МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА № 42

КУРСОВАЯ РАБОТА
ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

канд. техн. наук, доцент		Н. В. Богословская
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОЙ РАБОТЕ

БИБЛИОТЕКА УПРАВЛЕНИЯ ТЕКСТОВЫМИ ФАЙЛАМИ И ПРИЛОЖЕНИЕ «ФАЙЛОВЫЙ МЕНЕДЖЕР»

по дисциплине: ОСНОВЫ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. № 4326		Г. С. Томчук
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2024

СОДЕРЖАНИЕ

1 Анализ поставленной задачи

Целью курсовой работы является разработка пользовательской библиотеки классов для работы с текстовыми файлами, а также разработка программного приложения «Файловый менеджер» с возможностью управления узлами файловой системы (папками и файлами) с использованием объектно-ориентированных принципов программирования. Приложение должно обеспечивать функциональность для работы с файловой системой, хранения и управления данными о файловых узлах, а также взаимодействия объектов различных классов.

Для достижения поставленной цели было необходимо решить следующие задачи:

1. Объекты классов должны содержать следующую информацию:
    - Название узла файловой системы (папки или файла);
    - Путь к узлу;
    - Дата создания узла;
    - Для файлов:
        - Текстовое содержимое;
        - Тип файла;
        - Флаг Read-Only;
        - Иконка файла;
        - Авторы файла.
2. Объекты классов должны иметь методы для:
    - Доступа и изменения всех перечисленных свойств;
    - Создания узлов в файловой системе ОС;
    - Удаления, переименования и перемещения узлов;
    - Для рабочих текстовых файлов:
        - Отрисовки иконки;
        - Манипуляций со связанным файлом с метаинформацией.
3. Обеспечение структуры классов и их взаимодействия:

- Создание всех необходимых конструкторов (с параметрами и по умолчанию);
- Реализация свойств для установки и получения значений всех данных объекта;
- Связь рабочих текстовых файлов с файлами метаданных.

4. Использование ООП-принципов:
   - Применение механизма наследования для реализации иерархии классов: базовый класс для общих свойств и методов узлов файловой системы, производные классы для текстовых и мета-файлов.
   - Инкапсуляция данных объектов классов.
   - Использование интерфейсов для обеспечения единого подхода к работе с узлами файловой системы.
   - Организация событий для уведомления об изменении параметров объектов (например, изменение пути).
5. Обеспечение пользовательского интерфейса:
   - Реализация функциональности для взаимодействия пользователя с узлами файловой системы, включая создание, удаление, перемещение, изменение свойств и отображение информации об объектах.

Разработанные классы и их функциональность могут быть использованы как готовые компоненты в следующих категориях информационных систем, прикладных программных решений и интерфейсов пользователей:

1. Файловые менеджеры:

   Программное обеспечение, предназначенное для работы с файловыми системами, например, аналогов Windows Explorer или Total Commander. Разработанные классы могут быть использованы для представления, управления и отображения данных о файлах и папках.

2. Системы резервного копирования и восстановления данных:
   В подобных решениях классы могут применяться для управления файлами и каталогами, обеспечения сохранности данных и их восстановления.
3. Редакторы метаданных:
   Классы, содержащие информацию об авторах и свойствах файлов, могут использоваться в приложениях для редактирования метаинформации, например, в медиапроигрывателях, системах управления медиабиблиотеками и файловыми тегами.
4. Платформы для управления документами (DMS):
   Объекты классов могут применяться для работы с документами, включая текстовые файлы и их метаинформацию, в корпоративных системах управления документами.
5. Программные интерфейсы для визуализации данных:
   Разработанные классы могут быть включены в приложения с графическим интерфейсом для отображения структуры файловой системы, включая иконки и свойства узлов.
6. Системы виртуальных файловых систем:
   Классы могут быть адаптированы для управления файлами и папками в виртуальных файловых системах, таких как облачные хранилища или эмуляторы локальных файловых систем.
7. Инструменты для анализа файловой структуры:
   Программы, используемые для анализа, визуализации и отчётности по содержимому файловых систем, например, системы мониторинга свободного места или дубликатов файлов.

В результате выполнения курсовой работы должно быть создано программное приложение, демонстрирующее эффективное использование объектно-ориентированных подходов в разработке и реализации сложных структур данных.

2 Диаграмма классов

Архитектура библиотеки представляет собой иерархическую структуру классов, обеспечивающую взаимодействие между объектами файловой системы. Диаграмма классов представлена на рис. 1:

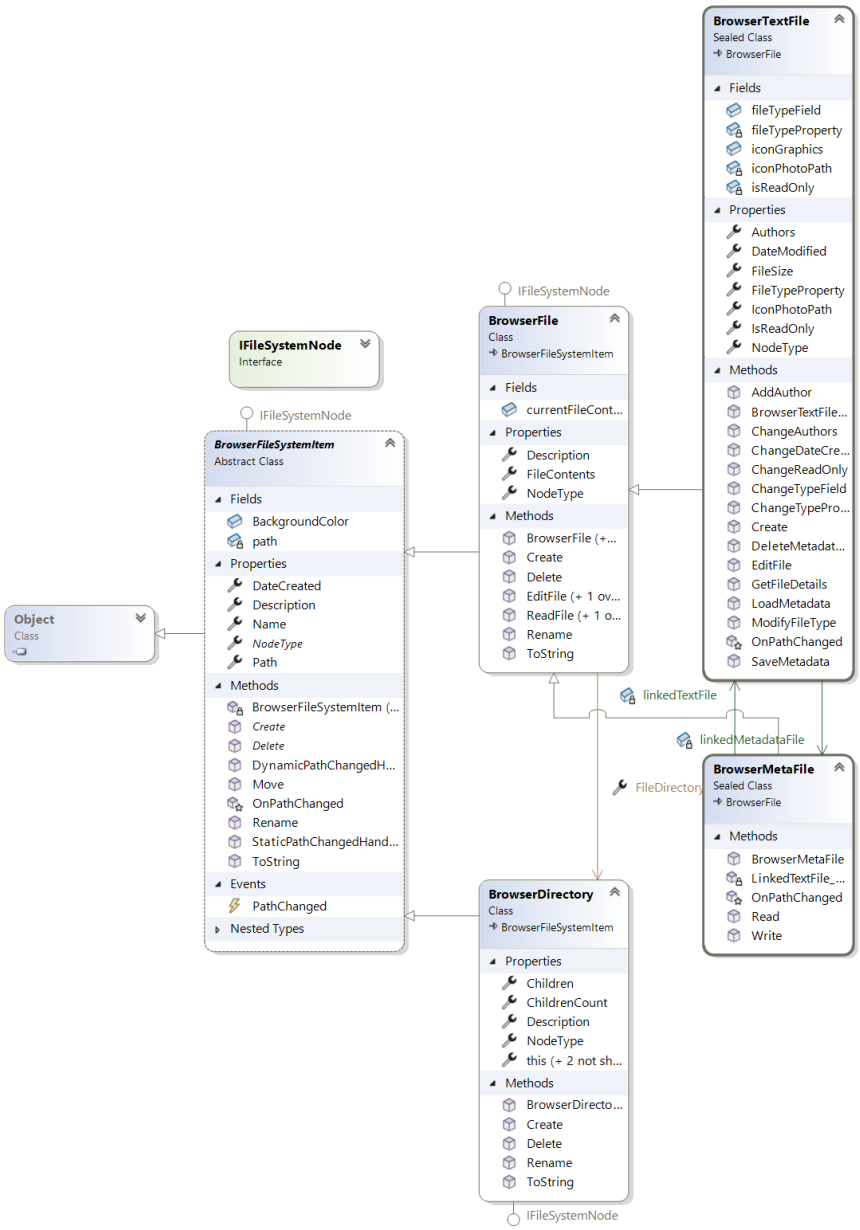


Рисунок 1

Основные элементы архитектуры:

1. Базовый абстрактный класс BrowserFileSystemItem:
   - Класс является основой для всех элементов файловой системы.
   - Определяет общие поля и свойства для хранения информации, такой как название узла, путь, дата создания.

