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

Самойленко Р. 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**

**Animation.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace PPPI

{

/// <summary>

/// Создание класса Анимации

/// </summary>

class Animation

{

/// <summary>

/// Переменная Вид анимации

/// </summary>

string typeAnimation;

/// <summary>

/// Переменная Формат сохранения

/// </summary>

string FormatSave;

/// <summary>

/// Конструктор Animation

/// </summary>

/// <param name="\_typeAnimation"> Вид анимации</param>

/// /// <param name="\_FormatSave"> Формат сохранения</param>

public Animation(string \_typeAnimation, string \_FormatSave)

{

typeAnimation = \_typeAnimation;

FormatSave = \_FormatSave;

}

/// <summary>

/// Метод расчитывающий длительность анимации

/// </summary>

/// <param name="\_start"> Начало анимации</param>

/// <param name="\_finish"> Конец анимации</param>

public void TimeAnimation(double \_start, double \_finish)

{

double Time;

Time = \_finish - \_start;

}

/// <summary>

/// Рендеринг

/// </summary>

/// <param name="\_FormatSave">Формат сохранения</param>

public void Rendering(string \_FormatSave)

{

}

}

}

**ActionFigureOne.cs**

using System;

using using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace PPPI

{

/// <summary>

/// Класс действий с одной вигуры

/// </summary>

class ActionFigureOne

{

/// <summary>

/// Конструктор ActionFigureOne

/// </summary>

public ActionFigureOne()

{

}

/// <summary>

/// Метод расчёто точек после перемещения фигуры

/// </summary>

/// <param name="\_startkoor"> Начальные координаты</param>

/// <param name="\_endkoor"> Конечные координаты</param>

public void CalculationPointsMove(int \_startkoor, int \_endkoor)

{

}

/// <summary>

/// Изменения параметров фигуры

/// </summary>

/// <param name="\_newparam">Новые парамерты фигуры</param>

public void ChangingParameters(int \_newparam)

{

}

/// <summary>

/// Изменения освещения фигуры

/// </summary>

public void ChangingIllumination()

{

}

/// <summary>

/// Печать фигуры

/// </summary>

public void Print()

{

}

/// <summary>

/// Привязка камеры к фигуре

/// </summary>

public void SnappCameraFigure()

{

}

}

}

**Shape.cs**

using System;

using System.Collections.Generic;

using using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace PPPI

{

/// <summary>

/// Создание класса Shape(фигура)

/// </summary>

class Shape

{

/// <summary>

/// Переменная длины фигуры

/// </summary>

double length;

/// <summary>

/// Переменная длины фигуры

/// </summary>

double width;

/// <summary>

/// Переменная высоты фигуры

/// </summary>

double height;

/// <summary>

/// Переменная Тип фигуры

/// </summary>

string typeShape;

/// <summary>

/// Конструктор Shape

/// </summary>

/// <param name="\_length"> Длина фигуры</param>

/// <param name="\_width"> Ширина фигуры</param>

/// <param name="\_height"> Высота фигуры</param>

/// /// <param name="\_typeShape"> Тип фигуры</param>

public Shape(double \_length, double \_width, double \_height, string \_typeShape)

{

length = \_length;

width = \_width;

height = \_height;

typeShape = \_typeShape;

}

/// <summary>

/// Метод создания куба

/// </summary>

/// <param name="\_length"> Длина фигуры</param>

/// <param name="\_width"> Ширина фигуры</param>

/// <param name="\_height"> Высота фигуры</param>

public void Cube(double \_length, double \_width, double \_height)

{

}

/// <summary>

/// Метод создания цилиндра

/// </summary>

/// <param name="\_length"> Длина цилиндра</param>

/// <param name="\_height"> Высота цилиндра</param>

public void Cylinder(double \_length, double \_height)

{

}

}

}

**Cодержимое файла docbook.xml**

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.5//EN"

"http://www.oasis-open.org/docbook/xml/4.5/docbookx.dtd">

<book>

<bookinfo>

<title>3D-редактор</title>

<author>

<firstname>Ruslan</firstname>

<surname>Samoilenko</surname>

<affiliation>

<orgname>seltia0203@gmail.ru</orgname>

</affiliation>

</author>

</bookinfo>

<chapter>

<title>WorkingArea</title>

<para><emphasis> Класс рабочей зоны</emphasis></para>

<section>

<title>Конструкторы</title>

<para><emphasis> 1) WorkingArea()</emphasis></para>

</section>

<section>

<title>Методы</title>

<para><emphasis> 1) ViewingShapeInViews(): Просмотр проекции в 3 видах</emphasis></para>

