# **Федеральное государственное бюджетное профессиональное образовательное учреждение**

**Торжокский политехнический колледж**

# **Федерального агентства по государственным резервам**

**ОТЧЁТ**

# о зимней работе

**по профессиональному модулю МДК.01.01**

«**РАЗРАБОТКА ПРОГРАМНЫХ МОДУЛЕЙ**»

наименование профессионального модуля

## Специальность ***09.02.07 «Информационные системы и программирование»***

Период с «28» декабря 2024г. по «08» февраля 2025г.

## Выполнил студент

Андреев А. М.



(ФИО, подпись)

| Преподаватель  Бойков А.М.    (ФИО, подпись) | оценка |
| --- | --- |

## 

2024

**СОДЕРЖАНИЕ**

Введение [3](https://docs.google.com/document/d/1HWeDTfHMxRBmSMBVK7UOflCPHa_DON7o7UZp3-8x5w0/edit#heading=h.wk52owbb64pu)

Раздел 1. Практическая значимость разрабатываемого программного

средства [4](https://docs.google.com/document/d/1HWeDTfHMxRBmSMBVK7UOflCPHa_DON7o7UZp3-8x5w0/edit#heading=h.syj7q3fre87a)

Раздел 2. Функциональные требования к ПС6

Раздел 3. Модульная структура проектируемого ПС10

Раздел 4. Описание алгоритмов15

Раздел 5. Разработка проекта Visual Studio C# [1](https://docs.google.com/document/d/1HWeDTfHMxRBmSMBVK7UOflCPHa_DON7o7UZp3-8x5w0/edit#heading=h.guvinl34bc8j)19

**ВВЕДЕНИЕ**

Цель работы – получить практический опыт парного программирования и разработки модульной структуры программы.

Задачи:

1) выявить практическую значимость проектируемого программного средства (далее – ПС) (с точки зрения использования, с точки зрения развития продуктов большего масштаба);

2) обосновать функции разрабатываемого программного средства и варианты его использования (какие задачи позволит решить и кому);

3) спроектировать модульную структуру программного средства (структура классов: данные и методы);

4) описание нескольких алгоритмов методов (как минимум – два), отвечающие непосредственно за решение поставленной задачи;

5) разработать программное средство в среде Visual Studio.

**Раздел 1. Практическая значимость разрабатываемого программного средства**

Разрабатываемое программное средство (ПС) — CSV-Viewer — предназначено для удобного преобразования данных из CSV-файлов в HTML-таблицу, обеспечивая визуально понятное представление информации.

CSV-файлы широко используются для хранения и обмена табличными данными, однако их чтение в исходном текстовом виде затруднительно. Разработка данного ПС позволяет автоматически формировать структурированное представление данных в виде HTML-таблицы, обеспечивая удобное форматирование, проверку ошибок и выбор визуального оформления.

Основными пользователями могут быть разработчики, аналитики, офисные работники и все, кто работает с CSV-данными и нуждается в их удобном отображении. Программа будет полезна как для личного использования, так и для применения в организациях, где необходимо преобразовывать CSV-данные в читаемый формат.

Данное ПС может применяться в различных сценариях:

* автоматизация отчетов и анализа данных: например, в компании регулярно формируются CSV-отчеты о продажах. Без CSV-Viewer сотрудникам приходится вручную загружать файлы в Excel, настраивать стили и преобразовывать данные в HTML для отчётности. С помощью разработанного ПС можно быстро создать таблицу в нужном формате и визуальном стиле без лишних действий;
* использование в качестве программного модуля на веб-ресурсах: CSV-Viewer может быть использован на сайтах, сервисах и блогах для автоматического представления табличных данных. Например, финансовые ресурсы могут динамически отображать биржевые котировки, интернет-магазины — обновляемые списки товаров, а блоги — сравнительные таблицы характеристик товаров или услуг;
* публикация результатов опросов: допустим, организация проводит опрос среди сотрудников или клиентов, собирая данные в CSV-файл. С помощью программы можно легко преобразовать эти данные в наглядную HTML-таблицу и разместить на сайте или внутреннем портале, чтобы результаты были удобны для просмотра.

Если бы не было данного программного средства, то пользователям пришлось бы либо вручную редактировать CSV-файлы в текстовых редакторах, что неудобно и может привести к ошибкам, либо использовать сложные табличные процессоры, которые не всегда подходят для простого и быстрого преобразования данных в HTML-формат. Также без данного ПС отсутствовала бы возможность автоматизированной проверки корректности CSV-файлов перед их конвертацией.

**Раздел 2. Функциональные требования к ПС**

Программное средство CSV-Viewer должно обеспечивать выполнение следующих функций:

1. Проверка исходного файла: пользователь вводит путь к CSV-файлу через консоль; программа проверяет корректность пути и формата файла, анализирует структуру данных. Реакция программы на возможные ошибки:

* путь не был введён: выводит сообщение об ошибке — «Путь к CSV-файлу не может быть пустым.»;
* некорректный путь к CSV-файлу: выводит сообщение об ошибке — «CSV-файл не найден. Пожалуйста, укажите корректный путь к CSV-файлу.»;
* CSV-файл пустой: выводит сообщение об ошибке — «CSV-файл не содержит данных.»;
* пустые значения в строке заголовка: выводит сообщение об ошибке — «Строка заголовка содержит пустые значения.»;
* пустая строка в CSV-файле или пустые значения в строке данных: программа информирует об этом пользователя, указав номер некорректной строки; программа пропускает некорректную строку, продолжая свою работу;
* неверное количество столбцов в строке данных: информирует пользователя о номере некорректной строки и количествах найденных и ожидаемых столбцов;
* отсутствие строк с данными в CSV-файле: выводит сообщение об ошибке — «В CSV-файле отсутствуют строки с данными.».

