**Руководство программиста**

**к тестовому заданию Simbirsoft**

**участника мастер-классов по C#**

**Матюшина Данилы**

**Полный список документированных пакетов:**

* TestTaskSimbirsoft
* TextHandlerInterface
* TextHandlerLibrary
* HTMLHandlerLibrary

Классы с их кратким описанием:

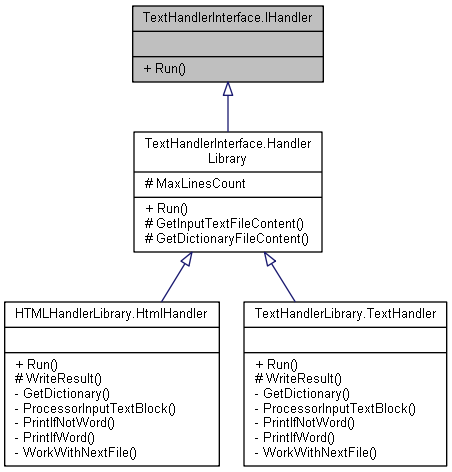
|  |  |
| --- | --- |
| **TestTaskSimbirsoft** | |
| class ***Program*** |  |
| **TestHandlerInterface** | |
| interface ***IHandler*** | Интерфейс обработчика. Описывает один метод - запуск обработки (например, входного текстового файла). |
| class ***HandlerLibrary*** | Абстрактный класс, наследующий интерфейс *IHandler*. Выступает как базовый для различных реализаций интерфейса обработчика. |
| **TextHandlerLibrary** | |
| class ***TextHandlerLibrary*** | Класс обработчика, формирующего из исходного текста в соответствии со словарем последовательности результирующих текстовых файлов, в которых слова, содержащиеся в словаре, написаны прописью. |
| **HTMLHandlerLibrary** | |
| class ***HtmlHandler*** | Класс обработчика, формирующего из исходного текста в соответствии со словарем последовательность результирующих html-файлов, в которых слова, содержащиеся в словаре, выделены жирным и курсивом. |

Иерархия классов:

A description...



Диаграмма наследования:



**Пакет TextHandlerInterface**

Классы:

* interface **IHandler**
* class **HandlerLibrary**

**Интерфейс TextHandlerInterface.IHandler**

Интерфейс обработчика. Описывает один метод - запуск обработки (например, входного текстового файла).

**Методы**

void **Run**(**string dictFileName, string textFileName, string resultFileName**)

Метод запуска обработки (например, входного текстового файла). Замещается в *HTMLHandlerLibrary.HtmlHandler*, *TextHandlerLibrary.TextHandler*, *TextHandlerInterface.HandlerLibrary*.

**Класс TextHandlerInterface.HandlerLibrary**

Абстрактный класс, наследующий интерфейс *IHandler*. Выступает как базовый для различных реализаций интерфейса обработчика.

Диаграмма наследования:

**Методы**

* **protected string[] GetDictionaryFileContent(string fileName)**

Метод используется для получения последовательности слов словаря из указанного входного файла. Возвращает массив слов словаря.

**Аргументы**:

fileName – Имя файла словаря

**Возвращает**:

Массив слов словаря.

* **protected IEnumerable<string> GetInputTextFileContent(string fileName)**

Метод используется для получения входной последовательности символов из указанного входного файла. Возвращает перечислитель, выполняющий итерацию считывания блока символов из входного файла.

**Аргументы**:

fileName – Имя входного файла с текстом.

**Возвращает**:

Перечислитель, по которому можно получить каждый следующий блок символов из входного потока (файла).

* **public abstract void Run(string dictFileName, string textFileName, string resultFileName)**

Абстрактный метод, реализующий метод Run() интерфейса обработчика IHandler. Используется для перегрузки метода в классах-наследниках.

Замещает *TextHandlerInterface.IHandler,* замещается в *HTMLHandlerLibrary.HtmlHandler* и *TextHandlerLibrary.TextHandler.*

**Аргументы**:

dictFileName – Имя файла словаря.

textFileName – Имя входного файла с текстом.

resultFileName – Имя результирующего файл.

**Данные класса**

**protected const int MaxLinesCount = 500**

Указывает максимальное количество строк, которые можно записать в выходной файл. При превышении порога, указанного в этой константе, должен формироваться новый выходной файл.

**Пакеты HTMLHandlerLibrary, TextHandlerLibrary**

**Класс HTMLHandlerLibrary.TextHandler**

Класс обработчика, формирующего из исходного текста в соответствии со словарем последовательность результирующих html-файлов, в которых слова, содержащиеся в словаре, выделены жирным и курсивом.

**Класс TextHandlerLibrary.TextHandler**

Класс обработчика, формирующего из исходного текста в соответствии со словарем последовательности результирующих тектовых файлов, в которых слова, содержащиеся в словаре, написаны прописью.

**Методы**

* **private HashSet<string> GetDictionary(string dictFileName)**

Формирует словарь и возвращает его вызвавшему методу.

**Аргументы**:

dictFileName – Имя файла со словарем.

**Возвращает**:

Словарь.

* **private static bool PrintIfNotWord(string curLine, StreamWriter outputStream, ref int curLinesCount, int charPos, string end)**

Записывает символ с указанной позицией в выходной поток. При нахождении конца строки увеличивает счетчик строк в выходном потоке, и указывает, требуется ли создание еще одного результирующего файла.

**Аргументы**:

curLine – Текущий блок символов, переданный на обработку.

outputStream – Выходной поток.

curLinesCount – Счетчик строк в выходном потоке.

charPos – Позиция символа во входном блоке.

end – Символы завершения предложения.

