**Отчет**

**Лабораторная работа №1**

**С дисциплины: «C#»**

**Тема: «** **Шаблоны проектирования. Порождающие шаблоны »**

**Автор: Ярцев Д.А.**

**Группа: ИС61**

**Цель**: ознакомиться с основными шаблонами проектирования, научиться применять их при проектировании и разработке ПО.

**Паттерн** — это не законченное архитектурное решение, которое можно напрямую преобразовать в исходный или машинный код. Это описание подхода к решению проблемы, который можно применять в разных ситуациях

**Вариант 10** (24)

Необходимо реализовать приложение, которое позволяет преобразовывать текст в специальном странном формате в несколько (не менее двух) других форматов. Например, в [html](http://ru.wikipedia.org/wiki/HTML) и [markdown](http://ru.wikipedia.org/wiki/Markdown).

**Формат исходного текста.**

Текст состоит из абзацев, разделённых одним или несколькими пустыми строками. Каждый абзац начинается с ключевого слова, за которым следует один пробел. Далее идёт смысловой текст.  
*Ключевые слова:*

p — обычный текст.

h1 — заголовок первого уровня.

h2 — заголовок второго уровня.

h3 — заголовок третьего уровня.

ordlist — нумерованный список. Далее каждая строка считается одним элементом списка. Символы начальной строки (которая содержит ключевое слово ordlist) игнорируются.

bullist — маркированный список.

Абзацы, начинающиеся с неизвестного ключевого слова (или без ключевого слова), не включаются в итоговый текст.

Пример текста:

h1 Заголовок 1

p Привет, мир!

p Внимание!

Это нумерованный список:

ordlist эти символы игнорируются

пункт 1;

пункт 2;

пункт 3.

h2 Заголовок 2

Этот текст игнорируется, так вот!

p А это маркированный список:

bullist

пункт 1;

пункт 2.

h1 Конец!

**Ход работы**

Для данной программы я применю паттерн «Factory method».Так как паттерн Factory Method позволяет базовым абстрактным классам передать ответственность за создание объектов-продуктов своим производным классам.

Общий алгоритм работы программы таков:

1. Пользователь вводит хаотичный текст
2. Программа фильтрует ввод на наличие систаксических едениц «специального» формата
3. Отталкиваясь от уже сформированного блока текста с «специальным форматом» формируется блок с html форматом
4. Отталкиваясь от уже сформированного блока текста с «специальным форматом» формируется блок с markdown форматом
5. Пользователю предлагают перейти к пункту 1 или завершить программу.

**Реализация: вывод**

================================================================================

Author: Yarcev Dmitry

group: IS-61

number laboratory work: 1

================================================================================

Please enter some text in a specific format:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

h1 Заголовок 1

p Привет, мир!

p Внимание!

Это нумерованный список:

ordlist эти символы игнорируются

пункт 1;

пункт 2;

пункт 3.

h2 Заголовок 2

Этот текст игнорируется, так вот!

p А это маркированный список:

bullist

пункт 1;

пункт 2.

h1 Конец!

End!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Сonverting received text into a specific format...

--------------------------------------------------------------------------------

h1 Заголовок 1

p Привет, мир!

p Внимание!

Это нумерованный список:

ordlist

пункт 1;

пункт 2;

пункт 3.

h2 Заголовок 2

p А это маркированный список:

bullist

пункт 1;

пункт 2.

h1 Конец!

--------------------------------------------------------------------------------

Сonverting received text into a HTML format...

--------------------------------------------------------------------------------

< !DOCTYPE html >

< html >

< head >

< meta charset = 'utf-8' />

< title > HTML5 </ title >

< !--[if IE]>

< script src = 'http://html5shiv.googlecode.com/svn/trunk/html5.js' ></ script >

< ![endif]-- >

< style >

article, aside, details, figcaption, figure, footer,header,

hgroup, menu, nav, section { display: block; }

</ style >

</ head >

< body >

< h1 >

Заголовок 1

</ h1 >

< p >

Привет, мир!

</ p >

< p >

Внимание!