В качестве разделителя данных выступает запятая («,»). Если требуется включить запятую в содержимое ячейки, не используя ее в качестве разделителя, весь столбец необходимо заключить в двойные кавычки (“...”).

Пример корректного исходного файла:

Id,Имя,Возраст

1,Артём,25

2,”Валерия, Лера”,30

Пример некорректного исходного файла:

Id,Имя,,

1;Артём

2,Валерия,30,438

1. Генерация HTML-таблицы: преобразование содержимого CSV-файла в HTML-таблицу; корректное отображение данных, включая специальные символы; возможность выбора темы оформления.

Пример исходного файла:

Id,Имя,Возраст

1,Артём,25

2,Валерия,30

Пример сгенерированной HTML-таблицы представлен на рисунке 1:

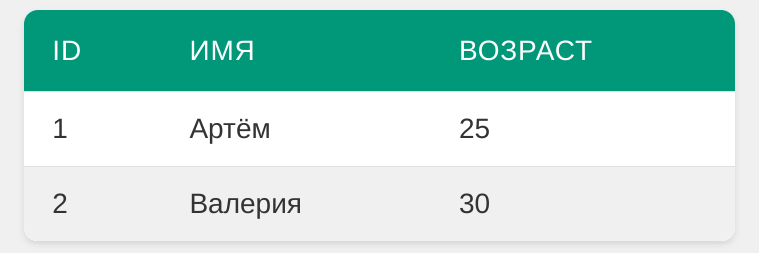


Рисунок 1 – Пример сгенерированной HTML-таблицы.

1. Выбор темы оформления и шрифта: пользователь через консоль выбирает одни из доступных тем и шрифтов; программа применяет выбранное оформление к итоговому HTML-файлу.

Доступные темы оформления HTML-таблицы: светлая (light), темная (dark), зеленая (green).

Доступные варианты шрифтов: Arial, Courier New, Times New Roman.

Способ указания темы и шрифта: программа просит пользователя ввести одну из тем и шрифтов, предоставляя доступные варианты и их номера; в квадратных скобках программа указывает номер варианта по умолчанию, который будет использоваться, если пользователь не станет вводить значение (нажав клавишу «Enter», можно пропустить ввод значения); пользователь по желанию вводит номер темы и шрифта через консоль.

При вводе значения, отличного от корректного номера (1, 2 или 3) темы или шрифта, программа выводит сообщение об ошибке о том, что данной темы или шрифта не существует.

Примеры различных стилей оформления HTML-таблицы:

1. Тема - светлая (light), шрифт - Courier New:

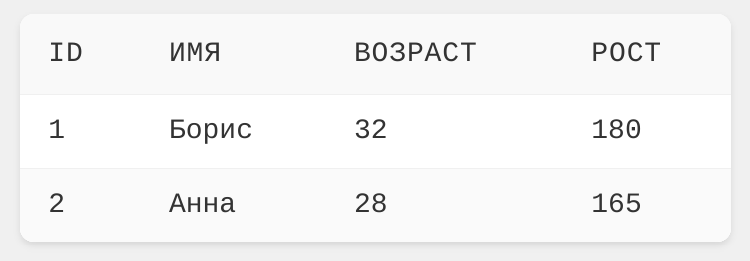


Рисунок 2 – Светлый стиль оформления HTML-таблицы.

1. Тема - темная (dark), шрифт - Arial:



Рисунок 3 – Тёмный стиль оформления HTML-таблицы.

1. Тема - зеленая (green), шрифт - Times New Roman:

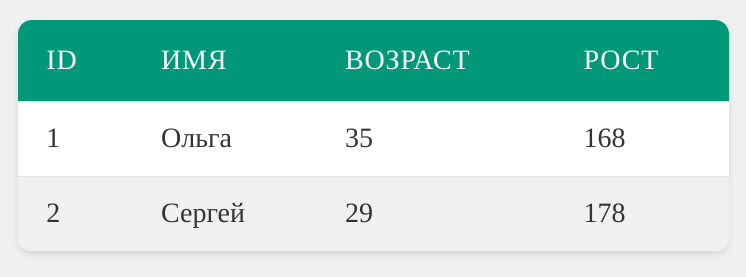


Рисунок 4 – Зеленый стиль оформления HTML-таблицы.

1. Сохранение и просмотр результата: итоговый HTML-файл сохраняется в указанной пользователем директории; возможность посмотреть сгенерированный файл, открыв его в браузере.

Пользователь указывает абсолютный или относительный (относительно папки, в которой запускается программа) путь до директории, в которую желает сохранить итоговый файл (например, C:\Users\Alex\Desktop\).

Программа проверяет введенный путь к директории:

* если директория существует, программа сохраняет в неё итоговый файл, выводя сообщение от успехе, указав полный путь до сгенерированного файла;
* в случае отсутствия введенной директории программа выводит сообщение об ошибке о том, что указанной директории не существует;
* если отсутствуют права на запись в указанную директорию, программа выводит сообщение об ошибке, указав полный путь до файла, который не удалось записать.