- Объявляет базовые методы, такие как создание, удаление, переименование и перемещение узлов.
- Содержит событие PathChanged, сигнализирующее о смене пути узла.
- Реализует интерфейс IFileSystemNode, который обеспечивает взаимодействие объектов классов через стандартный набор свойств и методов.

2. Производные классы:
   - BrowserFile:
     - Наследуется от BrowserFileSystemItem и представляет рабочий файл.
     - Добавляет свойства и методы для управления содержимым файла, его типом, состоянием Read-Only и отображением.
     - Выступает базовым классом для специализированных файлов.
   - BrowserDirectory:
     - Наследуется от BrowserFileSystemItem и представляет папку файловой системы.
     - Содержит методы для управления файлами и папками внутри каталога.
3. Специализированные классы:
   - BrowserTextFile:
     - Наследуется от BrowserFile и расширяет функциональность для работы с текстовыми файлами.
     - Содержит свойства для управления авторами, иконками файла, размера файла и других метаданных.
   - BrowserMetaFile:
     - Наследуется от BrowserFile и используется для хранения метаинформации о связанных текстовых файлах.
     - Реализует методы для чтения и записи метаданных.

4. Отношения между классами:
    - Ассоциация (BrowserFile и BrowserDirectory):
        - Свойство FileDirectory в классе BrowserFile позволяет связать файл с папкой, в которой он находится. Это обеспечивает возможность организации файловой системы, где каждый файл принадлежит конкретному каталогу.
    - Композиция (BrowserTextFile и BrowserMetaFile):
        - Свойство linkedMetadataFile в классе BrowserTextFile обеспечивает жёсткую связь между рабочим файлом и его метаинформацией. Удаление рабочего файла влечёт удаление связанного файла метаданных.
    - Агрегация (BrowserTextFile и BrowserMetaFile):
        - Свойство linkedTextFile в классе BrowserMetaFile позволяет передать ссылку на существующий текстовый файл при создании объекта метаинформации. Это демонстрирует более слабую связь, где файл метаданных не зависит от жизненного цикла рабочего файла.
5. Интерфейс IFileSystemNode:
    - Используется для обеспечения стандартного интерфейса взаимодействия всех классов, представляющих элементы файловой системы.
    - Объявляет свойства и методы для получения имени, пути, даты создания узла, а также операции с объектами файловой системы.

Архитектура библиотеки реализована на основе объектно-ориентированных принципов, таких как наследование, инкапсуляция и полиморфизм. Связи между классами обеспечивают гибкость при добавлении и изменении функциональности. Использование интерфейса и событий делает библиотеку расширяемой и совместимой с различными прикладными решениями.

3 Описание программного кода классов

3.1 Обобщенный класс сущности файловой системы

На рис. 2 показан код класса BrowserFileSystemItem, являющегося базовым классом в иерархии.

```
// Обобщенный класс сущности файловой системы
public abstract class BrowserFileSystemItem : IFileSystemNode
{
    public static Color BackgroundColor;

    public string Name { get; set; }

    private string path;
    public string Path...

    public DateTime DateCreated { get; set; }

    // Обработчик и событие, уведомляющее об изменении свойства Path
    public delegate void BrowserFileSystemItemHandler(object sender, BrowserFileSystemItemEventArgs e);
    public event BrowserFileSystemItemHandler PathChanged;

    // Класс аргументов события
    public class BrowserFileSystemItemEventArgs : EventArgs
    {
        public string NewPath { get; }

        public BrowserFileSystemItemEventArgs(string newPath)...
    }

    // Статический конструктор
    static BrowserFileSystemItem()...

    // Конструктор по умолчанию
    public BrowserFileSystemItem()...

    // Перегруженный конструктор с параметрами
    public BrowserFileSystemItem(string name) : this() => Name = name;

    // Перегруженный конструктор с двумя параметрами
    public BrowserFileSystemItem(string name, string path) : this(name) => Path = path;

    // Create, Delete - данные методы абстрактные, так как реализация создания и удаления сущностей ФС (папок и файлов) не имеет ничего общего
    public abstract void Create();
    public abstract void Delete();

    public virtual void Rename(string newName)...

    public virtual void Move(string newPath)...
}
}
```

Рисунок 2

Класс BrowserFileSystemItem представляет абстрактную сущность файловой системы и служит базовым классом для других классов, таких как файлы и папки. Он реализует интерфейс IFileSystemNode и содержит свойства, методы и события, обеспечивающие базовую функциональность для управления узлами файловой системы. Основные элементы класса:

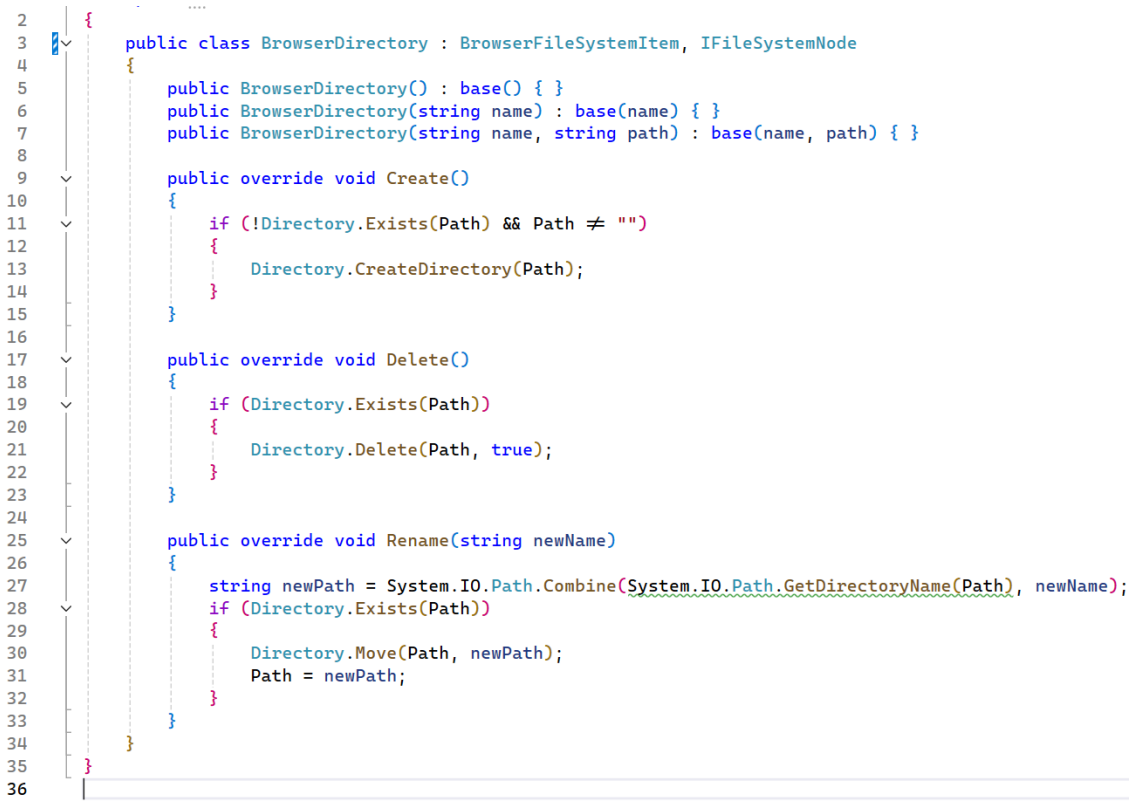
1. Свойства
    - Name (строка): название узла файловой системы. Может быть как имя файла, так и папки.
    - Path (строка): путь к узлу. При изменении этого свойства вызывается событие PathChanged.
    - DateCreated (DateTime): дата создания элемента.
2. Событие PathChanged