Это нумерованный список:

</ p >

< ol >

< li >пункт 1;</ li >

< li >пункт 2;</ li >

< li >пункт 3.</ li >

</ ol >

< h2 >

Заголовок 2

</ h2 >

< p >

А это маркированный список:

</ p >

< ul >

< li >пункт 1;</ li >

< li >пункт 2.</ li >

</ ul >

< h1 >

Конец!

</ h1 >

</ body >

</ html >

--------------------------------------------------------------------------------

Сonverting received text into a Markdown format...

--------------------------------------------------------------------------------

#Заголовок 1

Привет, мир!

Внимание!Это нумерованный список:

1 пункт 1;

2 пункт 2;

3 пункт 3.

##Заголовок 2

А это маркированный список:

\* пункт 1;

\* пункт 2.

#Конец!

--------------------------------------------------------------------------------

Are you want repeat this program? (y/n):

**Реализация: код**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Program

{

//данная функция выводит строку состоящую из указанного символа

static void lineCh(char ch)

{

for (int i = 0; i < 80; i++)

Console.Write(ch);

Console.WriteLine();

}

//данная функция выводит подпись автора

static void authorLine()

{

lineCh('=');

Console.WriteLine(" Author: Yarcev Dmitry");

Console.WriteLine(" group: IS-61");

Console.WriteLine(" number laboratory work: 1");

lineCh('=');

}

//основная функция программы

static void Main(string[] args)

{

//подпись автора

authorLine();

//основной цыкл программы

bool loop = true;

while (loop)

{

Console.WriteLine(" Please enter some text in a specific format:");

lineCh('\*');

//вводим набор строк

string input = "";

bool loopInput = true;

while (loopInput)

{

//добавляем новую строку

input += Console.ReadLine().ToString() + "\n";

//продолжаем до тех пор пока не появится строка с таким окончанием "h1 end!.

if (input.EndsWith("End!\n"))

loopInput = false;

}

lineCh('\*');

Console.WriteLine(" Сonverting received text into a specific format...");

//создаем текст странного формата

lineCh('-');

Creator creator = new XCreator();

TextBlock block = creator.CreateMethod(input);

block.ShowContent();

input = block.GetContent();

lineCh('-');

//создаем текст html формата

Console.WriteLine(" Сonverting received text into a HTML format...");

lineCh('-');

creator = new HtmlCreator();

block = creator.CreateMethod(input);

block.ShowContent();

lineCh('-');

//создаем текст markdown формата

Console.WriteLine(" Сonverting received text into a Markdown format...");

lineCh('-');

creator = new MarkdownCreator();

block = creator.CreateMethod(input);

block.ShowContent();

lineCh('-');

//проверяем, желает ли пользователь завершить программу

bool loopAsk = true;

while(loopAsk)

{

Console.Write(" Are you want repeat this program? (y/n): ");

char[] boolKey = Console.ReadLine().ToCharArray();

if (boolKey.Length == 0)

boolKey = new char[] { 'n'};

if (boolKey[0] == 'y')

{

loopAsk = false;

}

else if (boolKey[0] == 'n')

{

Console.WriteLine(" Now program is ending.");

loop = false;

loopAsk = false;

}

else

{

Console.WriteLine(" You don't use correct symbol.");

}

}

lineCh('=');

Console.WriteLine();

}

//пауза перед окончанием программы

Console.ReadLine();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

/// <summary>

/// Creator - Создатель:

/// Предоставляет интерфейс(абстрактные фабричные методы)

/// для порождения продуктов.

/// В частных случаях класс Creator может предоставлять реализацию фабричных методов,

/// которые возвращают экземпляры продуктов(ConcreteTextBlock)

/// </summary>

abstract class Creator

{

public abstract TextBlock CreateMethod(string text);

}

/// <summary>

/// ConcreteCreator - Конкретная фабрика:

/// Реализует интерфейс(фабричные методы) предоставляемый базовым классом Creator.

/// </summary>

class XCreator : Creator

{

public override TextBlock CreateMethod(string text)

{

TextBlock block = new XTextBlock();

//список допустимых строк

List<string> list\_starts= new List<string>()

{

"p ",//0

"h1 ",//1

"h2 ",//2

"h3 ",//3

"ordlist",//4

"bullist"//5

};

//отсортируем лишнее

string[] old\_lines = text.Split('\n');

text = "";

bool thisTrueText = false;

string labelTrueText = "";

int i = 0;

while (!old\_lines[i].EndsWith("End!"))