**Возвращает**:

Флаг превышения счетчиком строк в выходном файле максимально допустимого значения.

* **private static bool PrintIfWord(MatchCollection matchWordsInLine, int matchCounter, ref int charPos, StreamWriter outputStream, ICollection<string> dict)**

Определяет, является ли символ с указанной позицией частью слова, если да - записывает это слово в выходной поток, предварительно выполняя над ним действия по обработке. **Аргументы**:

matchWordsInLine – Набор слов во входном блоке.

matchCounter – Номер слова в наборе слов, с которого начинается поиск.

charPos – Позиция символа во входном блоке.

outputStream – Выходной поток.

dict – Словарь.

**Возвращает**:

Признак записи слова в выходной поток либо (осуществлялась ли запись).

* **private static string ProcessInputTextBlock(ICollection<string> dict, string curLine, StreamWriter outputStream, ref int curLinesCount)**

Метод обработки текущего блока символов.

**Аргументы**:

dict – Словарь.

curLine – Текущий блок символов.

outputStream – Открытый результирующий файл (поток), в который записываются результаты обработки.

curLinesCount – Счетчик записанных в результирующий файл (поток) строк (не больше MaxLinesCount).

**Возвращает**:

Часть переданного блока символов, не прошедший обработку, в случае, если счетчик записанных в результирующий файл (поток) строк превысит MaxLinesCount.

* **public override void Run(string dictFileName, string textFileName, string resultFileName)**

Перегруженный метод базового класса HandlerLibrary. Запускает обработку исходного текста.

**Аргументы**:

dictFileName – Имя файла словаря.

textFileName – Имя входного файла с текстом.

resultFileName – Имя результирующего файл.

* **protected static bool WorkWithNextFile(FileStrem fileStrem, IEnumerator<string> en, string resultFileName, ICollection<string> dict, ref string remainderLine)**

Обеспечивает вывод обработанного текста в переданный файловый поток.

**Аргументы**:

fileStrem – открытый результирующий файл (поток).

en - Перечислитель, по которому можно получить каждый следующий блок символов из входного потока (файла).

resultFileName - Имя (путь) результирующего файла (неизменная часть).

dict – Словарь.

remainderLine – Часть блока, остающаяся необработанной (если есть).

* **protected static void WriteResult(ICollection<string> dict, string resultFileName, IEnumerator<string> en, string fileExtension)**

Создает результирующий файл (поток) и записывает в него результат обработки. Возвращает флаг успешности записи результата обработки в результирующий файл.

**Аргументы**:

dict – Словарь.

resultFileName - Имя (путь) результирующего файла (неизменная часть).

en - Перечислитель, по которому можно получить каждый следующий блок символов из входного потока (файла).

fileExtension – Расширение результирующего файла.

**Пакет TestTaskSimbirsoft**

**Класс Program**

Класс программы.

**Методы**

* **protected string GetFileName(string message, string fileExtension)**

Возвращает имя (путь) рабочего файла. Метод используется для опроса пользователя и получения имени (пути) файла, с которым следует работать программе.

**Аргументы**:

message - Сообщение, которое следует отобразить пользователю. Может содержать явное указание, какой именно файл (словарь, исходный текст, результирующий файл) требуется программе.

fileExtension - Расширение (тип) файла (.txt, .html, и т.д.).

**Возвращает**:

Имя (путь) рабочего файла.

* **protected Dictionary<int, string> GetLibrariesInDirectory(string dirPath)**

Возвращает список (словарь) доступных в указанной директории обработчиков.

**Аргументы**:

dirPath – Рабочая директория приложения.

**Возвращает**:

Список (словарь) обработчиков.

* **protected void LoadHandler(string handlerLibName, string dictFileName, string textFileName, string resultFileName)**

Загружает выбранный обработчик и передает ему список имен входных и выходных файлов для обработки.

**Аргументы**:

handlerLibName – Имя обработчика.

dictFileName – Имя (путь) файла словаря (без указания расширения файла).

textFileName – Имя (путь) входного файла с текстом (без указания расширения файла).

resultFileName – Имя (путь) результирующего файла (без указания расширения файла).

* **protected void PrintListOfLibs(Dictionary<int, string> libs)**

Выводит на экран список доступных обработчиков.

**Аргументы**:

libs – Список (словарь) доступных обработчиков.

* **static void Main()**

Главный метод программы.

* **private static void Run()**

Запуск обработки с поиском доступных программе (в директории программы) обработчиков.

* **protected void SelectHandler(Dictionary<int, string> libs)**

Возвращает имя (путь) выбранного пользователем обработчика.

**Аргументы**:

libs – Список (словарь) доступных обработчиков.

**Возвращает**:

Имя (путь) выбранного пользователем обработчика.

**Обеспечение расширяемости решения**

Для расширяемости приложения рассмотрено использование рефлексии и позднего связывания.

Метод GetLibrariesInDirectory() ищет в заданной директории файлы (dll-библиотеки), содержащие в названии подстроку «HandlerLibrary.dll» (заканчивающиеся ею), составляет словарь найденных библиотек и возвращает его для дальнейшей работы.

Метод SelectHandler() отвечает за получение информации от пользователя, какой из найденных обработчиков следует использовать.

В методе LoadHandler() применяются указанные принципы. Выбранная сборка загружается и анализируется на наличие интересующего нас класса (реализации интерфейса IHandler). Если такой обнаруживается, то создается объект найденного класса и вызывается соответствующий метод обработчика (Run()).

Другой возможный способ обеспечить расширяемость приложения – проект ***MEF*** (Managed Extensibility Framework), включенный в Framework .NET 4.