списке записей

При нажатии на кнопку «Добавить событие», находящуюся на главной форме программы, открывается окно для создания события .

На форме для создания события размещены элементы: 1 Button, 4 Label, 1 dateTimePicker1, 2 ComboBox.

Компонент buSave отвечает за сохранении информации о событии.

Компонент label1 отвечает за пометку «Напоминание».

Компонент label2 отвечает за пометку «За какое время требуется напомнить о событии?».

Компонент label3 отвечает за пометку «Дни».

Компонент label4 отвечает за пометку «Часы».

Компонент dateTimePicker1 отвечает за установку времени события.

Компонент comboBox1 отвечает за установку часов до напоминания.

Компонент comboBox2 отвечает за установку дней до напоминания.

Описание созданных обработчиков событий приложения приведено в таблице 6.

Таблица 6 — Описание обработчиков событий модуля fmAddEvent

Название программного модуля	Наименование формы	Имя процедуры	Примечание
1	2	3	4
fmAddEvent	fmAddEvent	fmAddEvent()	Конструктор класса fmAddEvent
		void button2_Click(object sender, EventArgs e)	Обработчик события клика на кнопку «Сохранить событие»
		void comboBox1_SelectedIndexChanged(object sender, EventArgs e)	Обработчик события выбора часов до события
		void comboBox2_SelectedIndexChanged(object sender, EventArgs e)	Обработчик события выбора дней до события
		void dateTimePicker1_ValueChanged(object sender, EventArgs e)	Обработчик события выбора даты и времени события

При нажатии на появившуюся кнопку, после нажатия на событие в списке событий, «Редактировать событие», открывается окно для редактирования события.

На форме для редактирования события размещены элементы: 1 Button, 4 Label, 1 dateTimePicker1, 2 ComboBox.

Компонент buSave отвечает за сохранении информации о событии.

Компонент label1 отвечает за пометку «Напоминание».

Компонент label2 отвечает за пометку «За какое время требуется напомнить о событии?».

Компонент label3 отвечает за пометку «Дни».

Компонент label4 отвечает за пометку «Часы».

Компонент dateTimePicker1 отвечает за установку времени события.

Компонент comboBox1 отвечает за установку часов до напоминания.

Компонент comboBox2 отвечает за установку дней до напоминания.

Описание созданных обработчиков событий приложения приведено в таблице 7.

Таблица 7 — Описание обработчиков событий модуля fmRewriteEvent

Название программного модуля	Наименование формы	Имя процедуры	Примечание
1	2	3	4
fmRewriteEvent	fmRewriteEvent	fmRewriteEvent()	Конструктор класса fmRewriteEvent
		void button2_Click(object sender, EventArgs e)	Обработчик события клика на кнопку «Сохранить событие»
		void comboBox1_SelectedIndexChanged(object sender, EventArgs e)	Обработчик события выбора часов до события
		void comboBox2_SelectedIndexChanged(object sender, EventArgs e)	Обработчик события выбора дней до события
		void dateTimePicker1_ValueChanged(object sender, EventArgs e)	Обработчик события выбора даты и времени события

3.4 Используемые технические средства

Приложение «SystemAlarmClock» предназначенно для работы на персональных компьютерах, имеющих следующие минимальные характеристики:

- тактовая частота процессора - 2 ГГц;
- оперативная память - 512 Мбайт;
- свободное место на жестком диске - 30 Мбайт.

В качестве сервера используется компьютер, имеющий следующие минимальные характеристики:

- тактовая частота процессора - 2 ГГц;
- оперативная память - 1024Мбайт;
- свободное место на жестком диске - 500Мбайт.

3.5 Вызов и загрузка

Приложение запускается исполняемым файлом SystemAlarmClock.exe.

3.6 Входные данные

Входные данные: информация о напоминании, дата события(день, месяц, год), время события(часы, минуты, секунды), время за которое требуется оповестить пользователя.

3.7 Выходные данные

Выходные данные: напоминания о предстоящем событии.

4 Программа и методика испытаний

4.1 Объект испытаний

Объектом испытаний является приложение «SystemAlarmClock», предназначенное для показа работы системного будильника со следующими функциями: добавление, редактирование, удаление напоминаний, вывод напоминания о скором наступлении события с соответствующим звуком будильника, вывод напоминания о завершении события.

4.2 Цель испытаний

Испытания проводятся с целью проверки работоспособности и надежности программы «SystemAlarmClock». Для проверки правильности работы программы необходимо испытать её на тестовом примере. Сравнивая полученные результаты, можно определить правильность работы программы.

4.3 Требования к программе

Приложение «SystemAlarmClock» должно выполнять все функции, указанные в техническом задании. Работа приложения не должна приводить к сбою. Общим требованием является создание дружественного интерфейса, с помощью которого пользователь мог бы легко и быстро найти нужную информацию и выполнить необходимые функции.

4.4 Требования к программной документации

Программная документация, предъявляемая на испытание, должна содержать следующие разделы:

- техническое задание;
- проектирование объектно-ориентированной модели;
- проектирование приложения;
- описание программы;
- программа и методика испытаний;
- описание применения;
- текст программы.

4.5 Средства и порядок испытаний

Во время испытания приложения «SystemAlarmClock» необходимо использовать персональный компьютер с установленной операционной системой Windows 7 и выше.

Для проверки правильности работы системы был принят следующий порядок действий:

а) запустить программу на выполнение;

б) провести тестирование программы по тестам;

в) сравнить реакции программы с ожидаемым результатом;

г) сделать выводы по результатам тестирования о работоспособности программы.

4.6 Методы испытаний

Для проверки правильности работы программы были разработаны тестовые примеры.