{

//проверка на список

if (old\_lines[i] != "")

{

foreach (string element in list\_starts)

if (old\_lines[i].StartsWith(element))

{

thisTrueText = true;

labelTrueText = element;

}

}

else

{

thisTrueText = false;

}

//действие в зависимости от того

if (thisTrueText)

{

if (old\_lines[i].StartsWith(list\_starts[4]))

{

text += list\_starts[4] + '\n';

}

else if (old\_lines[i].StartsWith(list\_starts[5]))

{

text += list\_starts[5] + '\n';

}

else

{

text += old\_lines[i] + '\n';

}

}

else

{

if(!text.EndsWith("\n\n") && text != "")

text += '\n';

}

i++;

}

//присвоение

block.InputContent(text);

return block;

}

}

/// <summary>

/// ConcreteCreator - Конкретная фабрика:

/// Реализует интерфейс(фабричные методы) предоставляемый базовым классом Creator.

/// </summary>

class HtmlCreator : Creator

{

public override TextBlock CreateMethod(string text)

{

TextBlock block = new HtmlTextBlock();

//список допустимых строк

List<string> list\_starts = new List<string>()

{

"p ",//0

"h1 ",//1

"h2 ",//2

"h3 ",//3

"ordlist",//4

"bullist"//5

};

List<string> list\_tegs = new List<string>()

{

"\n< p >|</ p >",//0

"\n< h1 >|</ h1 >",//1

"\n< h2 >|</ h2 >",//2

"\n< h3 >|</ h3 >",//3

"\n< ol >|\n</ ol >",//4

"\n< ul >|\n</ ul >"//5

};

string new\_text = @"< !DOCTYPE html >

< html >

< head >

< meta charset = 'utf-8' />

< title > HTML5 </ title >

< !--[if IE]>

< script src = 'http://html5shiv.googlecode.com/svn/trunk/html5.js' ></ script >

< ![endif]-- >

< style >

article, aside, details, figcaption, figure, footer,header,

hgroup, menu, nav, section { display: block; }

</ style >

</ head >

< body >

";

//обработка

text += "\nEnd!";

string[] old\_lines = text.Split('\n');

text = "";

string[] currentTeg = new string[2] { "", ""};

int i = 0;

int current\_number = 0;

while (!old\_lines[i].EndsWith("End!"))

{

//проверка на список

if (old\_lines[i] != "")

{

bool check = false;

foreach (string element in list\_starts)

if (old\_lines[i].StartsWith(element))

{

check = true;

current\_number = list\_starts.IndexOf(element);

currentTeg = list\_tegs[current\_number].Split('|');

if (current\_number <= 3)

new\_text += old\_lines[i].Replace(element, currentTeg[0] + "\n") + "\n";

else

new\_text += currentTeg[0];

//Console.WriteLine("{=="+ element+"==}");

}

if (!check && current\_number >= 4)

new\_text += "\n< li >" + old\_lines[i] + "</ li >";

else if (!check)

{

new\_text += old\_lines[i]+"\n";

}

check = false;

}

else

{

new\_text += currentTeg[1] + "\n";

currentTeg = new string[2] { "", ""};

}

i++;

}

//закрываем документ

new\_text += @"

</ body >

</ html >";

new\_text += '\n';

//присвоение

block.InputContent(new\_text);

return block;

}

}

/// <summary>

/// ConcreteCreator - Конкретная фабрика:

/// Реализует интерфейс(фабричные методы) предоставляемый базовым классом Creator.