- Уведомляет об изменении свойства Path.
- Использует делегат BrowserFileSystemItemHandler, принимающий объект-отправитель и аргументы события.

3. Вложенный класс BrowserFileSystemItemEventArgs
   - Представляет аргументы для события PathChanged.
   - Содержит свойство NewPath, хранящее новое значение пути.
4. Конструкторы
   - Конструктор по умолчанию: устанавливает имя узла как "NewItem" и путь как @"NewDirectory\".
   - Перегруженный конструктор с параметром name: задает имя узла.
   - Перегруженный конструктор с параметрами name и path: задает имя и путь узла.
5. Абстрактные методы
   - Create(): абстрактный метод для создания элемента файловой системы (реализуется в производных классах).
   - Delete(): абстрактный метод для удаления элемента файловой системы (реализуется в производных классах).
6. Виртуальные методы
   - Rename(string newName): переименовывает элемент, обновляя его имя и путь.
   - Move(string newPath): перемещает элемент в новый путь.

3.2 Класс узла директории

На рис. 3 показан код класса BrowserDirectory – класса директории (папки) файловой системы.



```
{
    public class BrowserDirectory : BrowserFileSystemItem, IFileSystemNode
    {
        public BrowserDirectory() : base() { }
        public BrowserDirectory(string name) : base(name) { }
        public BrowserDirectory(string name, string path) : base(name, path) { }

        public override void Create()
        {
            if (!Directory.Exists(Path) && Path ≠ "")
            {
                Directory.CreateDirectory(Path);
            }
        }

        public override void Delete()
        {
            if (Directory.Exists(Path))
            {
                Directory.Delete(Path, true);
            }
        }

        public override void Rename(string newName)
        {
            string newPath = System.IO.Path.Combine(System.IO.Path.GetDirectoryName(Path), newName);
            if (Directory.Exists(Path))
            {
                Directory.Move(Path, newPath);
                Path = newPath;
            }
        }
    }
}
```

Рисунок 3

Класс BrowserDirectory представляет конкретную реализацию для работы с директориями (папками) файловой системы. Он наследуется от абстрактного базового класса BrowserFileSystemItem и реализует интерфейс IFileSystemNode. Класс обеспечивает функциональность для создания, удаления, переименования и перемещения папок. Основные элементы класса:

1. Конструкторы
    - Конструктор по умолчанию:

      Вызывает базовый конструктор, инициализирует объект с настройками по умолчанию.
    - Конструктор с параметром name:

      Вызывает соответствующий перегруженный конструктор базового класса, задающий имя директории.
    - Конструктор с параметрами name и path:

      Инициализирует объект с заданными именем и путем к папке.

2. Переопределенные методы

   Класс переопределяет абстрактные методы базового класса, чтобы реализовать функциональность работы с папками:

   - Create()

     Создает новую директорию в файловой системе, если она еще не существует и путь не пустой

   - Delete()

     Удаляет существующую директорию и все ее содержимое, если директория существует

   - Rename(string newName)

     Переименовывает директорию, обновляя ее путь:

     1. Использует метод Directory.Move для изменения имени директории в файловой системе.
     2. Обновляет свойство Path объекта, чтобы отражать изменения.

3.3 Класс узла файла

На рис. 4 показан код класса BrowserFile – класса файла файловой системы.

```
2   {
3       public class BrowserFile : BrowserFileSystemItem, IFileSystemNode
4       {
5           public BrowserDirectory FileDirectory { get; set; } // Ассоциация BrowserFile -> BrowserDirectory
6
7           public string currentFileContents;
8           public string FileContents...
17
18          public BrowserFile() : base() { }
19          public BrowserFile(string name) : base(name) { }
20          public BrowserFile(string name, string path) : base(name, path) { }
21
22          public override void Create()...
41
42          public override void Delete()...
49
50          public override void Rename(string newName)...
62
63          public void EditFile(string content)...
70
71          public void EditFile(SaveFileDialog saveFileDialog, string content)...
83
84          public string ReadFile()...
92
93          public string ReadFile(OpenFileDialog openFileDialog)...
105     }
106 }
107
```

Рисунок 4

Класс BrowserFile представляет реализацию сущности файла в

файловой системе. Он наследуется от базового класса BrowserFileSystemItem и реализует интерфейс IFileSystemNode. Класс предоставляет методы для создания, удаления, переименования, чтения, редактирования и управления содержимым файлов. Основные элементы класса:

1. Ассоциация с BrowserDirectory:
   Свойство FileDirectory связывает файл с его директорией, в которой он находится.
2. Содержимое файла:
   - Поле currentFileContents хранит текущее содержимое файла.
   - Свойство FileContents предоставляет доступ к содержимому, загружая его из файла, если оно еще не инициализировано.
3. Конструкторы:
   - Конструктор по умолчанию.
   - Конструкторы с параметрами name и path, позволяющие создавать объект с заданными характеристиками.
4. Переопределенные методы из BrowserFileSystemItem:
   - Create() — создает файл, а при необходимости и его директорию.
   - Delete() — удаляет файл из файловой системы.
   - Rename(string newName) — переименовывает файл, обновляя путь.
5. Методы редактирования:
   - EditFile(string content) — записывает указанное содержимое в файл.
   - EditFile(SaveFileDialog saveFileDialog, string content) — сохраняет файл в указанное пользователем место.
6. Методы чтения:
   - ReadFile() — возвращает содержимое файла как строку.
   - ReadFile(OpenFileDialog openFileDialog) — открывает файл, выбранный пользователем, и возвращает его содержимое.