Тестовый пример 1. Создание события. Необходимо запустить приложение. Затем на экране появится главная форма приложения (Рисунок Б.1). После этого требуется нажать на кнопку «Добавить событие». Откроется окно для создания события. Затем ввести в поле «Напоминание» - «Тестовое событие», ввести время «15:35:00» и дату 21.11.2021, также выбрать время за которое требуется оповестить о предстоящем событии (в поле ввода времени выбрать «10», а в поле ввода дней «1») и нажать кнопку «Сохранить событие» (Рисунок Б.2). Затем в списке событий появится это событие (Рисунок Б.3).

Тестовый пример 2. Редактирование события. Необходимо запустить приложение. Затем на экране появится главная форма приложения (Рисунок Б.1). После этого требуется выбрать левым кликом мыши событие в списке событий, которое необходимо актуализировать и нажать на появившуюся кнопку «Редактировать событие» (Рисунок Б.4). Откроется панель для редактирования события. Затем ввести в поля «Напоминание» - «Тестовое событие1», ввести время «16:15:00»,и дату 24.11.2021, также выбрать время за которое требуется оповестить о предстоящем событии (в поле ввода времени выбрать «10», а в поле ввода дней «1») и нажать кнопку «Сохранить событие» (Рисунок Б.5). Затем в списке событий эта запись будет отображаться в измененном виде (Рисунок Б.6).

Тестовый пример 3. Удаление события. Необходимо запустить приложение. Затем на экране появится главная форма приложения (Рисунок Б.1). После этого требуется выбрать левым кликом мыши событие, которое необходимо изменить и нажать на появившуюся кнопку «Удалить событие» (Рисунок Б.4). После чего, появится окно подтверждения удаления события (Рисунок Б.7). После подтверждения удаления события, событие удалится из списка (Рисунок Б.8).

Тестовый пример 4. Получение оповещения о приближении события. Необходимо запустить приложение. Затем на экране появится главная форма приложения (Рисунок Б.1). После этого при достижении времени напоминания

о событии система выведет оповещение о предстоящем событии с звуковым сопровождением (Рисунок Б.9).

Тестовый пример 5. Получение оповещения о завершении события. Необходимо запустить приложение. Затем на экране появится главная форма приложения (Рисунок Б.1). После этого при достижении времени события система выведет оповещение о завершении события с предложением удалить событие из списка событий (Рисунок Б.10). При подтверждении удаления события окно оповещения будет закрыто, а событие удалено из списка событий (Рисунок Б.11).

5 Описание применения

5.1 Назначение программы

Разрабатываемое приложение должно быть предназначено для работы с персональным компьютером.

Система должна выполнять следующие функции:

- добавление, редактирование, удаление напоминаний;
- вывод напоминания о скором наступлении события;
- вывод напоминания о завершении события;
- воспроизведение звука будильника при напоминании.

5.2 Условия применения

Приложение «SystemAlarmClock» предназначено для работы на персональных компьютерах стандартной комплектации с операционной системой Windows 7 и выше.

5.3 Управление программой

Для работы с будильником необходимо запустить приложение. Затем на экране появится главная форма приложения (Рисунок Б.1). После этого требуется нажать на кнопку «Добавить событие». Затем ввести в поле «Напоминание», ввести время и дату, также выбрать время за которое требуется оповестить о предстоящем событии и нажать кнопку «Сохранить событие» (Рисунок Б.2). Затем в списке событий появится это событие (Рисунок Б.3). Далее, если пользователю нужно актуализировать данные события, необходимо выбрать нужное событие в списке событий и нажать и нажать на появившуюся кнопку «Редактировать событие» (Рисунок Б.4). Затем ввести актуализированные данные во

всех полях и нажать кнопку «Сохранить событие» (Рисунок Б.5). После этого в списке событий отобразится актуализированное событие (Рисунок Б.6). Для удаления события из списка пользователю потребуется выбрать нужное событие из списка и нажать на кнопку «Удалить событие» (Рисунок Б.4).После чего, появится окно подтверждения удаления события (Рисунок Б.7). После подтверждения удаления события, событие удалится из списка (Рисунок Б.8). Оповещение о приближении события будет выведено, при достижении времени напоминания о событии система выведет оповещение о предстоящем событии с звуковым сопровождением (Рисунок Б.9).

Заключение

В ходе выполнения курсового проекта была разработана программа для работы с системным будильником. Выделены все основные функции будильника.

Получены практические навыки работы с CASE-средством StarUML. Разработка приложения выполнена в среде программирования Visual Studio 2019 Community. Результаты тестирования показали, что все требования технического задания выполнены в полном объёме и свидетельствуют о правильности работы программы.

Список используемых источников

1. Якобсон А., Буч Г., Рамбо Дж. Унифицированный процесс разработки программного обеспечения. - СПб.: Питер, 2002. - 496 с.

2. Буч Г., Рамбо Д., Джекобсон А. Язык UML. Руководство пользователя. - М.: ДМК, 2000. - 432 с.

3. Новиков Ф.А. Учебно-методическое пособие по дисциплине «Анализ и проектирование на UML». - Спб.: СПбГУ ИТМО, 2007. - 286 с.

4. Иванова, Г. С. Объектно-ориентированное программирование: учебник для вузов. - 3-е изд. / Г. С. Иванова, Т. Н. Никушкина, Е. К. Пугачев / Под ред. Г. С. Ивановой. - М.: Изд-во МГТУ им. Н. Э. Баумана, 2007. - 368 с.

5. Павловская, Т. А. C++. Объектно-ориентированное программирование: Практикум / Павловская Т. А., Щупак Ю. А. - Спб.: Питер, 2006. - 265 с. - (Учеб. пособие).