**Раздел 3. Модульная структура проектируемого ПС**

Диаграмма классов представлена на Рисунке 5:

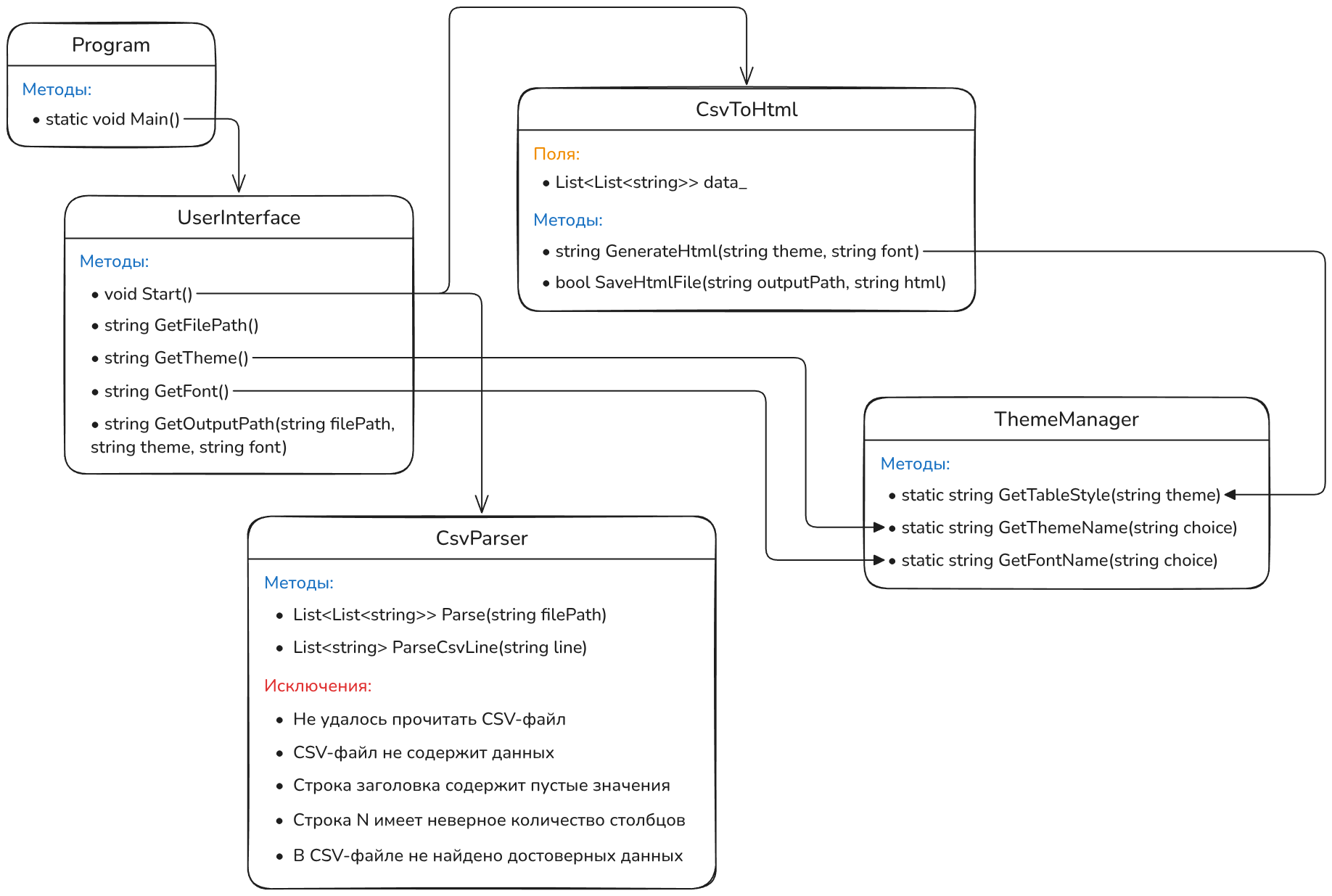
****

Рисунок 5 – Диаграмма классов программы.

Проектируемое программное средство (ПС) состоит из нескольких модулей, представленных классами, каждый из которых выполняет строго определённую функцию. Такая модульная структура повышает читаемость кода, облегчает его поддержку и расширение.

3.1. Перечень классов и их назначение

Таблица 1 – список классов и их назначений.

| Класс | Назначение |
| --- | --- |
| Program | Точка входа в программу. |
| UserInterface | Управляет взаимодействием с пользователем, запрашивает входные данные, вызывает обработку CSV-файла и сохранение HTML. |
| CsvParser | Отвечает за чтение и разбор CSV-файла, проверку структуры и обработку ошибок. |
| CsvToHtml | Генерирует HTML-таблицу из обработанных данных. |
| ThemeManager | Обрабатывает выбор темы и шрифта, предоставляет стили оформления. |

3.2. Описание классов, их данных и методов

1. Program (главный класс, точка входа)

Класс содержит один метод:

* static void Main() – создает объект UserInterface и запускает процесс выполнения программы; возвращаемое значение отсутствует (void).

1. UserInterface (работа с пользователем)

Класс отвечает за ввод и вывод информации, а также за управление основными этапами работы программы.

Данные: нет постоянных данных, вся информация запрашивается и передаётся по мере выполнения.