3.4 Класс узла рабочего файла

На рис. 5 показан код класса BrowserTextFile – класса рабочего (текстового) файла файловой системы.

```
{
    public class BrowserTextFile : BrowserFile
    {
        private BrowserMetaFile linkedMetadataFile;

        public DateTime DateCreated { get; set; } // Свойство даты и времени

        private string fileTypeProperty;
        public string FileTypeProperty // Свойство с проверкой на первую прописную букву...

        public string fileTypeField;

        // Числовое свойство с ограниченным get (доступ только из класса)
        public int FileSize...

        private bool isReadOnly;
        public bool IsReadOnly...

        private string iconPhotoPath;
        public string IconPhotoPath { get; set; }

        private Graphics iconGraphics;

        public BindingList<string> Authors { get; set; } = new BindingList<string>();

        public BrowserTextFile(string name, string path) ...
        public BrowserTextFile(string name, string path, string fileType) ...
        public BrowserTextFile(string name, string path, string fileType, DateTime dateCreated) ...
        public BrowserTextFile(string name, string path, string fileType, DateTime dateCreated, bool isReadOnly) ...

        // Переопределяем метод Create для определения некоторых параметров текстового файла
        public override void Create()...

        public void ChangeTypeProperty(string fileType)...

        public void ChangeTypeField(string fileType)...

        public void ChangeDateCreated(DateTime dateCreated)...

        public void ChangeReadOnly(bool isReadOnly)...

        public void ShowPhoto(PictureBox pictureBox)...

        public void ShowPhoto(Form formBox)...

        public void ResetPhoto(PictureBox pictureBox)...

        public void ResetPhoto(Form formBox)...

        // Сохранение и чтение метаинформации
        public void SaveMetadata()...

        public void LoadMetadata()...

        public void DeleteMetadataFile()...

        // Метод изменения авторов с использованием массива
        public void ChangeAuthors(List<string> newAuthors)...

        // Метод с параметром по ссылке (ref)
        public void ModifyFileType(ref string type)...

        // Метод с двумя выходными параметрами (out)
        public void GetFileDetails(out string type, out DateTime creationDate)...

        // Метод с необязательным параметром
        public void AddAuthor(string authorName = "Неизвестный автор")...
    }
}
```

Рисунок 5

Класс BrowserTextFile представляет текстовый файл и расширяет функциональность класса BrowserFile. Он включает дополнительные свойства и методы для управления текстовыми файлами, их метаинформацией и ассоциированными данными, такими как авторы, тип файла и метаданные. Класс поддерживает взаимодействие с графическими элементами интерфейса, а также работу с метаинформацией через

композицию с классом BrowserMetaFile. Основные элементы класса:

1. Свойства:
    - DateCreated — дата и время создания файла.
    - FileTypeProperty — тип файла с проверкой корректности ввода (первая буква заглавная).
    - fileTypeField — строковое поле для хранения типа файла.
    - FileSize — размер файла (доступ на чтение только внутри класса).
    - IsReadOnly — флаг, указывающий на доступность файла для записи.
    - IconPhotoPath — путь к иконке, связанной с файлом.
    - Authors — список авторов файла, реализованный через BindingList.
2. Ассоциация и композиция:
    - Связь с классом BrowserMetaFile через объект linkedMetadataFile для управления метаинформацией.
3. Конструкторы:
    - Поддержка различных комбинаций параметров (name, path, fileType, dateCreated, isReadOnly) для создания объектов.
4. Переопределенный метод:
    - Create() — переопределен для автоматического определения даты создания, размера файла и статуса "только чтение".
5. Методы управления свойствами:
    - Изменение типа файла (ChangeTypeProperty, ChangeTypeField).
    - Изменение даты создания (ChangeDateCreated).
    - Изменение статуса "только чтение" (ChangeReadOnly).
6. Методы работы с иконками:
    - Отображение иконки на графических элементах (ShowPhoto, ResetPhoto).
7. Методы работы с метаинформацией:

- Сохранение метаданных (SaveMetadata).
- Загрузка метаданных (LoadMetadata).
- Удаление файла метаданных (DeleteMetadataFile).

8. Работа с авторами:
   - Изменение списка авторов (ChangeAuthors).
   - Добавление автора с необязательным параметром (AddAuthor).
9. Дополнительные методы:
   - Изменение типа файла через параметр по ссылке (ModifyFileType).
   - Получение деталей файла через параметры out (GetFileDetails).

3.5 Класс узла метафайла

На рис. 6 показан код класса BrowserMetaFile – класса файла-соседа с метаинформацией о рабочем файле.

```
{
    public class BrowserMetaFile : BrowserFile
    {
        private BrowserTextFile linkedTextFile;

        public BrowserMetaFile(string name, string path, BrowserTextFile linkedFile) : base(name, path)
        {
            linkedTextFile = linkedFile; // Агрегация BrowserMetaFile <-> BrowserTextFile (ссылка на имеющийся объект)
            linkedTextFile.PathChanged += LinkedTextFile_PathChanged; // Подписка на событие
        }

        // Обработчик события изменения пути текстового файла
        private void LinkedTextFile_PathChanged(object sender, BrowserFileSystemItemEventArgs e)
        {
            Debug.WriteLine($"Новый путь текстового файла: {e.NewPath}");
            this.Path = System.IO.Path.ChangeExtension(e.NewPath, ".meta"); // Меняем рабочий путь метафайла на соответсвующий новому
        }

        public void Write(List<string> authors, string fileType, DateTime dateCreated, bool isReadOnly, string IconPhotoPath)
        {
            using (StreamWriter writer = new StreamWriter(this.Path))
            {
                writer.WriteLine(string.Join(",", authors));
                writer.WriteLine(fileType);
                writer.WriteLine(dateCreated);
                writer.WriteLine(isReadOnly);
                writer.WriteLine(IconPhotoPath);
            }
        }

        public void Read()
        {
            if (!File.Exists(this.Path))
            {
                linkedTextFile.SaveMetadata();
                Read();
            }
            else
            {
                using (StreamReader reader = new StreamReader(this.Path))
                {
                    // Здесь, поскольку BindingList является "оберткой" для передаваемого при его инициализации списка,
                    // нужно использовать копию списка, чтобы избежать ошибки "Read-Only List" при попытке изменения BindingList
                    BindingList<string> parsedAuthors = new BindingList<string>(reader.ReadLine().Split(',').ToList());
                    if (parsedAuthors[0] != "") linkedTextFile.Authors = parsedAuthors;

                    linkedTextFile.FileTypeProperty = reader.ReadLine();
                    linkedTextFile.DateCreated = DateTime.Parse(reader.ReadLine());
                    linkedTextFile.IsReadOnly = bool.Parse(reader.ReadLine());
                    linkedTextFile.IconPhotoPath = reader.ReadLine();
                }
            }
        }
    }
}
```

Рисунок 6

В классе используется агрегация (один объект включает в себя ссылку на другой объект). Также имеется обработчик событий, который реагирует на изменения пути текстового файла и корректирует путь метафайла. Основные элементы класса:

1. Поля:
    - linkedTextFile: Экземпляр класса BrowserTextFile, который ассоциируется с метафайлом.
2. Конструктор:
    - Принимает параметры name, path, и linkedFile, где linkedFile — это объект типа BrowserTextFile, с которым осуществляется агрегация.
    - В конструкторе происходит подписка на событие изменения пути у связанного текстового файла.
3. Событие:
    - PathChanged: Событие, которое обрабатывается в методе LinkedTextFile_PathChanged, обновляя путь метафайла при изменении пути текстового файла.
4. Методы:
    - Write: Записывает метаданные в файл, включая авторов, тип файла, дату создания, флаг доступности для редактирования и путь к иконке.
    - Read: Считывает данные из метафайла и обновляет связанные параметры объекта BrowserTextFile. Если файл не существует, вызывает метод SaveMetadata у связанного текстового файла.

