ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

"ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ"

Факультет ИСП

Кафедра ПИ им. Л.П. Фельдмана

Лабораторная работа №4

по курсу: «Профессиональная практика программной инженерии»

по теме: «Создание самодокументирующегося кода»

Выполнил:

ст. гр. ПИ-19а

Гембицкий А. В.

Проверил:

Филипишин Д.А.

Ищенко А.П.

ДОНЕЦК – 2023

**Цель работы**

Научиться добавлять в программный код специальным образом оформление докблок-комментарии, для последующей автоматической генерации API reference, а также познакомиться с форматом оформления документации DocBook.

**Примеры докблоков**

**Collision.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace PPPI

{

/// <summary>

/// Класс коллизии.

/// Задаёт взаимодействие блока с другими блоками и игровым полем.

/// </summary>

internal class Collision

{

/// <summary>

/// Метод isInside отвечает за проверку нахождения блока внутри игрового поля.

/// Проверка осуществляется на основании размеров игрового поля и текущих координат проверяемого блока.

/// </summary>

/// <param name="block">Принимаемый объект класса Block, в котором храниться проверяемый блок.</param>

/// <param name="playingField">Принимаемый объект класса PlayingField, в котором храниться игровое поле.</param>

public bool isInside(Block block, PlayingField playingField)

{

return true;

}

/// <summary>

/// Метод isPosibleMove отвечает за проверку возможности сдвига блока на новые координаты.

/// Проверка осуществляется на основании размеров игрового поля, текущих и новых координат проверяемого блока.

/// </summary>

/// <param name="block">Принимаемый объект класса Block, в котором храниться проверяемый блок.</param>

/// <param name="playingField">Принимаемый объект класса PlayingField, в котором храниться игровое поле.</param>

/// <param name="newX">Принимаемая переменная int, в котором храниться новое положение блока по оси абсцисс в пикселях.</param>

/// <param name="newY">Принимаемая переменная int, в котором храниться новое положение блока по оси ординат в пикселях.</param>

public bool isPosibleMove(Block block, PlayingField playingField, int newX, int newY)

{

return true;

}

/// <summary>

/// Метод isOnBottom отвечает за проверку нахождения блока на дне игрового поля или других блоках.

/// Проверка осуществляется на основании размеров игрового поля и текущих координат проверяемого блока.

/// </summary>

/// <param name="block">Принимаемый объект класса Block, в котором храниться проверяемый блок.</param>

/// <param name="playingField">Принимаемый объект класса PlayingField, в котором храниться игровое поле.</param>

public bool isOnBottom(Block block, PlayingField playingField)

{

return true;

}

}

}

**Block.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace PPPI

{

/// <summary>

/// Класс блока.

/// Задаёт положение блока, его форму и внешний вид.

/// </summary>

internal class Block

{

/// <summary>

/// Переменная x, имеет тип данных - int и задаёт положение блока на игровом поле по оси абсцисс в пикселях.

/// </summary>

public int x;

/// <summary>

/// Переменная y, имеет тип данных - int и задаёт положение блока на игровом поле по оси ординат в пикселях.

/// </summary>

public int y;

/// <summary>

/// Переменная shape, имеет тип данных - int и задаёт номер формы блока.

/// </summary>

public int shape;

/// <summary>

/// Переменная color, имеет тип данных - Color и задаёт цвет блока.

/// </summary>

public Color color;

/// <summary>

/// Переменная skin, имеет тип данных - Bitmap и хранит в себе рисунок, являющийся

/// отображаемой моделью блока.

/// </summary>

public Bitmap skin;

/// <summary>

/// Метод Draw отвечает за отрисовку блока.

/// </summary>

/// <param name="pictureBox">Принимаемый объект класса PictureBox, в котором будет отрисовываться блок,

/// в соответствии с внешним видом хранящимся в переменных skin, color и shape и координатами

/// на игровом поле хранящимися в переменных x и y.</param>

public void Draw(PictureBox pictureBox)

{

}

/// <summary>

/// Метод SetSkin отвечает за установку внешнего вида блока.

/// </summary>

/// <param name="skinData">Принимаемая переменная byte[], в которой храниться массив байт, который будет преобразован в

/// Bitmap и записан в переменную skin.</param>

public void SetSkin(byte[] skinData)

{

}

/// <summary>

/// Метод Movement отвечает за установку новых координат в переменные x и y.