Приложение А

(обязательное)

Результаты выполнения тестовых примеров

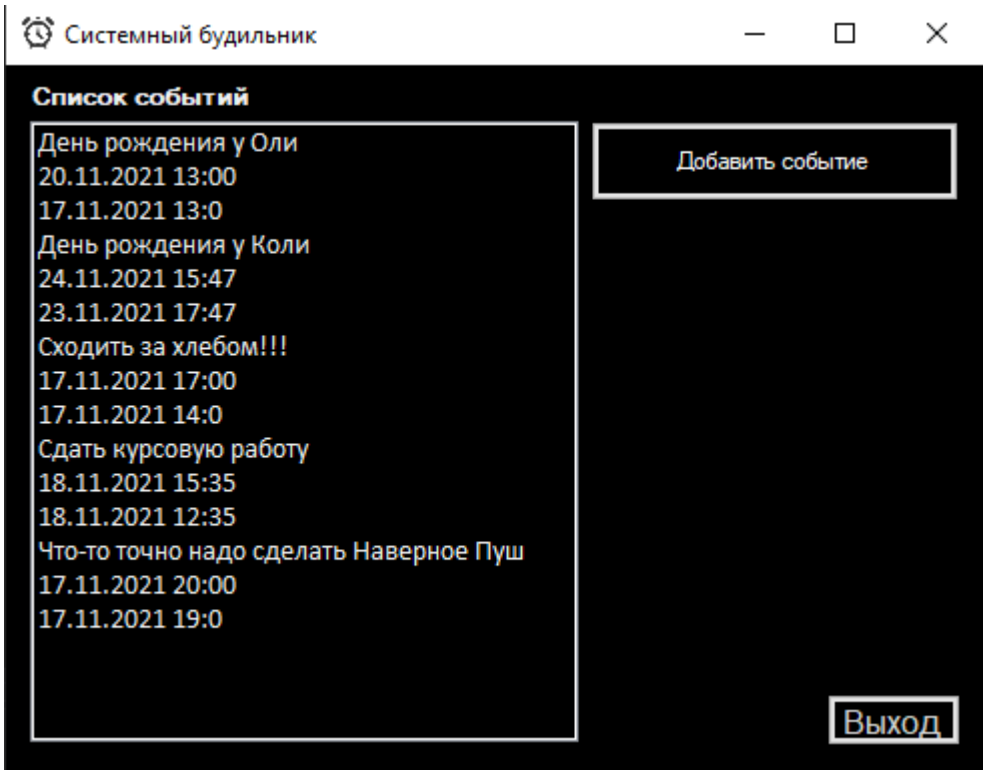

Рисунок А.1 – Главная форма программы

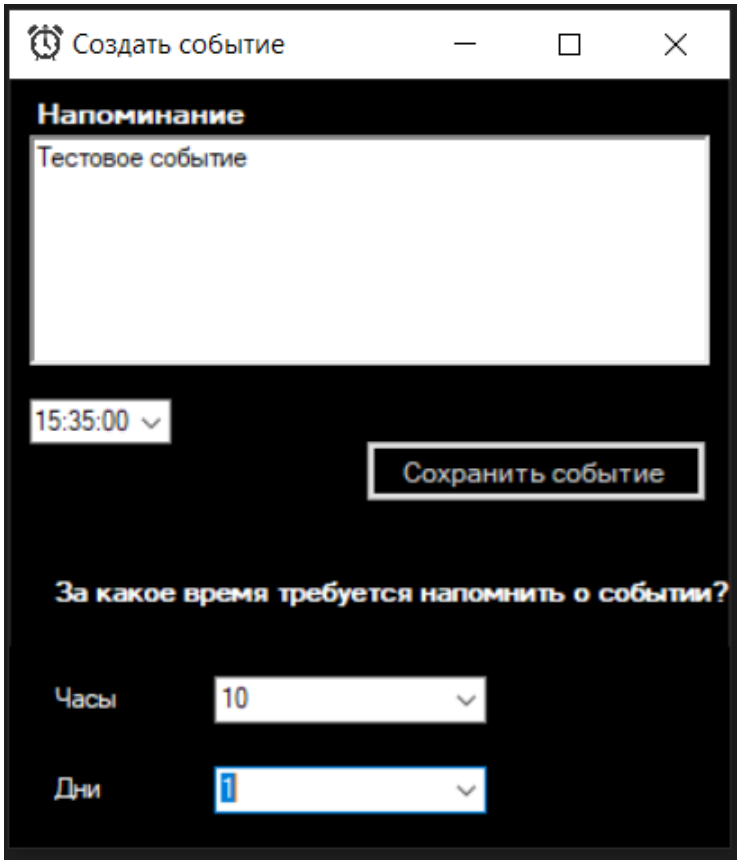

Рисунок А.2 – Ввод данных для создания события

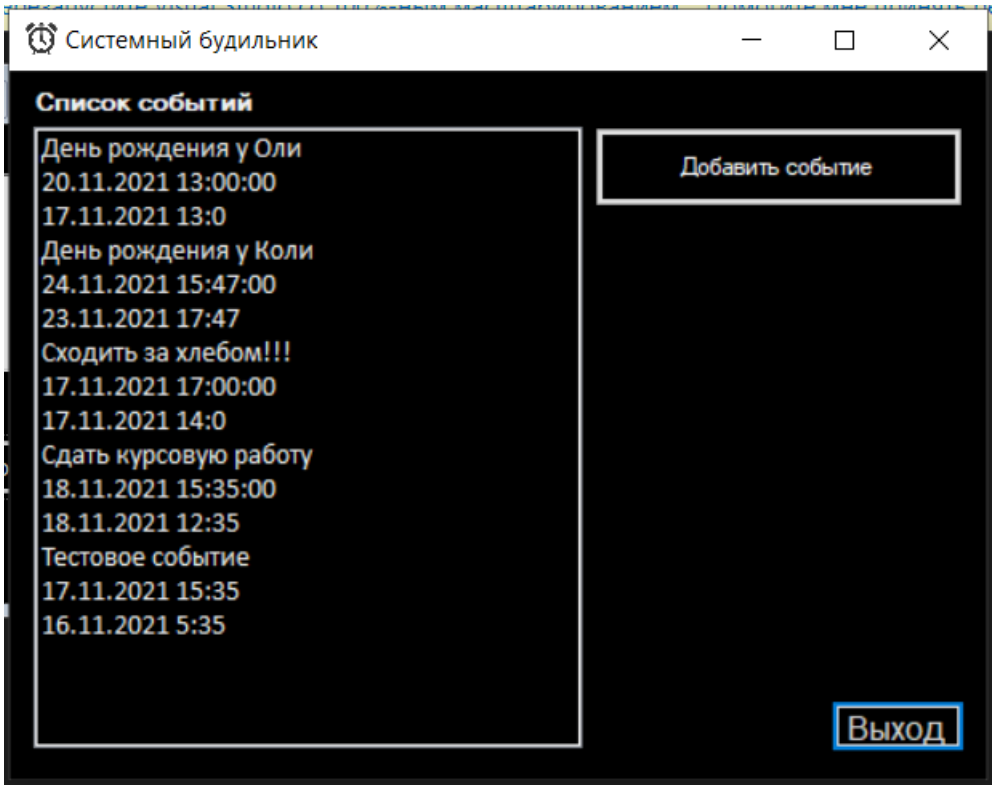