/// </summary>

class MarkdownCreator : Creator

{

public override TextBlock CreateMethod(string text)

{

TextBlock block = new MarkdownTextBlock();

//список допустимых строк

List<string> list\_starts = new List<string>()

{

"p ",//0

"h1 ",//1

"h2 ",//2

"h3 ",//3

"ordlist",//4

"bullist"//5

};

List<string> list\_tegs = new List<string>()

{

"\n|",//0

"\n#|",//1

"\n##|",//2

"\n###|",//3

"|",//4

"|"//5

};

string new\_text = "";

//обработка

text += "\nEnd!";

string[] old\_lines = text.Split('\n');

text = "";

string[] currentTeg = new string[2] { "", "" };

int i = 0;

int num = 1;

int current\_number = 0;

while (!old\_lines[i].EndsWith("End!"))

{

//проверка на список

if (old\_lines[i] != "")

{

bool check = false;

foreach (string element in list\_starts)

if (old\_lines[i].StartsWith(element))

{

check = true;

current\_number = list\_starts.IndexOf(element);

currentTeg = list\_tegs[current\_number].Split('|');

if (current\_number <= 3)

new\_text += old\_lines[i].Replace(element, currentTeg[0] + "") + "";

else

new\_text += currentTeg[0];

}

if (!check && current\_number >= 4)

{

if(current\_number == 5)

new\_text += "\n\* " + old\_lines[i];

else

{

new\_text += "\n" + num + " " + old\_lines[i];

num++;

}

}

else if (!check)

{

new\_text += old\_lines[i];

}

check = false;

}

else

{

new\_text += currentTeg[1] + "\n";

num = 1;

currentTeg = new string[2] { "", "" };

}

i++;

}

//закрываем документ

new\_text += '\n';

//присвоение

block.InputContent(new\_text);

return block;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

/// <summary>

/// Textblock - Продукт:

/// Предоставляет интерфейс для взаимодействия с текстовыми блоками.

/// </summary>

abstract class TextBlock

{

public abstract void InputContent(string input);

public abstract void ShowContent();

public abstract string GetContent();

}

/// <summary>

/// ConcreteTextblock - Конкретный продукт:

/// Реализует интерфейс предоставляемый базовым классом Textblock.

/// </summary>

class XTextBlock : TextBlock

{

public string content = "";

public XTextBlock()

{ }

public override void InputContent(string input)

{

this.content = input;

}

public override void ShowContent()

{

Console.Write(this.content);

}

public override string GetContent()

{

return this.content;

}

}

/// <summary>

/// ConcreteTextblock - Конкретный продукт:

/// Реализует интерфейс предоставляемый базовым классом Textblock.

/// </summary>

class HtmlTextBlock : TextBlock

{

public string content = "";

public HtmlTextBlock()

{ }

public override void InputContent(string input)

{

this.content = input;

}

public override void ShowContent()

{

Console.Write(this.content);

}

public override string GetContent()

{

return this.content;

}

}

/// <summary>

/// ConcreteTextblock - Конкретный продукт:

/// Реализует интерфейс предоставляемый базовым классом Textblock.

/// </summary>

class MarkdownTextBlock : TextBlock

{

public string content = "";

public MarkdownTextBlock()

{ }

public override void InputContent(string input)

{

this.content = input;

}

public override void ShowContent()

{

Console.Write(this.content);

}

public override string GetContent()