</section>

</chapter>

<chapter>

<title>HorizonView</title>

<para><emphasis> Класс горизонтального вида</emphasis></para>

<section>

<title>Конструкторы</title>

<para><emphasis> 1) HorizonView()</emphasis></para>

</section>

</chapter>

<chapter>

<title>VerticView</title>

<para><emphasis> Класс вертикального вида</emphasis></para>

<section>

<title>Конструкторы</title>

<para><emphasis> 1) VerticView()</emphasis></para>

</section>

</chapter>

<chapter>

<title>SideView</title>

<para><emphasis> Класс бокового вида</emphasis></para>

<section>

<title>Конструкторы</title>

<para><emphasis> - SideView()</emphasis></para>

</section>

</chapter>

<chapter>

<title>Shape</title>

<para><emphasis> Класс Shape(фигура)</emphasis></para>

<section>

<title>Конструкторы</title>

<para><emphasis> 1) Shape(double \_length, double \_width, double \_height, string \_typeShape)()</emphasis></para>

<para>- double \_length - длина фигуры</para>

<para>- double \_width - ширина фигуры</para>

<para>- double \_height - высота фигуры</para>

<para>- double \_typeShape - тип фигуры</para>

</section>

<section>

<title>Поля</title>

<para><emphasis> 1) length : Параметр отвечающий за длину фигуры</emphasis></para>

<para><emphasis> 2) width : Параметр отвечающий за ширину фигуры</emphasis></para>

<para><emphasis> 3) height : Параметр отвечающий за высоту фигуры</emphasis></para>

<para><emphasis> 4) typeShape : Параметр отвечающий за тип фигуры</emphasis></para>

</section>

<section>

<title>Методы</title>

<para><emphasis> 1) void Cube(double \_length, double \_width, double \_height): Метод создания куба</emphasis></para>

<para>- double \_length - длина фигуры</para>

<para>- double \_width - ширина фигуры</para>

<para>- double \_height - высота фигуры</para>

<programlisting>

public void Cube(double \_length, double \_width, double \_height)

{

}

</programlisting>

<para><emphasis> 2) void Cylinder(double \_length, double \_height): Метод создания цилиндра</emphasis></para>

<para>- double \_length - длина фигуры</para>

<para>- double \_width - ширина фигуры</para>

</section>

</chapter>

<chapter>

<title>Animation</title>

<para><emphasis> Класс Анимации</emphasis></para>

<section>

<title>Конструкторы</title>

<para><emphasis> 1) Animation(string \_typeAnimation, string \_FormatSave)()</emphasis></para>

<para>- string \_typeAnimation - вид анимации</para>

<para>- string \_FormatSave - формат сохранения</para>

</section>

<section>

<title>Поля</title>

<para><emphasis> 1) string typeAnimation : Параметр отвечающий вид анимации</emphasis></para>

<para><emphasis> 2) string FormatSave : Параметр отвечающий формат сохранения</emphasis></para>

</section>

<section>

<title>Методы</title>

<para><emphasis> 1) void TimeAnimation(double \_start, double \_finish):Метод расчитывающий длительность анимации</emphasis></para>

<para>- double \_start - начало анимации</para>

<para>- double \_finish - конец анимации</para>

<programlisting>

public void TimeAnimation(double \_start, double \_finish)

{

double Time;

Time = \_finish - \_start;

}

</programlisting>

<para><emphasis> 2) void Rendering(string \_FormatSave):Рендеринг</emphasis></para>

<para>- string \_FormatSave - формат сохранения</para>

</section>

</chapter>

<chapter>

<title>Copy</title>

<para><emphasis> Класс копирования фигур</emphasis></para>

<section>

<title>Конструкторы</title>

<para><emphasis> 1) Copy()</emphasis></para>

</section>

</chapter>

<chapter>

<title>Сut</title>

<para><emphasis> Класс для вырезания части или же всей фигуры</emphasis></para>

<section>

<title>Конструкторы</title>

<para><emphasis> 1) Сut()</emphasis></para>

</section>

</chapter>

<chapter>

<title>Mirror</title>

<para><emphasis> Класс отзеркаливания фигур</emphasis></para>

<section>

<title>Конструкторы</title>

<para><emphasis> 1) Mirror()</emphasis></para>

</section>

</chapter>

<chapter>

<title>GroupTogether</title>

<para><emphasis> Класс копирования фигур</emphasis></para>

<section>

<title>Конструкторы</title>

<para><emphasis> 1) GroupTogether()</emphasis></para>

</section>

<section>

<title>Методы</title>

<para><emphasis> 1) void Ungroup(): Метод для разгруппировки фигур</emphasis></para>