Рисунок А.3 – Событие добавлено в список событий

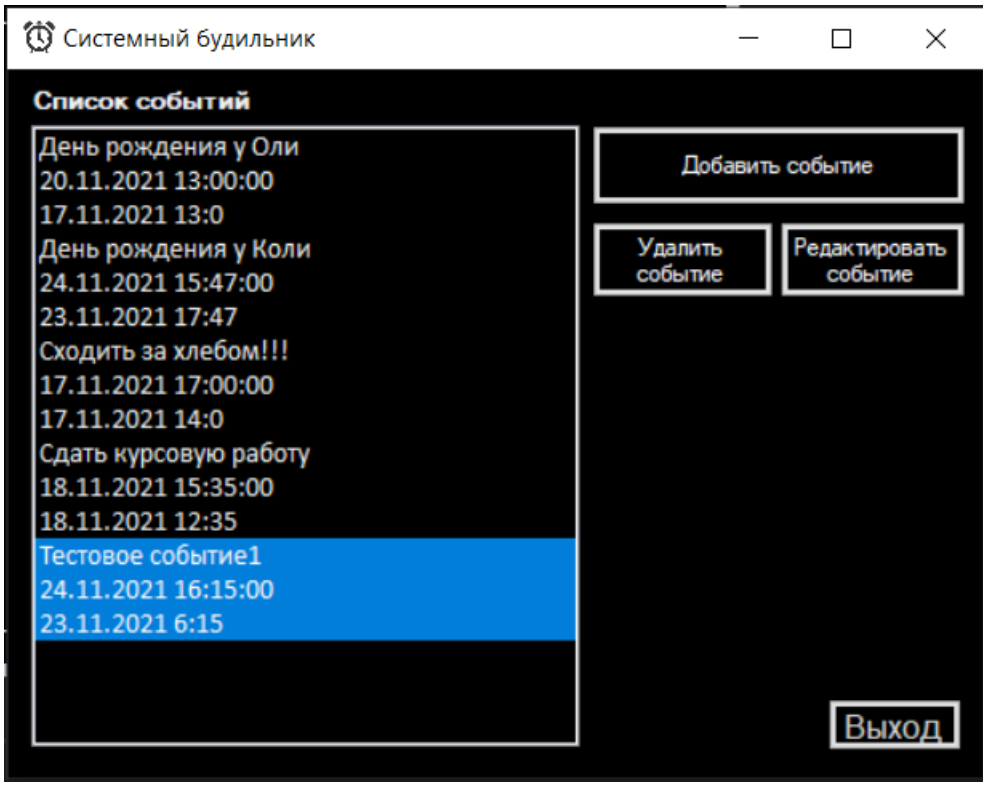

Рисунок А.4 — Появление кнопок для удаления и редактирования события

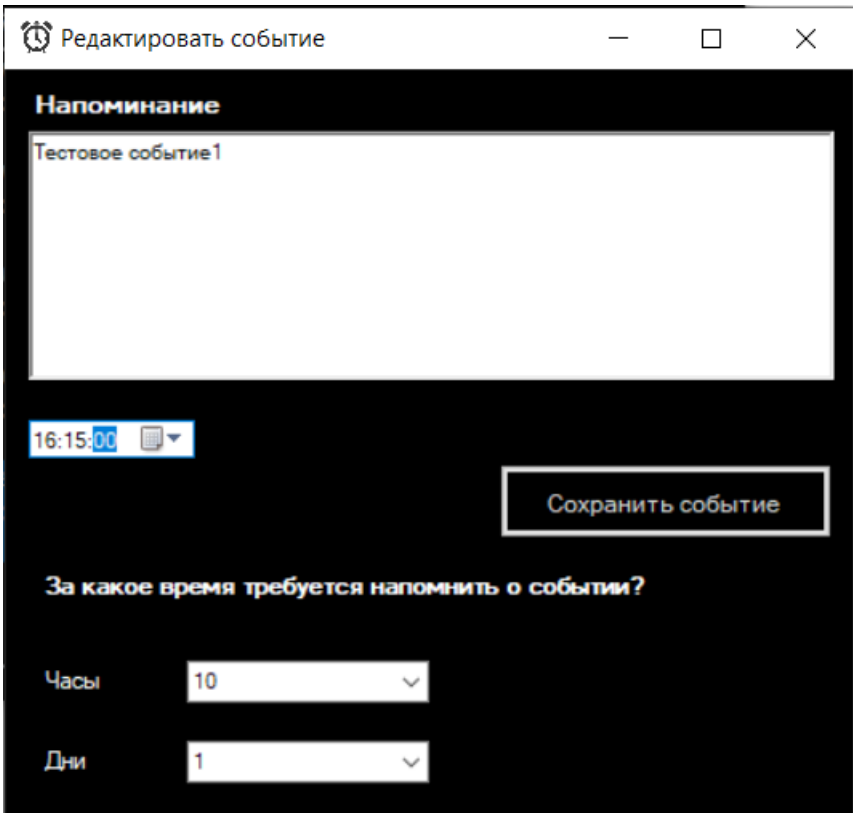

Рисунок Б.5 — Актуализация данных о событии

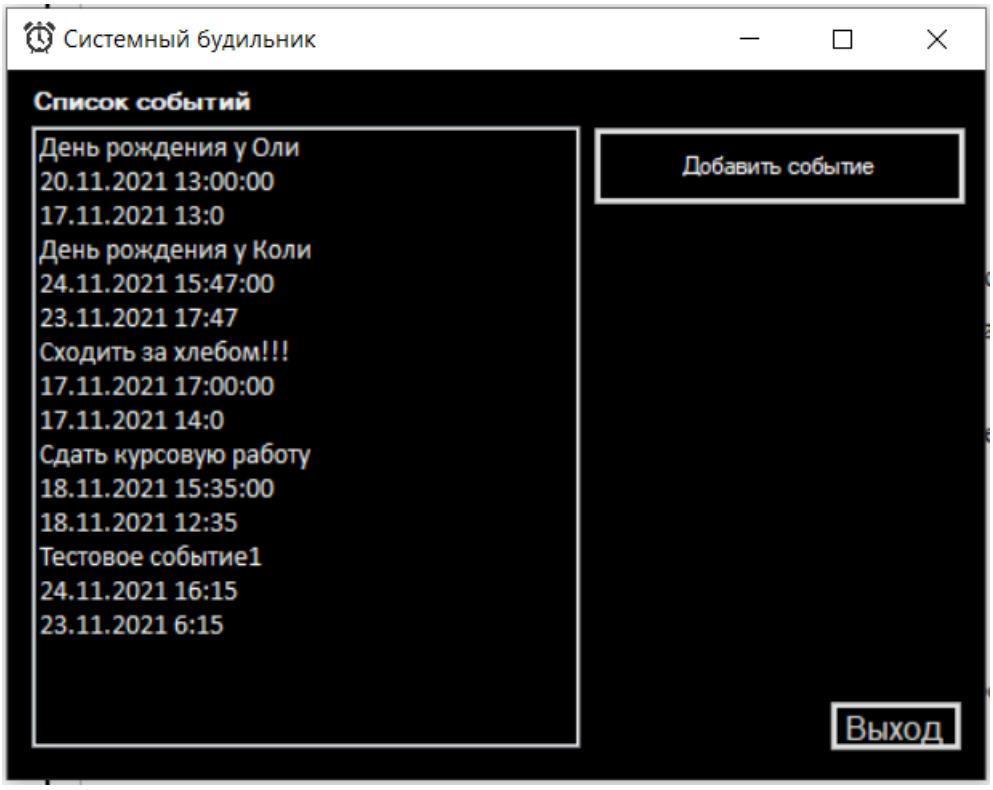