4 Тестирование функциональности классов

На протяжении всего времени разработки проводилось тестирование разрабатываемой библиотеки с помощью приложения, написанного на WinForms.

На рис. 7 показана структура рабочей папки программы при первом запуске, в которой находятся файлы метаданных.

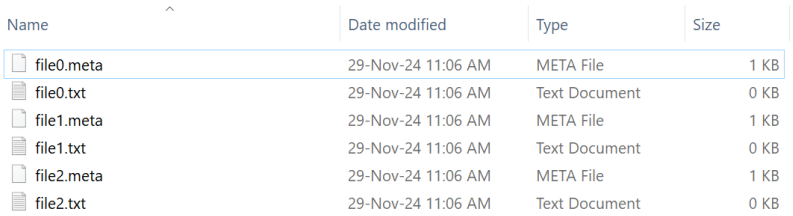


Рисунок 7

На рис. 8 показано окно тестирования метода ModifyFileType с ссылочным параметром ref.

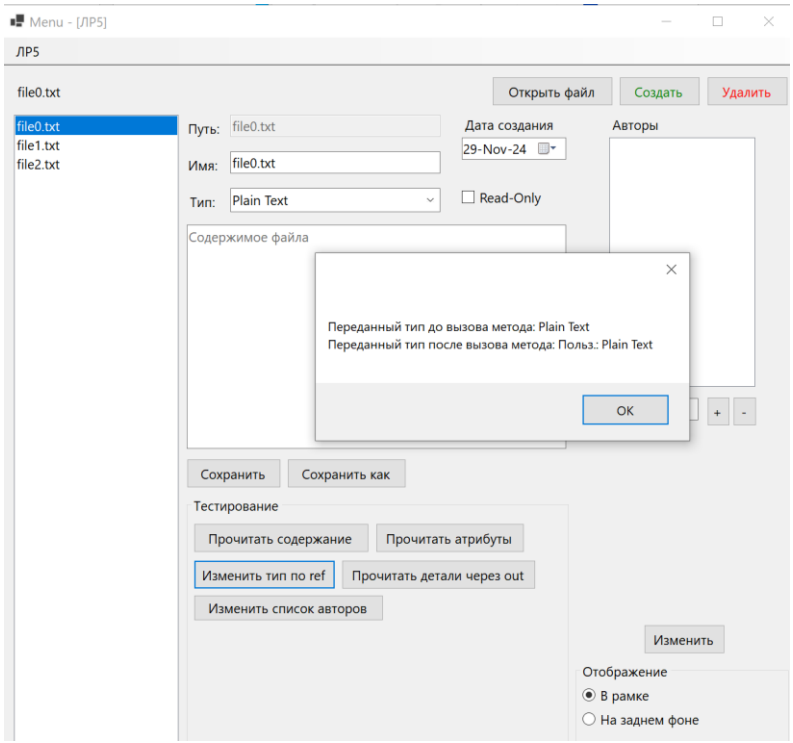


Рисунок 8

На рис. 9 показано окно тестирования метода GetFileDetails с двумя выходными параметрами out.

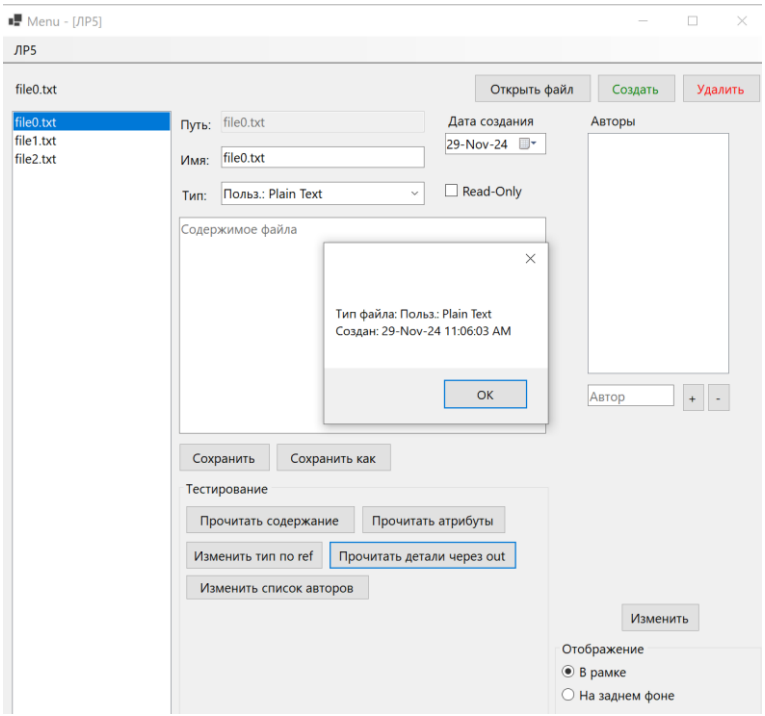


Рисунок 9

На рис. 10 изображено окно тестирования метода ChangeAuthors, изменяющего передаваемый в него список ссылочно.

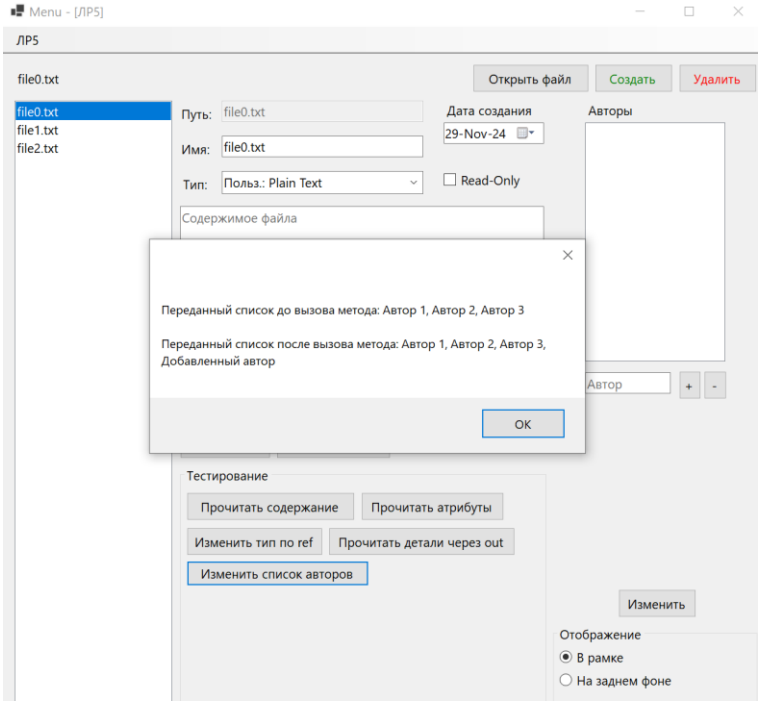


Рисунок 10

На рис. 11 показано окно вызова метода тестирования объектов

производных классов и их UpCasting, DownCasting преобразований. Выведенная информация говорит о том, что BrowserFile, как и все остальные классы неявно происходят от класса Object. UpCasting - объект производного класса BrowserFile представляет объект fsItem1 класса BrowserFileSystemItem, однако объекту fsItem1 недоступны параметры объекта browserFile. DownCasting - явно преобразуем объект fsItem1 к объекту производного класса BrowserFile и тем самым получаем доступ к параметрам объекта browserFile.