</section>

</chapter>

<chapter>

<title>ActionWithFigure</title>

<para><emphasis> Класс действий</emphasis></para>

<section>

<title>Конструкторы</title>

<para><emphasis> 1) ActionWithFigure()</emphasis></para>

</section>

</chapter>

<chapter>

<title>ActionFigureOne</title>

<para><emphasis> Класс действий с одной фигуры</emphasis></para>

<section>

<title>Конструкторы</title>

<para><emphasis> 1) </emphasis></para>

</section>

<section>

<title>Методы</title>

<para><emphasis> 1) void CalculationPointsMove(int \_startkoor, int \_endkoor): Метод расчёто точек после перемещения фигуры</emphasis></para>

<para>- double \_startkoor - начальные координаты</para>

<para>- double \_endkoor - конечные координаты</para>

<para><emphasis> 2) void ChangingParameters(int \_newparam): Метод изменения параметров фигуры</emphasis></para>

<para>- double \_newparam - начальные координаты</para>

<para><emphasis> 3) void ChangingIllumination(): Метод изменения освещения фигуры</emphasis></para>

<para><emphasis> 4) void Print(): Метод печати фигуры</emphasis></para>

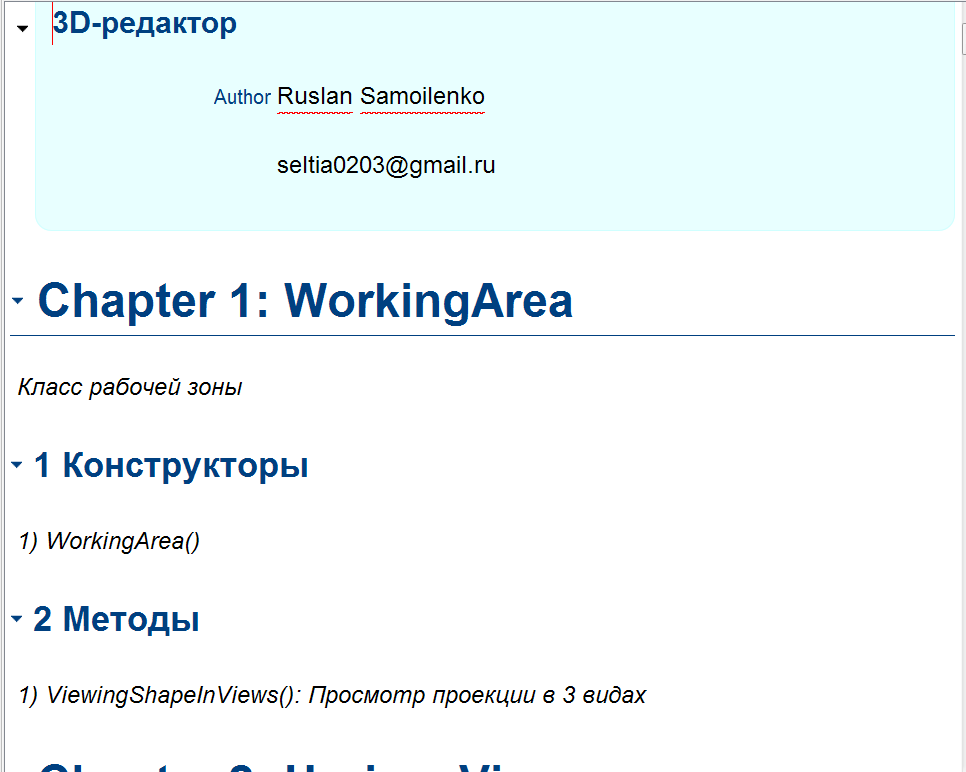
<para><emphasis> 5) void SnappCameraFigure(): Метод привязки камеры к фигуре</emphasis></para>