Рисунок А.6 — Актуализированный список событий

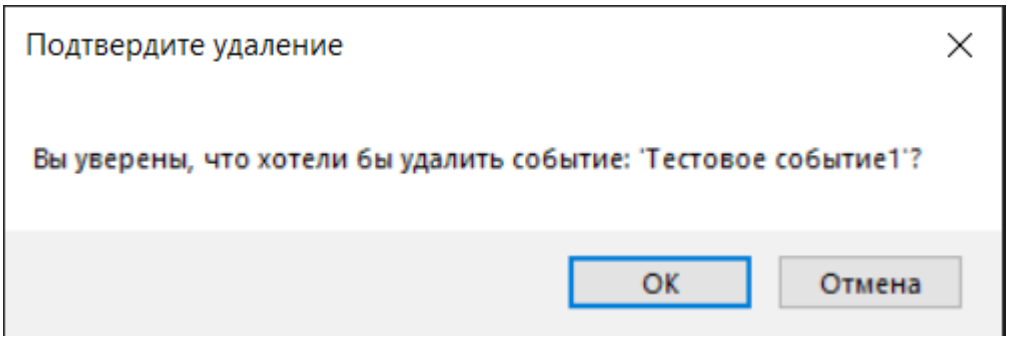

Рисунок А.7 — Подтверждение удаления события

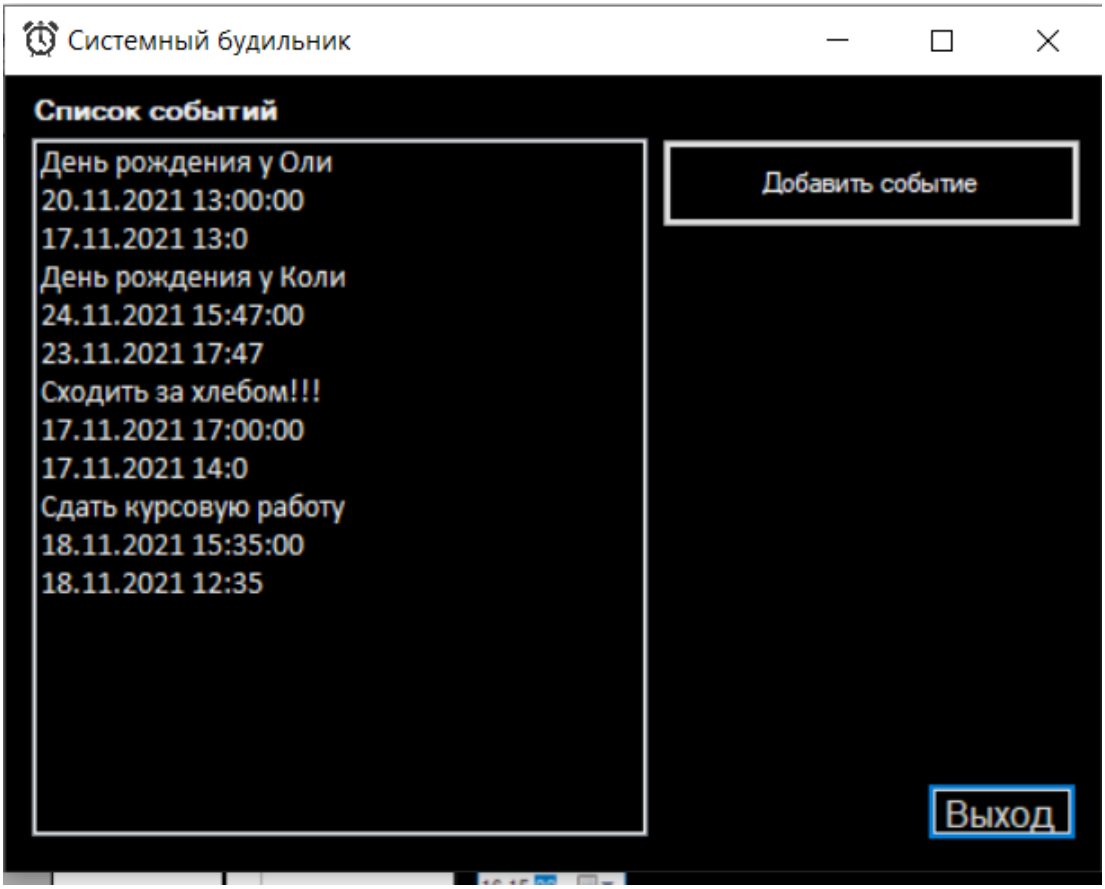

Рисунок Б.8 — Событие удалено

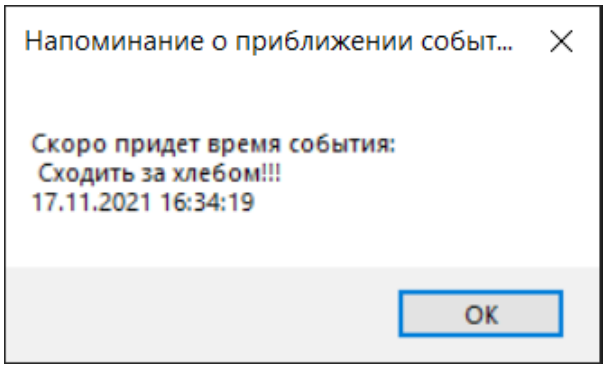