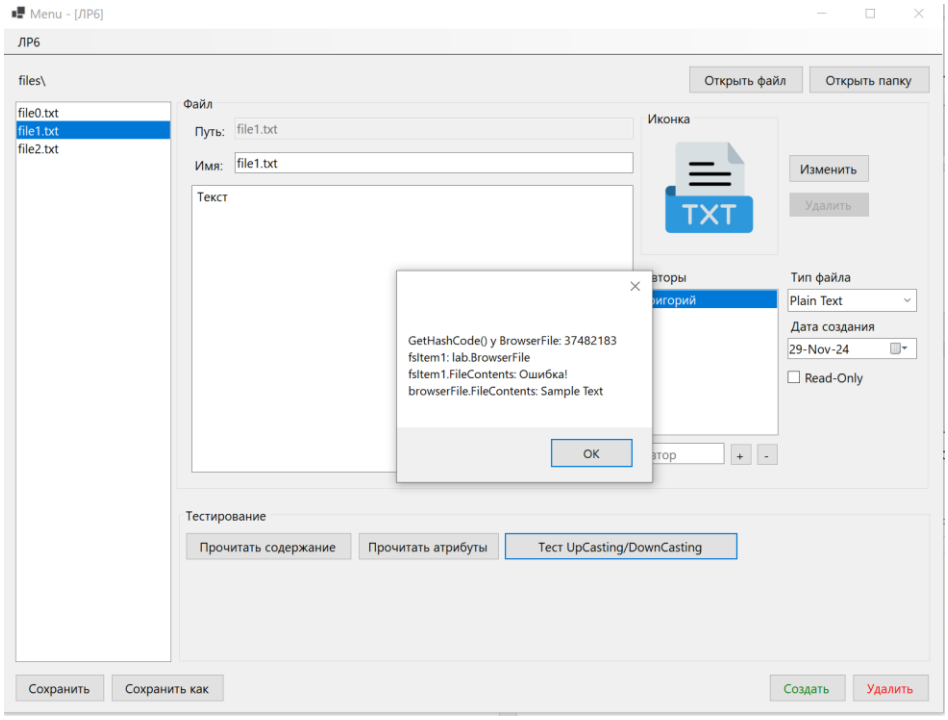


Рисунок 11

На рис. 12 изображен код метода тестирования, описанного выше.

```
private void buttonTestCasting_Click(object sender, EventArgs e)
{
    // UpCasting - объект BrowserFile (как и объект любого другого класса) представляет объект object
    object obj = new BrowserFile();

    // DownCasting - явно преобразуем объект object к объекту производного класса BrowserFileSystemItem
    BrowserFileSystemItem fsItem = (BrowserFileSystemItem)obj;

    // UpCasting - объект производного класса BrowserFile представляет объект BrowserFileSystemItem
    BrowserFileSystemItem fsItem1 = new BrowserFile();
    //fsItem1.FileContents = "Sample Text"; //- Ошибка, так как объекту BrowserFileSystemItem недоступны параметры объекта BrowserFile

    // DownCasting - явно преобразуем объект BrowserFileSystemItem к объекту производного класса BrowserFile
    BrowserFile browserFile = (BrowserFile)fsItem1;
    browserFile.FileContents = "Sample Text"; // Ошибки нет

    string content = $"GetHashCode() у BrowserFile: {browserFile.GetHashCode()}\n" +
                     $"fsItem1: {fsItem1.GetType()}\n" +
                     $"fsItem1.FileContents: Ошибка!\n" +
                     $"browserFile.FileContents: {browserFile.FileContents}";
    MessageBox.Show(content);
}
```

Рисунок 12

На рис. 13 показана рабочая среда при взаимодействии с программой: протестирован функционал переименования, изменения содержания и метаинформации файла.

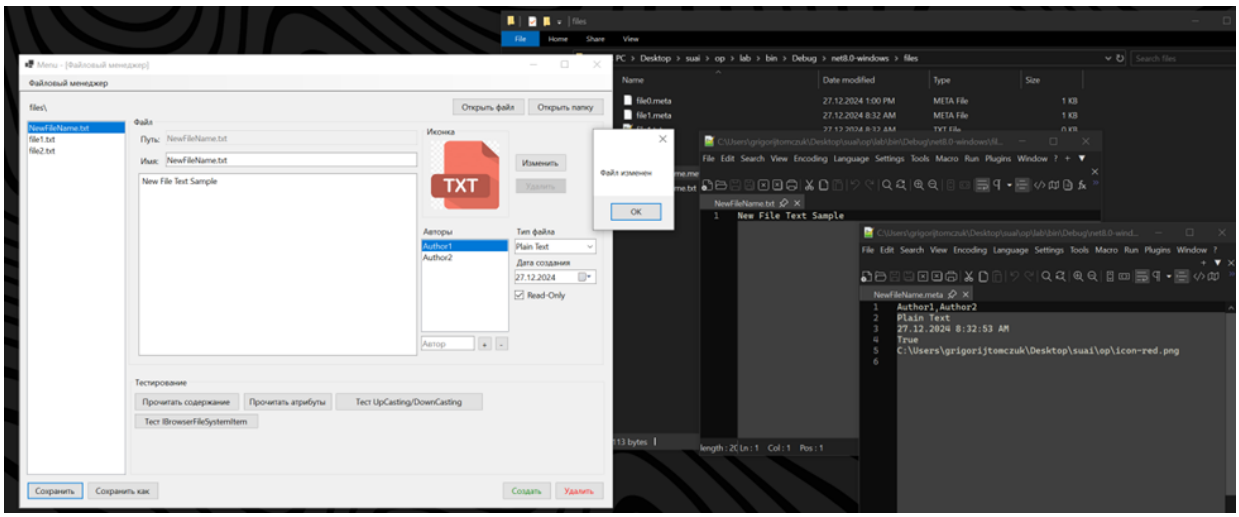


Рисунок 13

5 Пример использования библиотеки для построения интерфейса прикладного решения

С помощью разработанной библиотеки можно создавать, редактировать и сохранять текстовые файлы, а также генерировать метафайлы, связанные с ними. Чтобы предоставить пользователю удобный интерфейс для взаимодействия с этой библиотекой, я создал десктоп-приложение на Windows Forms.

Основные функциональные возможности приложения:

– Создание и редактирование текстовых файлов: Пользователь может создавать новые текстовые файлы, редактировать их содержимое и сохранять изменения.

– Создание метафайлов: Для каждого текстового файла создается метафайл, который хранит метаданные, такие как авторы, тип файла, дата создания и т. д.