/// </summary>

/// <param name="newX">Принимаемая переменная int, в котором храниться новое положение блока по оси абсцисс в пикселях.</param>

/// <param name="newY">Принимаемая переменная int, в котором храниться новое положение блока по оси ординат в пикселях.</param>

public void Movement(int newX, int newY)

{

}

/// <summary>

/// Метод Fallen отвечает за остановку блока при достижении им дна или падении на другой упавший блок.

/// </summary>

public void Fallen()

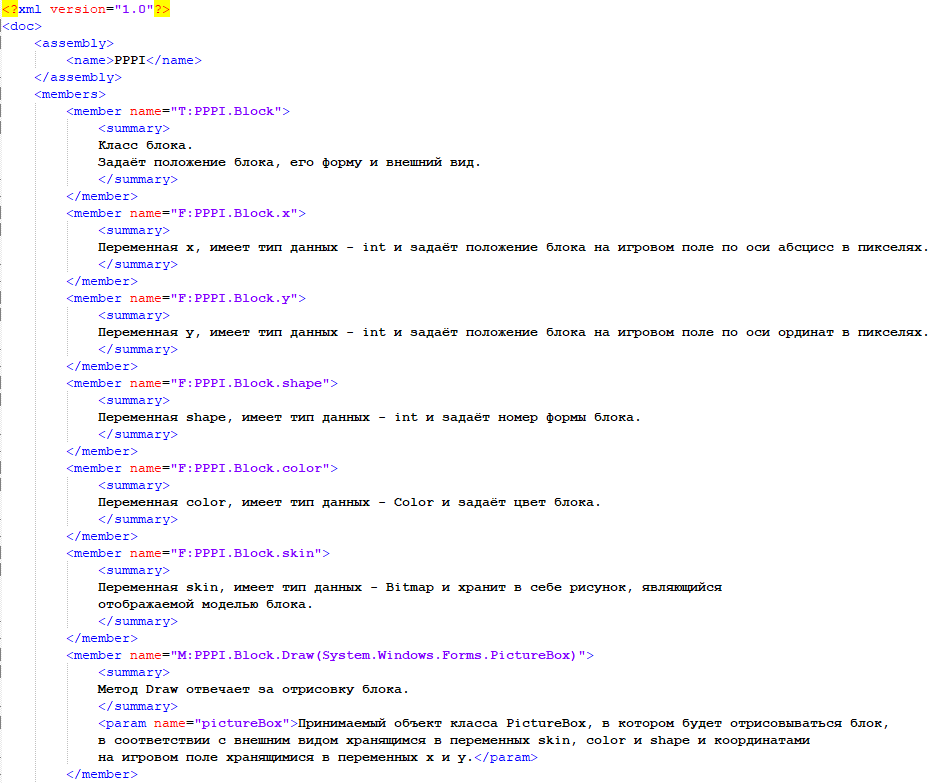
{

}