Методы:

* void Start() – основной метод, запускающий выполнение программы; возвращаемое значение отсутствует (void);
* string GetFilePath() – запрашивает у пользователя путь к CSV-файлу, проверяет его существование и возвращает его в виде строки;
* string GetTheme() – запрашивает у пользователя выбор темы оформления HTML; возвращает название темы в формате строки; возможные варианты тем: dark, light, green;
* string GetFont() – запрашивает у пользователя выбор шрифта; возвращает название шрифта в формате строки; возможные варианты шрифтов: Courier New, Arial, Times New Roman;
* string GetOutputPath(string filePath, string theme, string font) – запрашивает у пользователя путь для сохранения HTML-файла и формирует уникальное название файла, основываясь на пути к исходному файлу, теме и шрифте; возвращает путь к выходному файлу в виде строки.

Этот класс необходим для организации взаимодействия с пользователем. Без него программа не смогла бы принимать вводимые данные и передавать их на обработку.

1. CsvParser (чтение и обработка CSV-файла)

Класс отвечает за загрузку данных из CSV и их валидацию.

Данные: нет постоянных данных, работа идёт с временными списками строк.

Методы:

* List<List<string>> Parse(string filePath) – читает и разбирает CSV-файл, возвращает данные в виде списка строковых списков; каждая внутренняя коллекция — это строка CSV, разбитая на ячейки;
* List<string> ParseCsvLine(string line) – разбирает одну строку CSV, учитывая кавычки; возвращает список строковых значений (ячейки одной строки таблицы);

Класс необходим для структурированной обработки CSV-файлов. Он содержит всю логику валидации, предотвращая появление ошибок на этапе генерации HTML.

Исключения:

* не удалось прочитать CSV-файл;
* CSV-файл не содержит данных;
* строка заголовка содержит пустые значения;
* строка N имеет неверное количество столбцов;
* в CSV-файле не найдено достоверных данных.

Выбор типов данных:

List<List<string>> – используется для представления данных, где каждая вложенная коллекция представляет строку CSV. Это позволяет легко перебирать данные при генерации HTML.

1. CsvToHtml (генерация HTML-таблицы)

Этот класс преобразует полученные данные в HTML-код.

Данные:

* \_data (List<List<string>>) – хранит разобранные данные CSV-файла; заполняется через конструктор CsvToHtml(List<List<string>> data).

Методы:

* string GenerateHtml(string theme, string font) – создаёт HTML-таблицу с указанными стилями; возвращает строку с HTML-кодом таблицы;
* bool SaveHtmlFile(string outputPath, string html) – сохраняет сгенерированную HTML-таблицу в файл; возвращает true, если файл сохранен успешно, иначе false.

Этот класс изолирует логику генерации HTML от других частей программы.

1. ThemeManager (настройки оформления)

Класс отвечает за выбор темы и шрифта.

Методы:

* string GetTableStyle(string theme) – возвращает CSS-стиль в виде строки для указанной темы.
* string GetThemeName(string choice) – преобразует ввод пользователя в название темы; возвращает название темы;
* string GetFontName(string choice) – преобразует ввод пользователя в название шрифта; возвращает название шрифта.

Этот класс позволяет легко расширять и модифицировать систему оформления HTML-файла без изменения других частей программы.

3.3. Взаимодействие классов

Последовательность работы программы:

1. UserInterface запрашивает у пользователя путь к CSV-файлу;
2. CsvParser читает файл, проверяет структуру и возвращает данные;
3. UserInterface запрашивает тему и шрифт у пользователя.
4. CsvToHtml создаёт HTML-таблицу;
5. UserInterface сохраняет HTML-файл в указанную папку.

3.4. Порядок использования классов и методов

1. UserInterface.Start():

* вызывает GetFilePath(), получает путь к CSV-файлу;
* передаёт путь в CsvParser.Parse(), получает List<List<string>>;
* запрашивает у пользователя тему (GetTheme()) и шрифт (GetFont());
* запрашивает путь для сохранения (GetOutputPath()).

1. CsvToHtml.GenerateHtml(theme, font): преобразует данные в HTML-таблицу.
2. CsvToHtml.SaveHtmlFile(outputPath, html): записывает HTML-файл в указанное пользователем место.

Программа структурирована по принципу разделения ответственности. Каждый класс выполняет конкретную задачу, что делает код удобным для поддержки и расширения.

**Раздел 4. Описание алгоритмов**

Функция генерации HTML-таблицы (GenerateHtml).

#### Текстовое описание шагов алгоритма:

1. Получить строку CSS-стилей таблицы из класса ThemeManager на основе заданной темы theme.
2. Сформировать строку CSS-шрифта на основе значения font.
3. Создать начальную часть HTML-документа: открывающий тег <html>; <head> с meta-тегом и стилями оформления таблицы; <body> с контейнером <div> и таблицей <table> с примененным шрифтом.
4. Добавить строку <thead> и цикл для заголовков: начать строку заголовков: <tr>; для каждой ячейки из первой строки данных (\_data[0]) добавить элемент <th>{значение}</th>; закрыть строку заголовков </tr> и тег <thead>.
5. Добавить тело таблицы <tbody>. Для каждой строки данных, начиная со второй (\_data.Skip(1)): открыть строку таблицы <tr>; для каждой ячейки строки добавить элемент <td><span>{значение}</span></td>; закрыть строку </tr>.
6. Закрыть тег <tbody>, затем таблицу </table>, контейнер </div>, </body> и весь HTML-документ </html>.
7. Вернуть итоговую строку html.