{

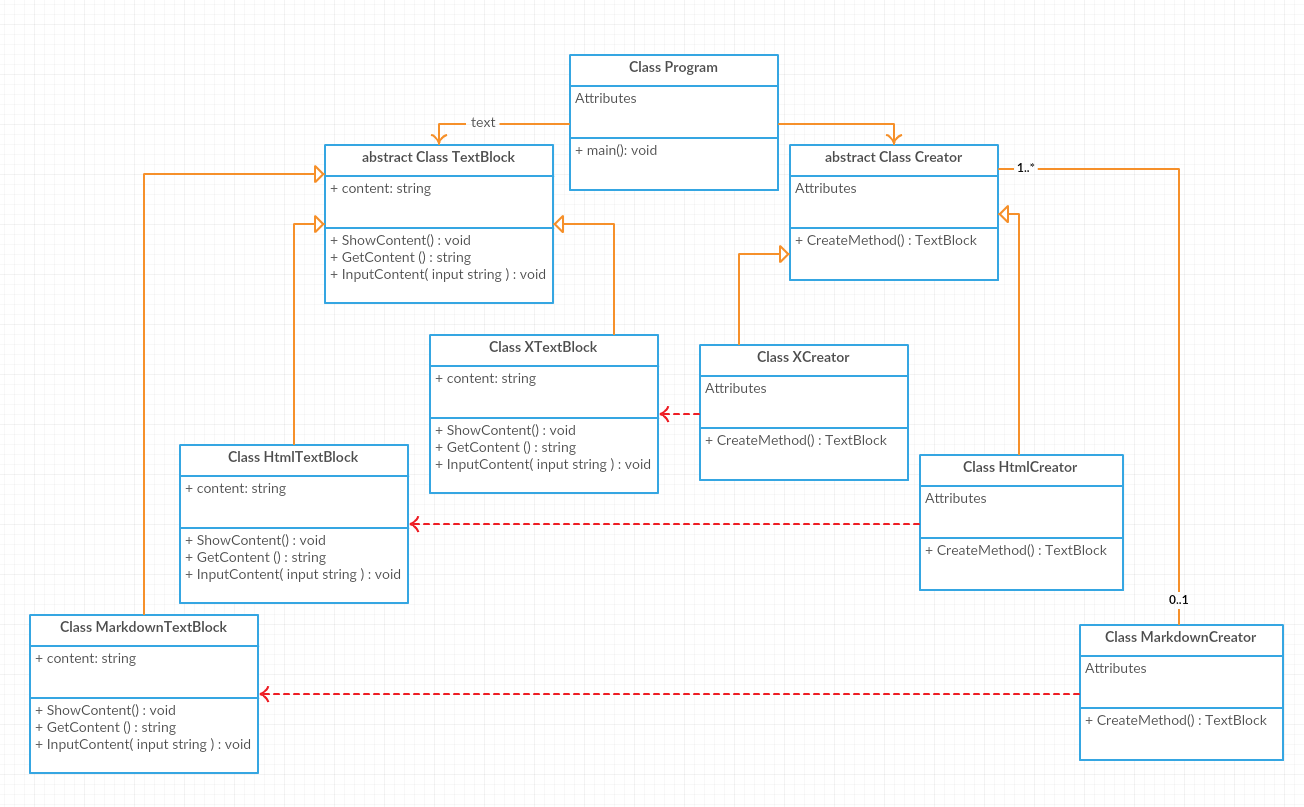
return this.content;

}

}

}

**Структура программы на языке UML**



**Классы**

|  |  |
| --- | --- |
| **Паттерн Factory method**  **Составные классы и их описание** | |
| **Наименование** | **Описание** |
| class Program | Основной класс программы с основным методом main() |
| abstract class Creator : Creator | Creator - Создатель:  Предоставляет интерфейс (абстрактные фабричные методы) для порождения продуктов(в нашем случае «текстовых блоков»). |
| class XCreator : Creator | XCreator - Конкретная фабрика:  Реализует интерфейс (фабричные методы) предоставляемый базовым классом Creator. |
| class HtmlCreator : Creator | HtmlCreator - Конкретная фабрика:  Реализует интерфейс (фабричные методы) предоставляемый базовым классом Creator. |
| class MarkdownCreator : Creator | MarkdownCreator - Конкретная фабрика:  Реализует интерфейс (фабричные методы) предоставляемый базовым классом Creator. |
| abstract class TextBlock | TextBlock - Продукт:  Предоставляет интерфейс для взаимодействия с продуктами(в нашем случае текстовыми блоками). |
| class XTextBlock : TextBlock | XTextBlock - Продукт:  Предоставляет интерфейс для взаимодействия с продуктами(в нашем случае текстовыми блоками). |
| class HtmlTextBlock : TextBlock | HtmlTextBlock - Продукт:  Предоставляет интерфейс для взаимодействия с продуктами(в нашем случае текстовыми блоками). |
| class MarkdownTextBlock : TextBlock | MarkdownTextBlock - Продукт:  Предоставляет интерфейс для взаимодействия с продуктами(в нашем случае текстовыми блоками). |

**Вивод:**

Паттерны довольно сильно ускоряют процесс разработки ПО. Обеспечивают четко прослеживаемую структуру.