Рисунок Б.9 — Оповещение о событии

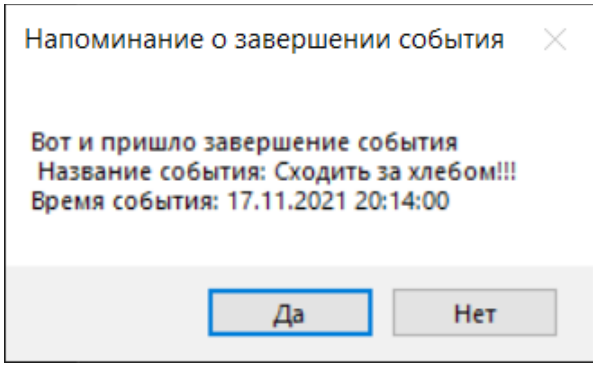

Рисунок Б.10 — Оповещение о завершении события

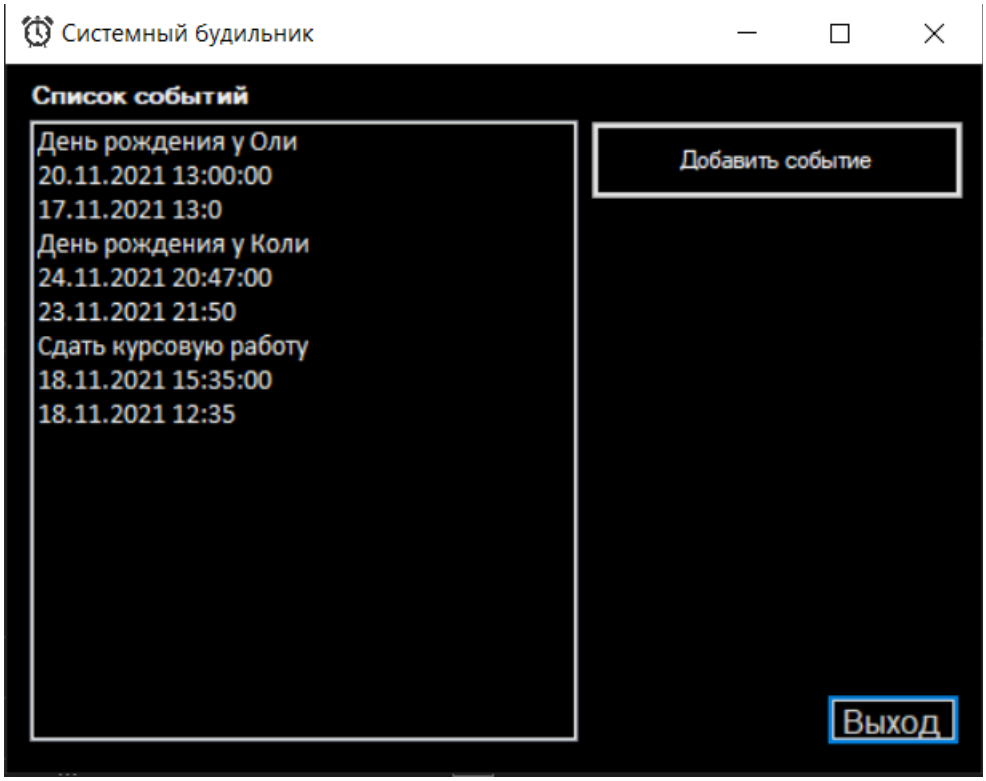

Рисунок Б.11 — Завершенное событие удалено

Приложение Б

(обязательное)

Листинг программы

// Имя файла «Program.cs»