Блок схема метода GenerateHtml представлена на рисунке 6:

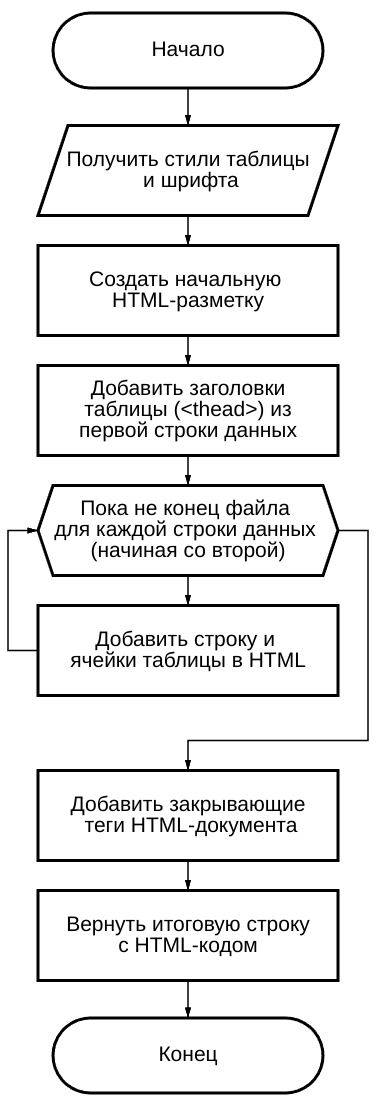


Рисунок 6 – блок-схема метода генерации HTML-таблицы.

Функция парсинга данных из CSV-файла (Parse).

Текстовое описание шагов алгоритма:

1. Принять путь к CSV-файлу как входной параметр.
2. Попробовать прочитать все строки из файла: если чтение не удалось — выбросить исключение: «Не удалось прочитать CSV-файл».
3. Проверить, содержит ли файл хотя бы одну строку: если нет — выбросить исключение: «CSV-файл не содержит данных».
4. Создать список data (List<List<string>>) для хранения всех строк таблицы.
5. Создать переменную headerLength для хранения количества столбцов в заголовке.
6. Для каждой строки в файле:

* удалить пробелы по краям;
* если строка пустая — вывести предупреждение и перейти к следующей;
* разбить строку на ячейки с помощью ParseCsvLine;
* если это первая строка (заголовок): проверить, что в заголовке нет пустых значений, иначе выбросить исключение — «Строка заголовка содержит пустые значения»; сохранить количество столбцов как headerLength;
* если это строка данных: вывести предупреждение, если есть пустые ячейки; проверить, что количество столбцов совпадает с headerLength, иначе выбросить исключение — «Строка N имеет неверное количество столбцов»;
* добавить разобранную строку в data.

1. Проверить, что после заголовка остались строки с данными: если нет — выбросить исключение: «В CSV-файле не найдено достоверных данных».
2. Вернуть data как результат.

Блок схема метода Parse представлена на рисунке 7:

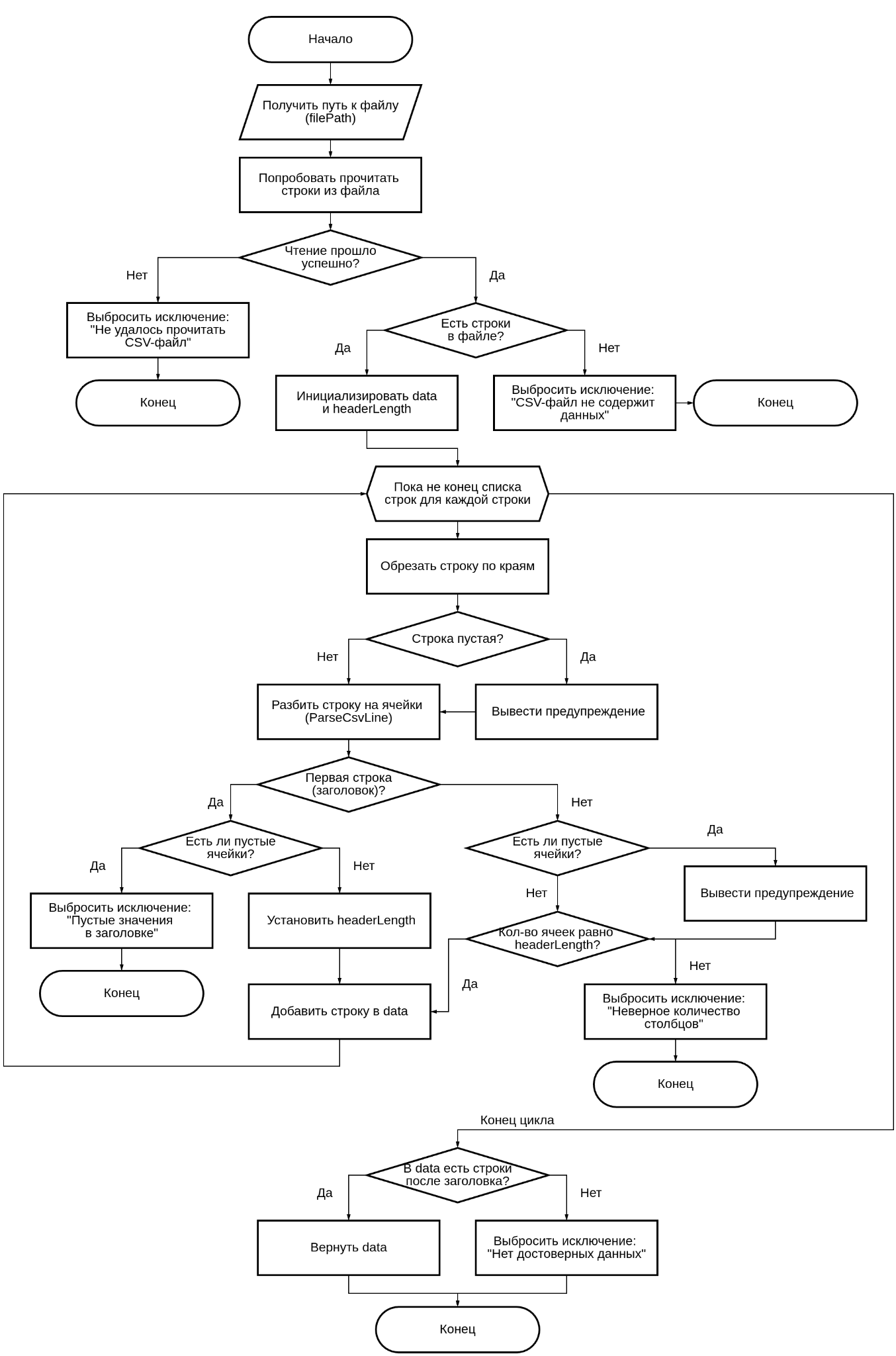


Рисунок 7 – блок-схема функции парсинга данных из CSV-файла.

**Раздел 5. Разработка проекта Visual Studio C#**

На рисунке 8 представлен класс Program:



Рисунок 8 – класс Program.

На рисунке 9 представлен класс ThemeManager:



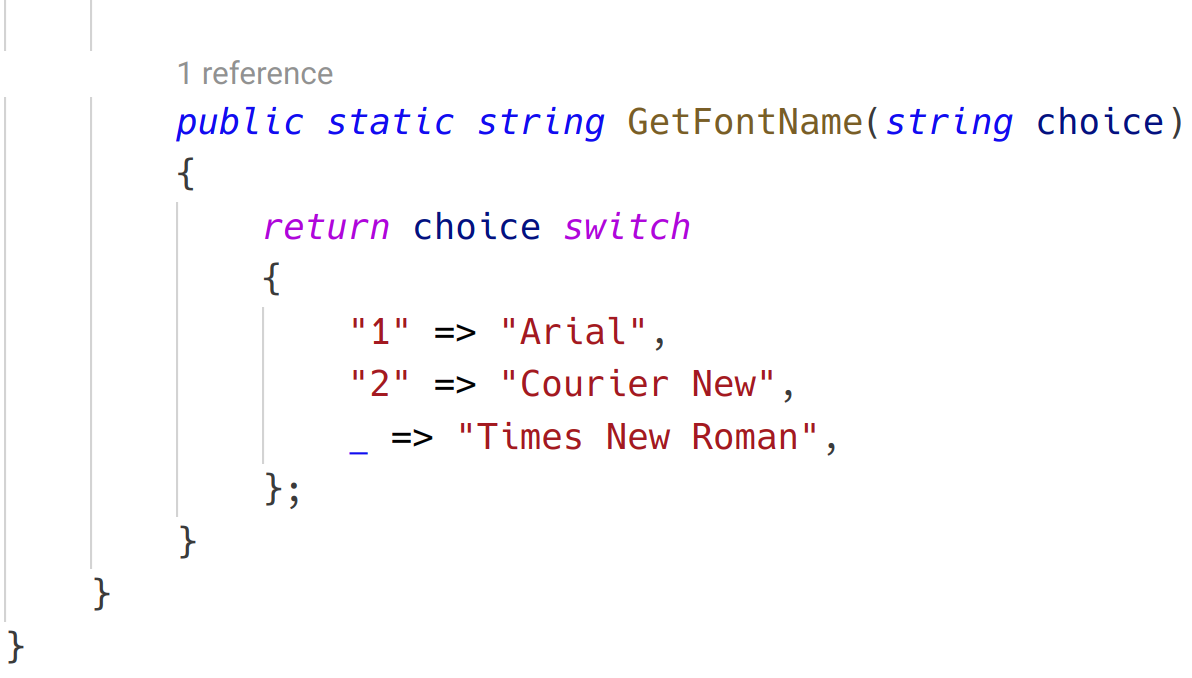
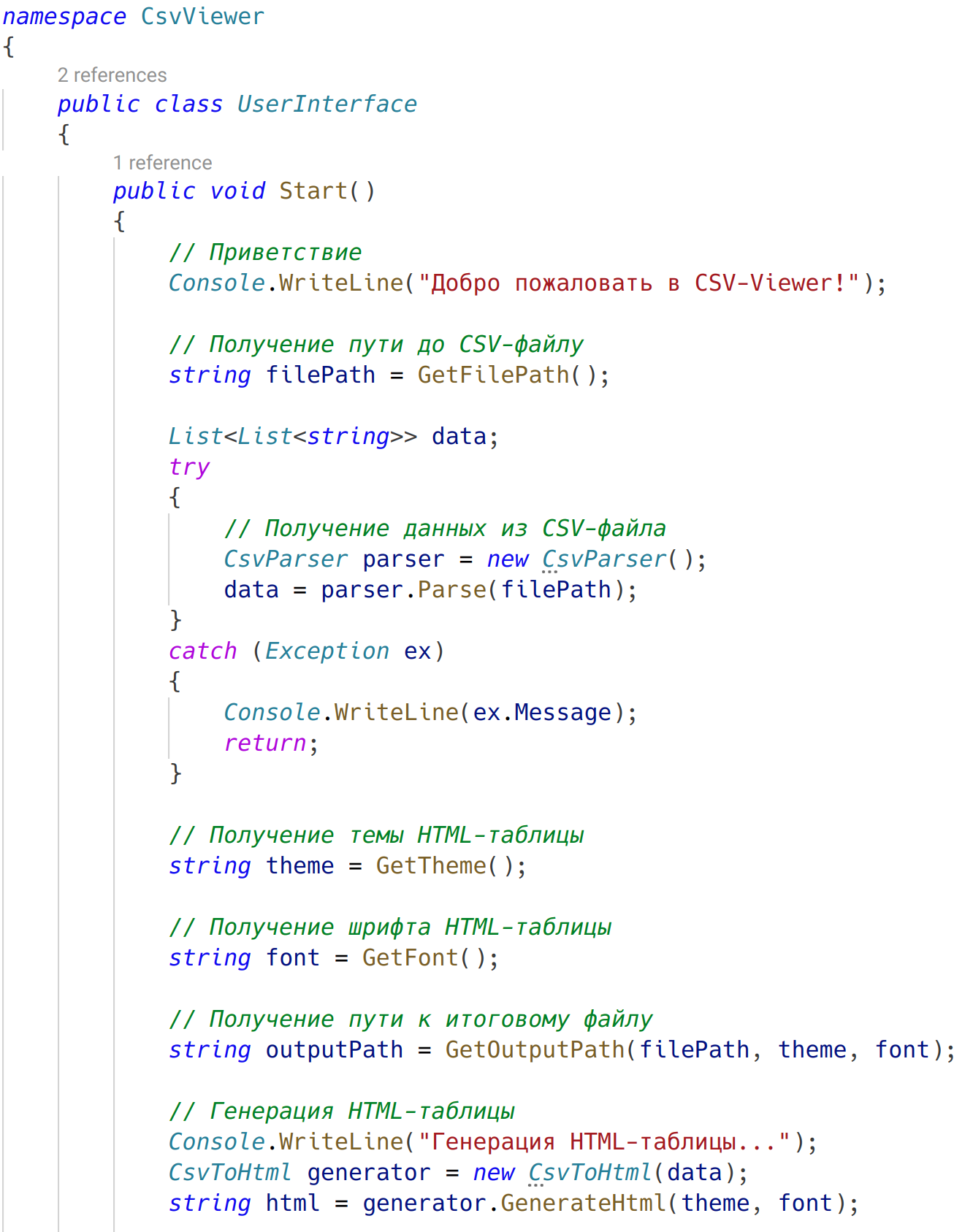