На рис. 14, 15, 16 изображены скриншоты интерфейса программы в работе.

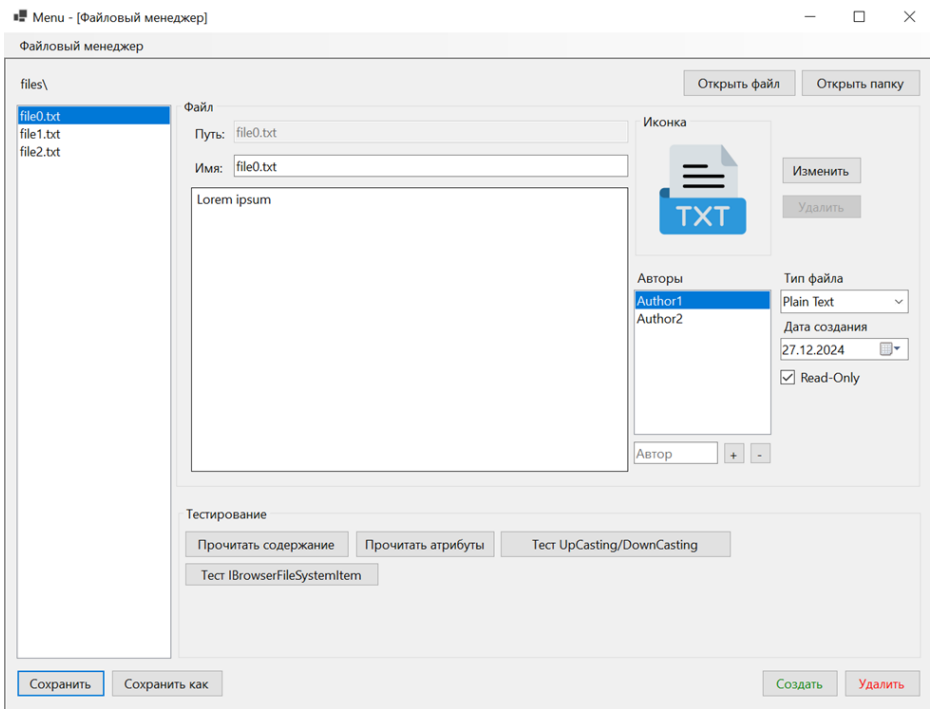


Рисунок 14

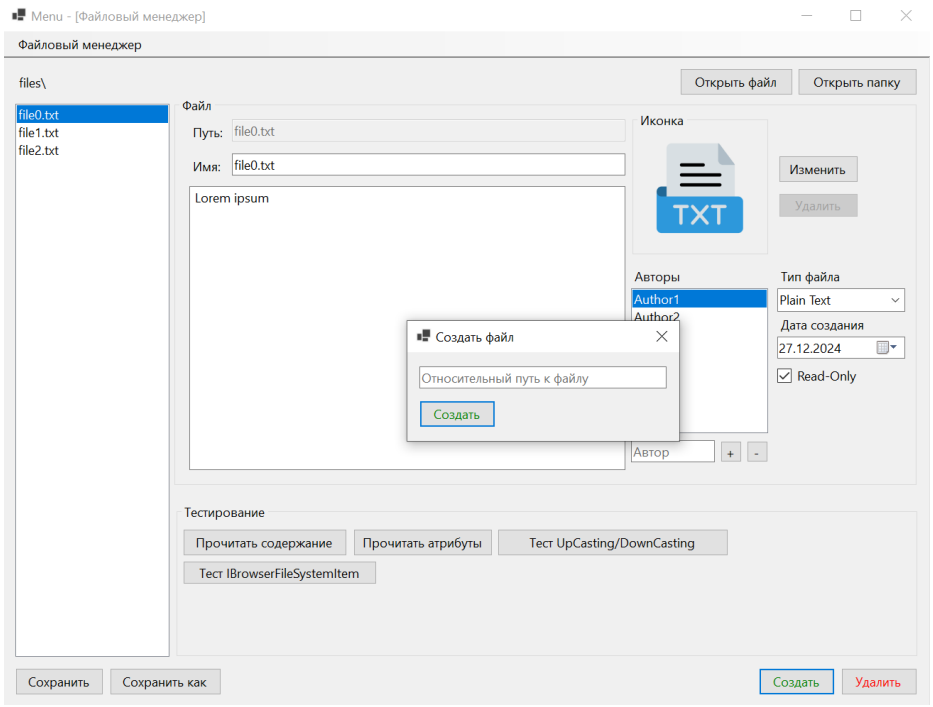


Рисунок 15

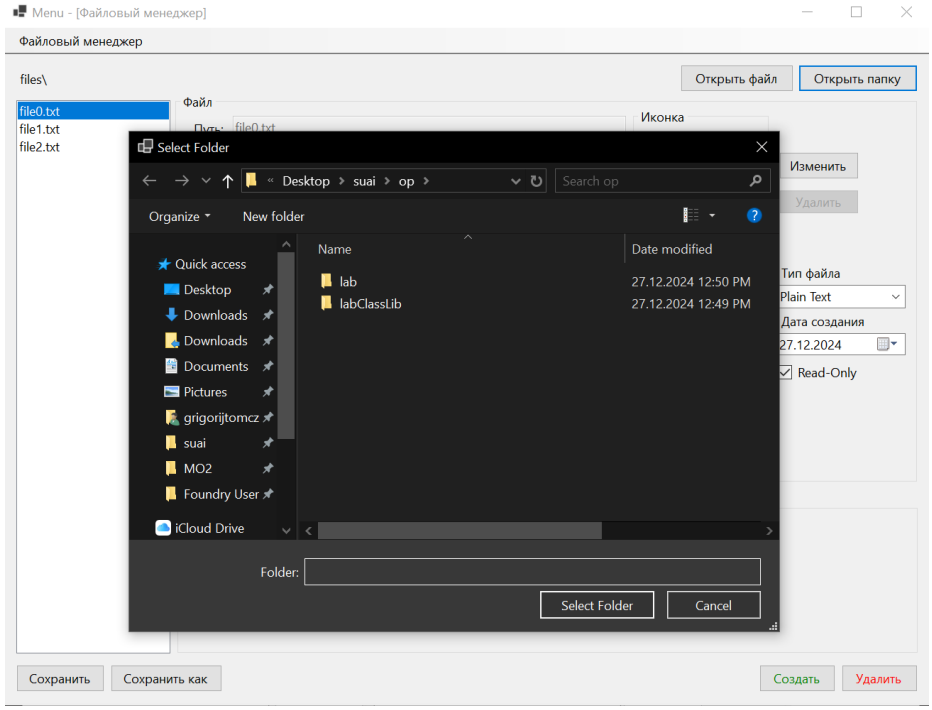


Рисунок 16

6 Руководство пользователя-программиста при использовании созданной библиотеки как ресурса

Библиотека предназначена для работы с файловой системой в контексте объектно-ориентированного программирования и предоставляет пользователю набор классов для управления файлами и папками, их создания, удаления, переименования и изменения метаинформации. Рассмотрим, как использовать основные классы и их функциональность.