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SystemAlarmClock
{
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new MainForm());
        }
    }
}
```

// Имя файла «MainForm.cs»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Media;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SystemAlarmClock
{
       public partial class MainForm : Form
       {
              String str = "";
              string FileName = "DB.txt";
              int a = -1;

              public MainForm()
              {
                     InitializeComponent();
              }

              private void Form1_Load(object sender, EventArgs e)
              {
                     timer1.Enabled = true;
                     StreamReader reader = new StreamReader(FileName);
                     while (!reader.EndOfStream)
                     {
                            listBox1.Items.Add(reader.ReadLine());
                     }
                     reader.Close();
              }

              private void buCreateEvent_Click(object sender, EventArgs e)
              {
```

```
                fmAddEvent form2 = new fmAddEvent();
                form2.Owner = this;
                form2.Show();

        }

        private void button2_Click(object sender, EventArgs e)
        {
                if (MessageBox.Show($"Вы уверены, что хотели бы удалить событие:
'{str}'?", "Подтвердите удаление",
                        MessageBoxButtons.OKCancel) == DialogResult.OK)
                {
                        deleteEvent(a);
                        rewriteBDEvent(FileName);
                }
        }

        private void button3_Click(object sender, EventArgs e)
        {
                Close();
        }

        private void button4_Click(object sender, EventArgs e)
        {
                fmRewriteEvent form3 = new fmRewriteEvent();
                form3.Owner = this;
                form3.Show();
                form3.rewritableRecords(str);
        }

        private void listBox1_MouseClick(object sender, MouseEventArgs e)
        {
                if (a != -1)
                {
                        listBox1.ClearSelected();
                        a = -1;
                        buDeleteEvent.Visible = false;
                        buEditEvents.Visible = false;
                }
                else
                {
                        a = listBox1.SelectedIndex;
                        if (a % 3 == 0)
                        {
                                listBox1.ClearSelected();
                                listBox1.SetSelected(a, true);
                                str = listBox1.Items[a].ToString();
                                listBox1.SetSelected(a + 1, true);
                                listBox1.SetSelected(a + 2, true);
                                //listBox1.SetSelected(a + 3, true);
                        }
                        if (a % 3 == 1)
                        {
                                listBox1.ClearSelected();
                                listBox1.SetSelected(a - 1, true);
                                str = listBox1.Items[a-1].ToString();
                                listBox1.SetSelected(a, true);
                                listBox1.SetSelected(a + 1, true);
                                //listBox1.SetSelected(a + 2, true);
                        }
                        if (a % 3 == 2)
                        {
                                listBox1.ClearSelected();
```

```
                                listBox1.SetSelected(a - 2, true);
                                str = listBox1.Items[a-2].ToString();
                                listBox1.SetSelected(a - 1, true);
                                listBox1.SetSelected(a, true);
                                //listBox1.SetSelected(a + 1, true);
                            }

                            buDeleteEvent.Visible = true;
                            buEditEvents.Visible = true;
                        }
                    }

                    private void timer1_Tick(object sender, EventArgs e)
        {
            DateTime systemTime = DateTime.Now;
            eventReminder(systemTime);
            deletionReminder(systemTime);
        }

        private void eventReminder(DateTime systemTime)
        {
            for (int i = 0; i < (listBox1.Items.Count + 1) / 3; i++)
            {
                DateTime date = DateTime.Parse(Convert.ToString(listBox1.Items[2 + i *
3])); // это время до события
                if (date.Year == systemTime.Year)
                    if (date.Month == systemTime.Month)
                        if (date.Day == systemTime.Day)
                            if (date.Hour == systemTime.Hour)
                                if (date.Minute == systemTime.Minute)
                                {
                                                        SoundPlayer sp = new
SoundPlayer("F:\\JOB\\7 семак\\Системное
ПО\\КУРСАЧ\\SystemAlarmClock\\Resourses\\sound.wav");
                                                        sp.Play();
                                                        MessageBox.Show("Скоро придет
время события: \n " +

       $"{Convert.ToString(listBox1.Items[0 + i * 3])}\n" +

       $"{Convert.ToString(listBox1.Items[1 + i * 3])}",
                                                                    "Напоминание о
приближении события");
                                                    }
            }
        }

        private void deletionReminder(DateTime systemTime)
        {
            for (int i = 0; i < (listBox1.Items.Count + 1) / 3; i++)
            {
                            DateTime date1 =
DateTime.Parse(Convert.ToString(listBox1.Items[1 + i * 3])); // это время события
                if (date1.Year <= systemTime.Year)
                    if (date1.Month <= systemTime.Month)
                        if (date1.Day <= systemTime.Day)
                            if (date1.Day == systemTime.Day)
                            {
                                if (date1.Hour <= systemTime.Hour)
                                    if (date1.Hour == systemTime.Hour)
                                    {
                                        if (date1.Minute <= systemTime.Minute)
                                        {
```

```
                                        deletingEvent(i);
                                    }
                                }
                                else
                                {
                                    deletingEvent(i);
                                }
                            }
                            else
                            {
                                deletingEvent(i);
                            }
                        }
                    }

                    private void deletingEvent(int i)
                    {
                        if (MessageBox.Show($"Вот и пришло завершение события \n " +
                                                $"Назавение события:
{Convert.ToString(listBox1.Items[0 + i * 3])} \n"+
                                                $"Время события:
{Convert.ToString(listBox1.Items[1 + i * 3])}",
                                                "Напоминание о завершении события",
                                                MessageBoxButtons.YesNo) ==
DialogResult.Yes)
                        {
                            deleteEvent(2 + i * 3);
                            rewriteBDEvent(FileName);
                        }
                    }

                    public void recordList(string str)
                    {
                        listBox1.Items.Add(str);
                    }

                    public void recordList(DateTime dt)
                    {
                        listBox1.Items.Add(dt);
                    }

                    public void rewriteBDEvent(String FileName)
                    {
                        FileStream file = new FileStream(FileName, FileMode.Create); //создаем
файловый поток
                        StreamWriter writer = new StreamWriter(file); //создаем «потоковый
писатель» и связываем его с файловым потоком
                        for (int i = 0; i < listBox1.Items.Count; i++)
                        {
                            writer.WriteLine(listBox1.Items[i]);
                        }
                        writer.Close();
                    }

                    public void deleteEvent(int count)
                    {
                        if (count % 3 == 0)
                        {
                            listBox1.ClearSelected();
                            listBox1.Items.RemoveAt(count);
                            listBox1.Items.RemoveAt(count);
                            listBox1.Items.RemoveAt(count);
                        }
```

```
                if (count % 3 == 1)
                {
                        listBox1.ClearSelected();
                        listBox1.Items.RemoveAt(count - 1);
                        listBox1.Items.RemoveAt(count - 1);
                        listBox1.Items.RemoveAt(count - 1);
                }
                if (count % 3 == 2)
                {
                        listBox1.ClearSelected();
                        listBox1.Items.RemoveAt(count - 2);
                        listBox1.Items.RemoveAt(count - 2);
                        listBox1.Items.RemoveAt(count - 2);
                }
                buDeleteEvent.Visible = false;
                buEditEvents.Visible = false;
                a = -1;
        }

        public void rewriteEvent(String str1, DateTime dateTime1, String str2)
        {
                if (a % 3 == 0)
                {
                        listBox1.ClearSelected();
                        listBox1.Items[a] = str1;
                        listBox1.Items[a + 1] = dateTime1;
                        listBox1.Items[a + 2] = str2;
                }
                else if (a % 3 == 1)
                {
                        listBox1.ClearSelected();
                        listBox1.Items[a - 1] = str1;
                        listBox1.Items[a] = dateTime1;
                        listBox1.Items[a + 1] = str2;
                }
                else if (a % 3 == 2)
                {
                        listBox1.ClearSelected();
                        listBox1.Items[a - 2] = str1;
                        listBox1.Items[a - 1] = dateTime1;
                        listBox1.Items[a] = str2;
                }
                buDeleteEvent.Visible = false;
                buEditEvents.Visible = false;
                buCreateEvent.Enabled = true;
                a = -1;
                rewriteBDEvent(FileName);
        }


    }
}
```