Рисунок 9 – класс ThemeManager.

На рисунке 10 представлен класс UserInterface:



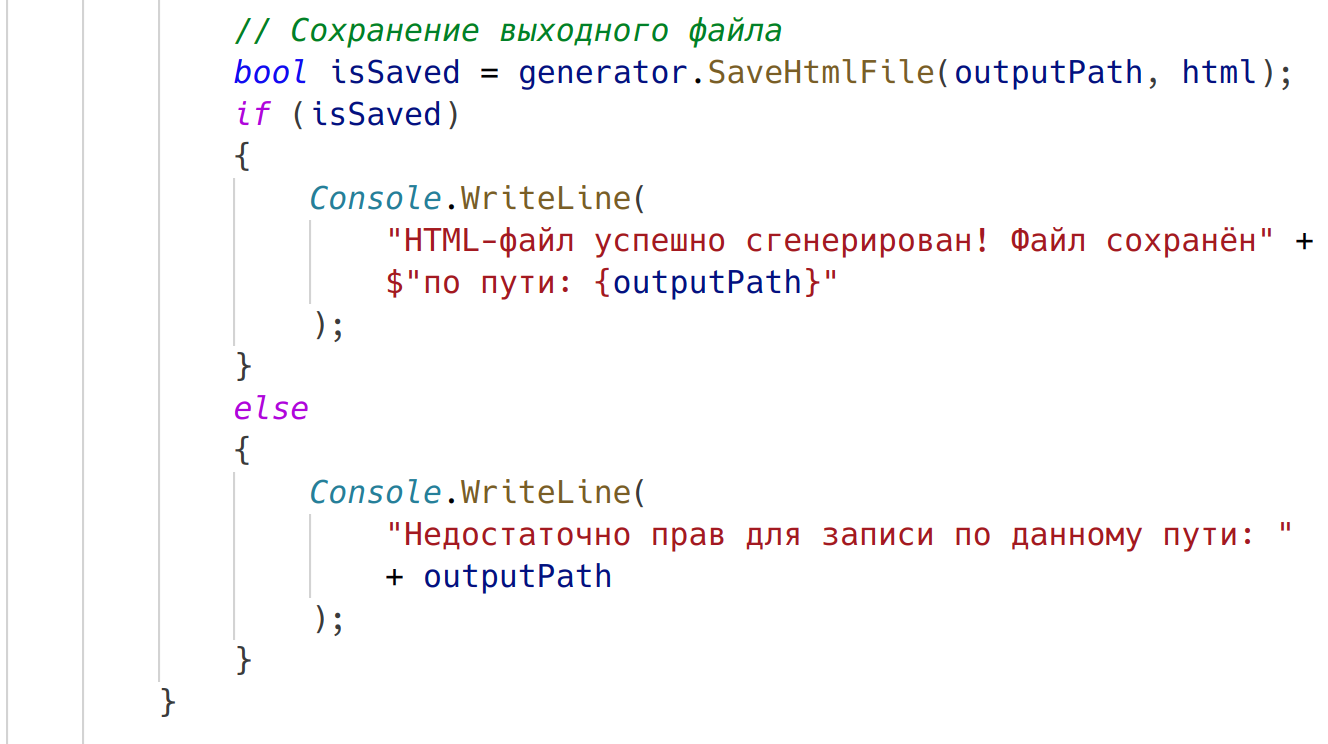










Рисунок 10 – класс UserInterface.

На рисунке 11 представлен класс CsvToHtml:





Рисунок 11 – класс CsvToHtml.

На рисунке 12 представлен класс CsvParser:

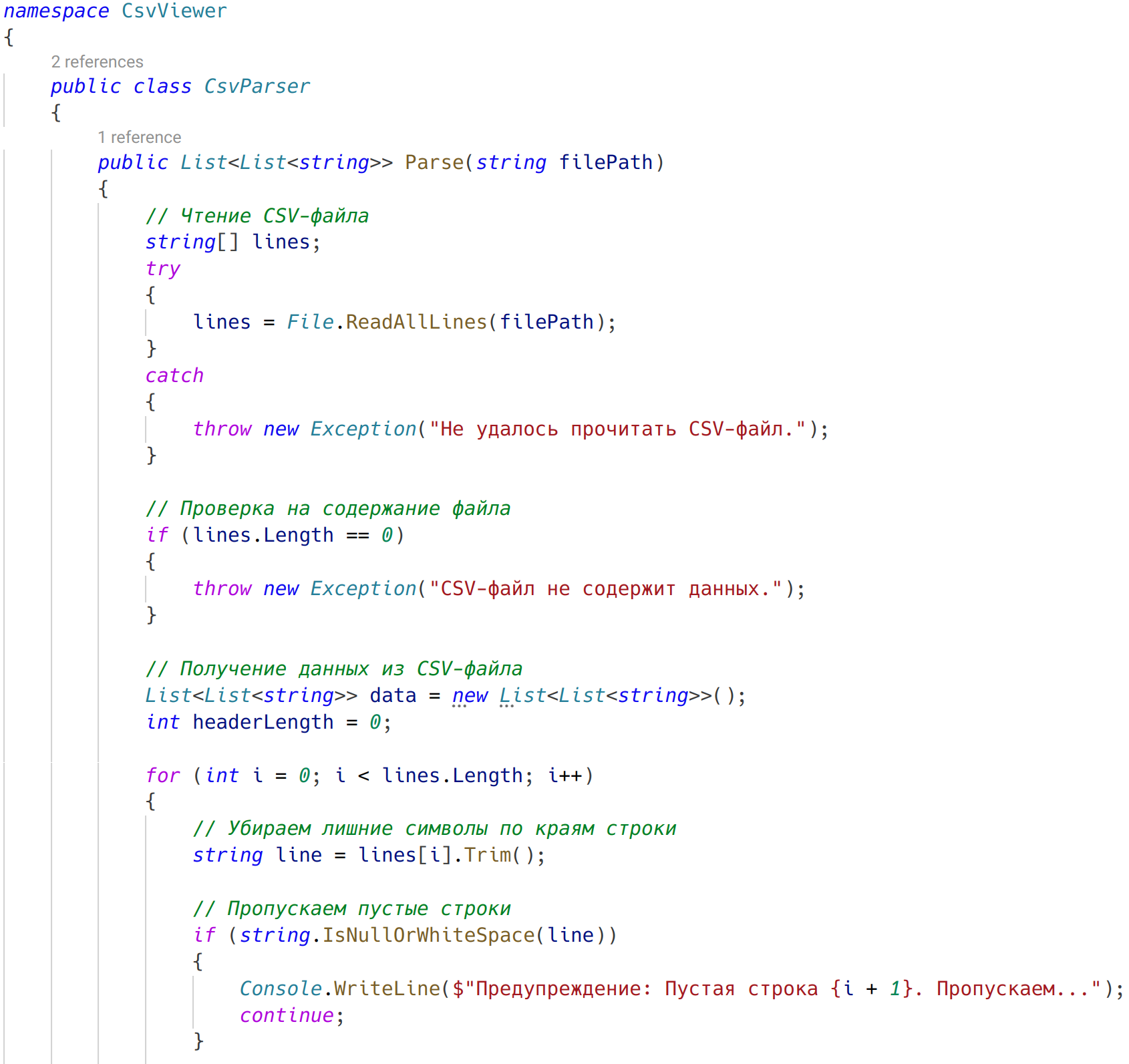






Рисунок 12 – класс CsvParser.