1. Основные классы библиотеки

- BrowserFileSystemItem: Абстрактный класс, который является основой для всех элементов файловой системы, таких как файлы и папки. Он реализует основные свойства, такие как имя, путь и дату создания. Также включает события изменения пути и методы для переименования и перемещения элементов.
- BrowserDirectory: Наследует `BrowserFileSystemItem` и представляет директории файловой системы. Этот класс предоставляет методы для создания, удаления и переименования директорий.
- BrowserFile: Наследует `BrowserFileSystemItem` и представляет файлы в файловой системе. Включает методы для создания, удаления и редактирования файлов, а также чтения и записи их содержимого.
- BrowserTextFile: Наследует `BrowserFile` и добавляет дополнительные свойства и методы, связанные с текстовыми файлами, такие как управление метаинформацией, авторами и типом файла, а также изменение фотографии и сохранение метаданных.
- BrowserMetaFile: Наследует `BrowserFile` и служит для работы с метаинформацией текстового файла. Этот класс отслеживает изменения пути текстового файла и обновляет путь метафайла, сохраняет и читает метаданные.

2. Основные функциональные возможности

2.1 Работа с файлами и директориями

- Создание файла/папки:

- Для создания папки используйте метод `Create` у объекта `BrowserDirectory`.
- Для создания файла используйте метод `Create` у объекта `BrowserFile`.

Пример:

```
BrowserDirectory dir = new
BrowserDirectory("NewDirectory", @"C:\MyDir");
dir.Create();
BrowserFile file = new BrowserFile("MyFile.txt",
@"C:\MyDir\MyFile.txt");
file.Create();
```

- Удаление файла/папки:
  - Для удаления папки используйте метод `Delete` у объекта `BrowserDirectory`.
  - Для удаления файла используйте метод `Delete` у объекта `BrowserFile`.

Пример:

```
dir.Delete();
file.Delete();
```

- Переименование файла/папки:
  - Для переименования используйте метод `Rename` у объекта `BrowserFile` или `BrowserDirectory`.

Пример:

```
file.Rename("NewFileName.txt");
dir.Rename("NewDirectoryName");
```

2.2 Работа с содержимым файлов

- Для чтения содержимого текстового файла используйте метод `ReadFile`:

```
string content = file.ReadFile();
```

- Для редактирования файла используйте метод `EditFile`:
  ```
  file.EditFile("New content");
  ```
- Если файл открылся через диалоговое окно, можно сохранить изменения с помощью метода:
  ```
  file.EditFile(saveFileDialog, "New content");
  ```

2.3 Метафайлы и метаинформация

- Для управления метаинформацией, используйте `BrowserMetaFile`. Это позволит сохранять и загружать данные о файле, такие как авторы, тип файла, дата создания и фото. Пример:
  ```
  BrowserTextFile textFile = new
  BrowserTextFile("TextFile.txt",
  @"C:\MyDir\TextFile.txt");
  textFile.SaveMetadata();
  textFile.LoadMetadata();
  ```
- Метаданные могут быть изменены с помощью методов, например, `ChangeTypeProperty` для изменения типа файла:
  ```
  textFile.ChangeTypeProperty("NewType");
  ```

2.4 Работа с авторами и изображениями

- Для работы с авторами текста используйте коллекцию `Authors` в `BrowserTextFile`:
  ```
  textFile.AddAuthor("New Author");
  ```
- Для отображения фотографии файла используйте методы `ShowPhoto` и `ResetPhoto`:
  ```
  textFile.ShowPhoto(pictureBox);
  textFile.ResetPhoto(pictureBox);
  ```

2.5 Обработка событий

- Класс `BrowserFileSystemItem` поддерживает событие `PathChanged`, которое вызывается при изменении пути

элемента. Вы можете подписаться на это событие для отслеживания изменений. Пример:

```
file.PathChanged += (sender, e) =>
Console.WriteLine("Путь изменен на: " +
e.NewPath);
```

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта была достигнута основная цель — разработка пользовательской библиотеки классов для работы с текстовыми файлами и создание программного приложения «Файловый менеджер» на платформе C# с использованием Windows Forms. Эта задача позволила мне применить теоретические знания и практические навыки, полученные в ходе обучения, а также углубить понимание принципов объектно-ориентированного программирования, работы с файловыми системами и разработки пользовательских интерфейсов.

Одной из важнейших задач было создание библиотеки, которая учитывает особенности работы с файловыми узлами — текстовыми файлами, метафайлами и папками, а также реализует необходимые функции для их обработки, записи и чтения данных. В процессе работы была применена концепция отношений объектов, событий, а также методы взаимодействия классов, что позволило разработать гибкую и масштабируемую архитектуру. Кроме того, приложение было оснащено функционалом для удобного взаимодействия с пользователем через графический интерфейс, что сделало его доступным и интуитивно понятным.

Работа над курсовым проектом позволила мне систематизировать теоретические знания, полученные в ходе дисциплины «Основы программирования». Я смог закрепить на практике основные концепции объектно-ориентированного подхода, такие как инкапсуляция, наследование и полиморфизм, а также использовать их для решения конкретных задач в контексте разработки программного обеспечения для работы с файловыми системами. Процесс создания библиотеки классов и интерфейса пользователя также способствовал улучшению навыков работы с учебной литературой и источниками информации, необходимыми для разработки программных систем.

Одним из значительных достижений стало развитие умений системного мышления и творческой инициативы при проектировании и реализации

функционала приложения. В ходе работы над проектом мне удалось развить навыки планомерной работы, а также научиться более эффективно организовывать и распределять задачи для достижения конечной цели. Параллельно с этим была развита профессиональная письменная и устная речь, что стало особенно актуальным при подготовке документации, описания кода и презентовании результатов.

Решение поставленных задач также позволило мне развить самостоятельность в принятии решений, ответственности за их реализацию и понимание важности четкого соблюдения требований к функционалу и структуре создаваемого программного обеспечения. Процесс создания и отладки библиотеки и приложения в целом подтвердил важность дисциплины и ответственности при разработке программных продуктов, а также необходимость внимательного подхода к каждому этапу работы.

В заключение можно сказать, что курсовая работа позволила значительно углубить мои знания в области программирования, а также предоставила ценный опыт в разработке сложных программных решений с использованием современных технологий и подходов, что является неотъемлемой частью профессиональной подготовки в области информационных систем и технологий.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Грей, Дж. Программирование на C# / Дж. Грей. — СПб.: БХВ-Петербург, 2016. — 528 с.
2. Ричардс, Д. Программирование на C# 7 / Д. Ричардс. — М.: ДМК Пресс, 2018. — 672 с
3. Петров, А. Разработка Windows Forms приложений на C# / А. Петров. — М.: Бином, 2014. — 352 с.
4. Шмидт, Л. Основы объектно-ориентированного программирования / Л. Шмидт. — М.: Наука, 2015. — 320 с.
5. Методические рекомендации по проектированию и разработке прикладных программных систем / С. Кузнецов. — М.: Физматлит, 2015. — 280 с.
6. Джонсон, Б. Архитектура программного обеспечения: объектно-ориентированные подходы / Б. Джонсон. — М.: Радио и связь, 2016. — 412 с.