// Имя файла «fmAddEvent.cs»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
```

```
namespace SystemAlarmClock
{
    public partial class fmAddEvent : Form
    {


        public fmAddEvent()
        {
            InitializeComponent();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            String reminder = "";
            DateTime eventDateTime;
            String reminderDateTime;
            String FileName = "DB.txt";

            if (richTextBox1.Text == "" || dateTimePicker1.Value < DateTime.Now)
            {
                MessageBox.Show("Ошибка ввода данных", "Ошибка!!!", MessageBoxButtons.OK);
            }
            else
            {
                reminder = richTextBox1.Text;
                eventDateTime = dateTimePicker1.Value;
                int d = (int)eventDateTime.Day;
                int m = (int)eventDateTime.Month;
                int cd = comboBox2.SelectedIndex;
                int ch = comboBox1.SelectedIndex;
                cd++;
                ch++;
                if (d - cd < 0)
                {
                    d = d + (d - cd);
                    m--;
                }
                else
                {
                    d = d - cd;
                }
                int h = (int)eventDateTime.Hour;
                if (h - ch < 0)
                {
                    h = 24 + (h - ch);
                    d--;
                }
                else
                {
                    h = h - ch;
                }
                reminderDateTime = $"{d}.{m}." +
                    $"{eventDateTime.Year} {h}:{eventDateTime.Minute}";
                if (this.Owner is MainForm owner)
                {
                    owner.recordList(reminder);
                    owner.recordList(eventDateTime);
                    owner.recordList(reminderDateTime);
                    owner.rewriteBDEvent(FileName);
                }

                Close();
```

```
            }

        }

        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            buSave.Visible = true;

        }

        private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
        {
            buSave.Visible = true;
        }

        private void dateTimePicker1_ValueChanged(object sender, EventArgs e)
        {
            this.Controls.Add(panel2);
            panel2.Visible = true;
            this.Height = 380;
        }
    }
}
```

// Имя файла «fmRewriteEvent.cs»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SystemAlarmClock
{
    public partial class fmRewriteEvent : Form
    {
        public fmRewriteEvent()
        {
            InitializeComponent();
        }

        public void rewritableRecords(string str)
        {
            richTextBox1.Text = str;
        }

        private void dateTimePicker1_ValueChanged(object sender, EventArgs e)
        {
                this.Controls.Add(panel2);
                panel2.Visible = true;
                this.Height = 410;
        }

        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            buSave.Visible = true;
        }

        private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
        {
```

```
            buSave.Visible = true;
        }

        private void buSave_Click(object sender, EventArgs e)
        {
            String reminder = "";
            DateTime eventDateTime;
            String reminderDateTime;
            String FileName = "DB.txt";

            if (richTextBox1.Text == "" || dateTimePicker1.Value < DateTime.Now)
            {
                MessageBox.Show("Ошибка ввода данных", "Ошибка!!!", MessageBoxButtons.OK);
            }
            else
            {
                reminder = richTextBox1.Text;
                eventDateTime = dateTimePicker1.Value;
                int d = (int)eventDateTime.Day;
                int m = (int)eventDateTime.Month;
                int cd = comboBox2.SelectedIndex;
                int ch = comboBox1.SelectedIndex;
                cd++;
                ch++;
                if (d - cd < 0)
                {
                    d = d + (d - cd);
                    m--;
                }
                else
                {
                    d = d - cd;
                }
                int h = (int)eventDateTime.Hour;
                if (h - ch < 0)
                {
                    h = 24 + (h - ch);
                    d--;
                }
                else
                {
                    h = h - ch;
                }
                reminderDateTime = $"{d}.{m}." +
                    $"{eventDateTime.Year} {h}:{eventDateTime.Minute}";
                if (this.Owner is MainForm owner)
                {
                    owner.rewriteEvent(reminder, eventDateTime, reminderDateTime);
                }

                Close();
            }
        }
    }
}
```