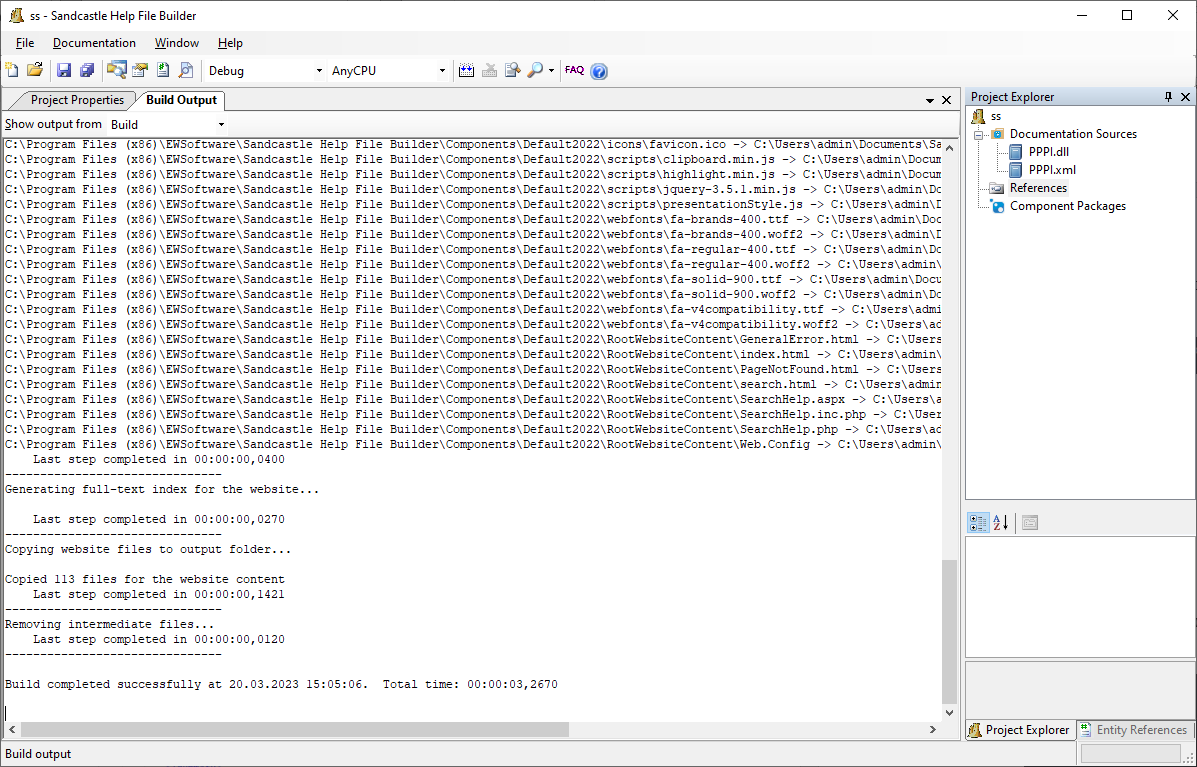
}

}

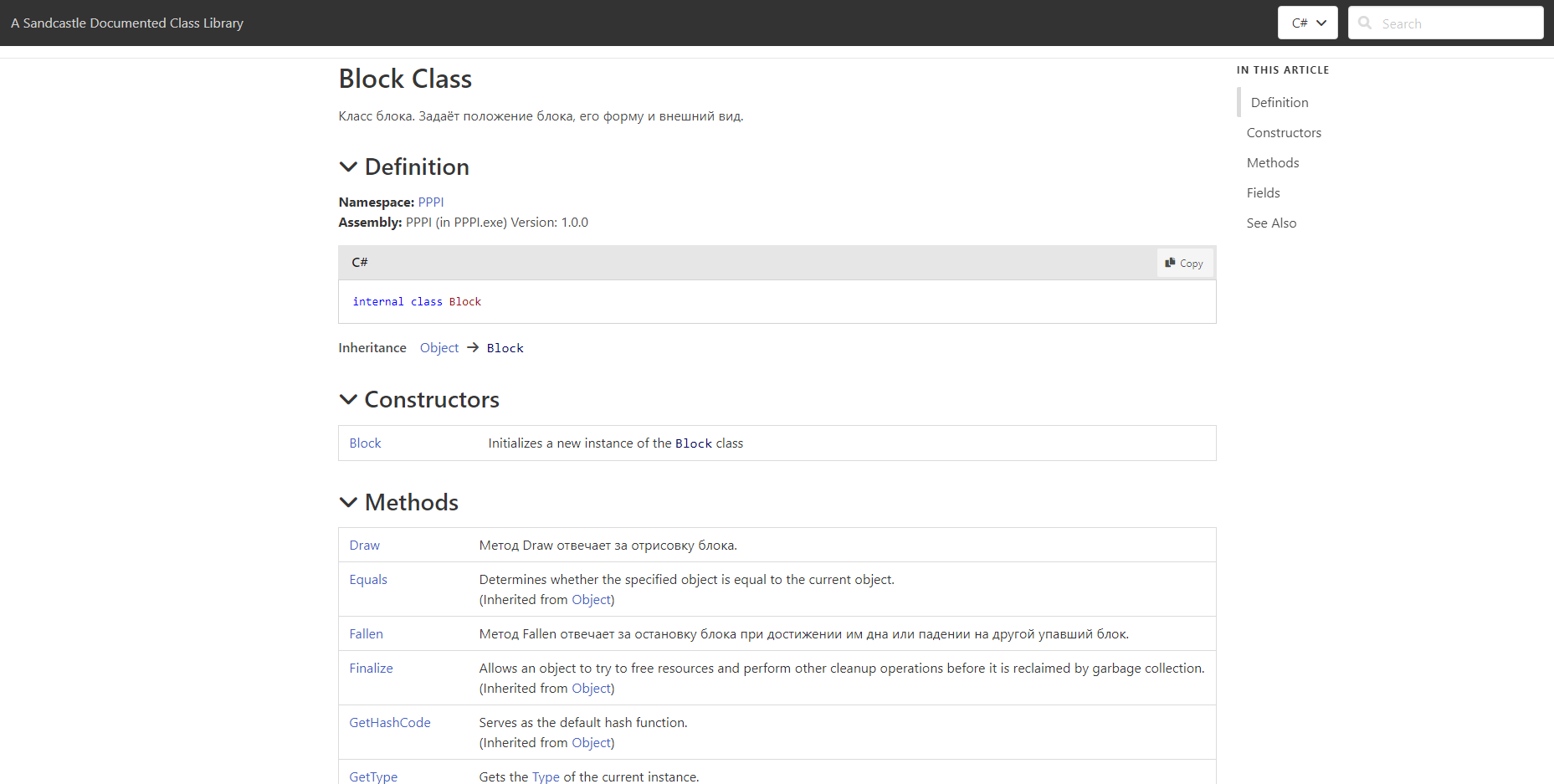
**Часть содержимого файла xml**

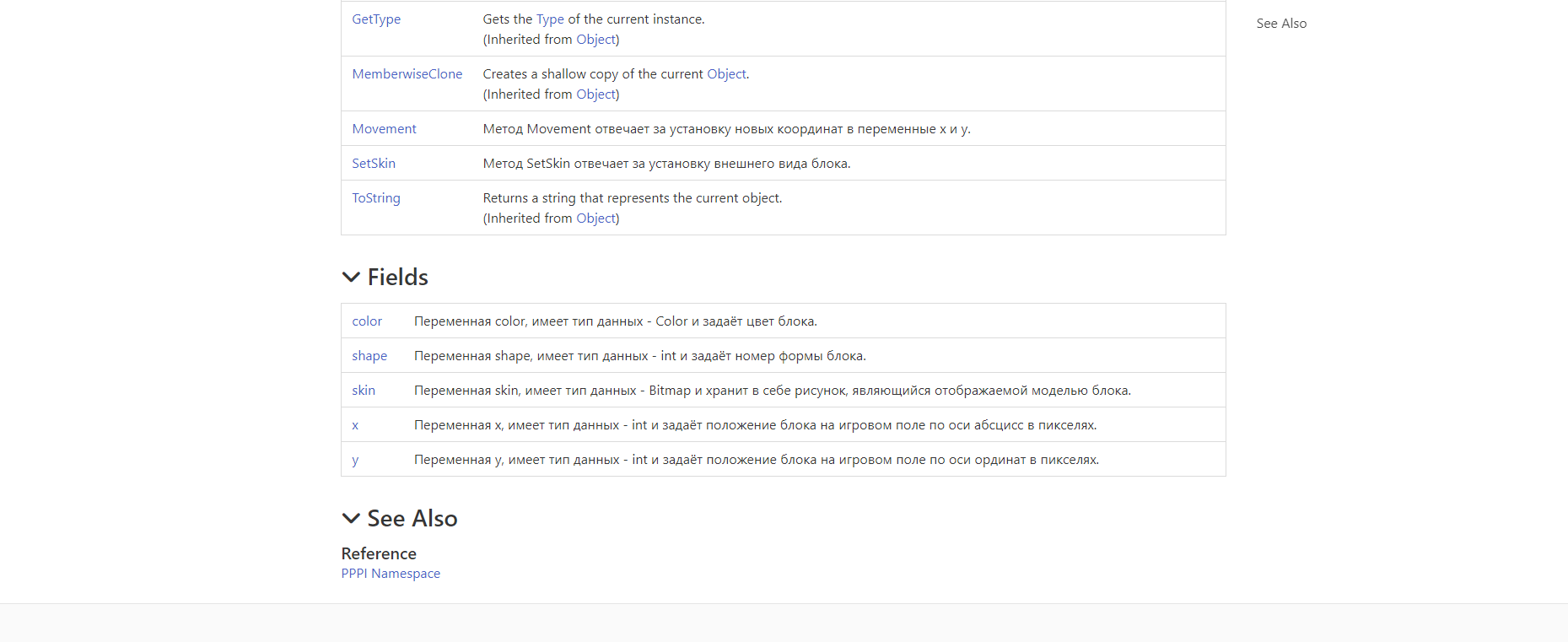
****

**Конвертация xml в сайт документации средством Sandcastle**



**Демонстрация работы**

****

****