</section>

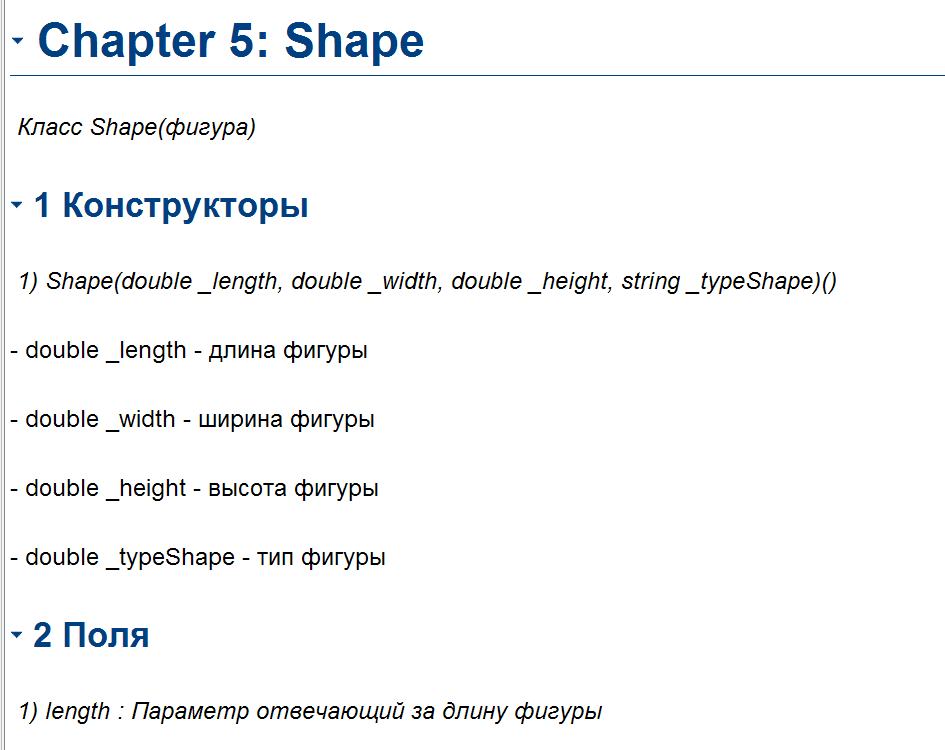
</chapter>

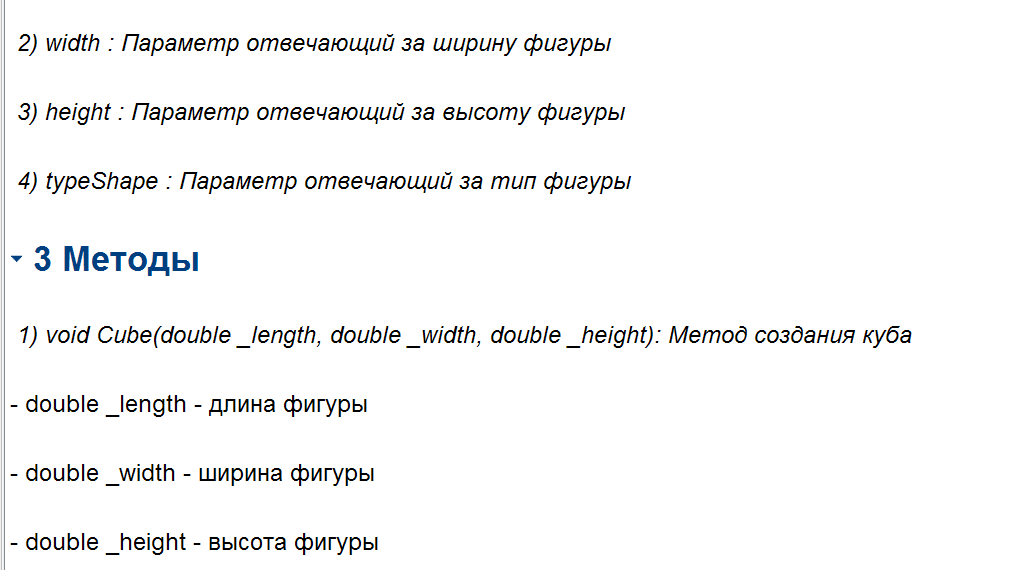
</book>

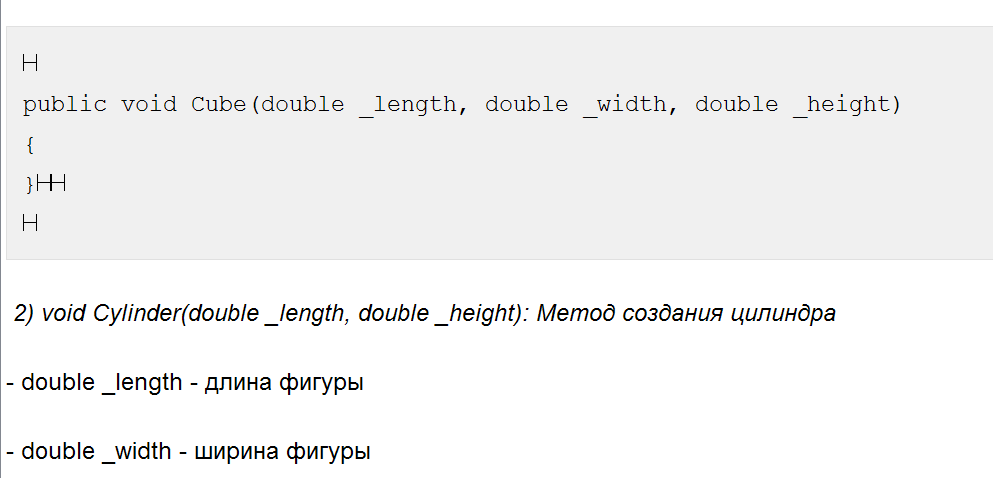
**Фрагменты DocBook**



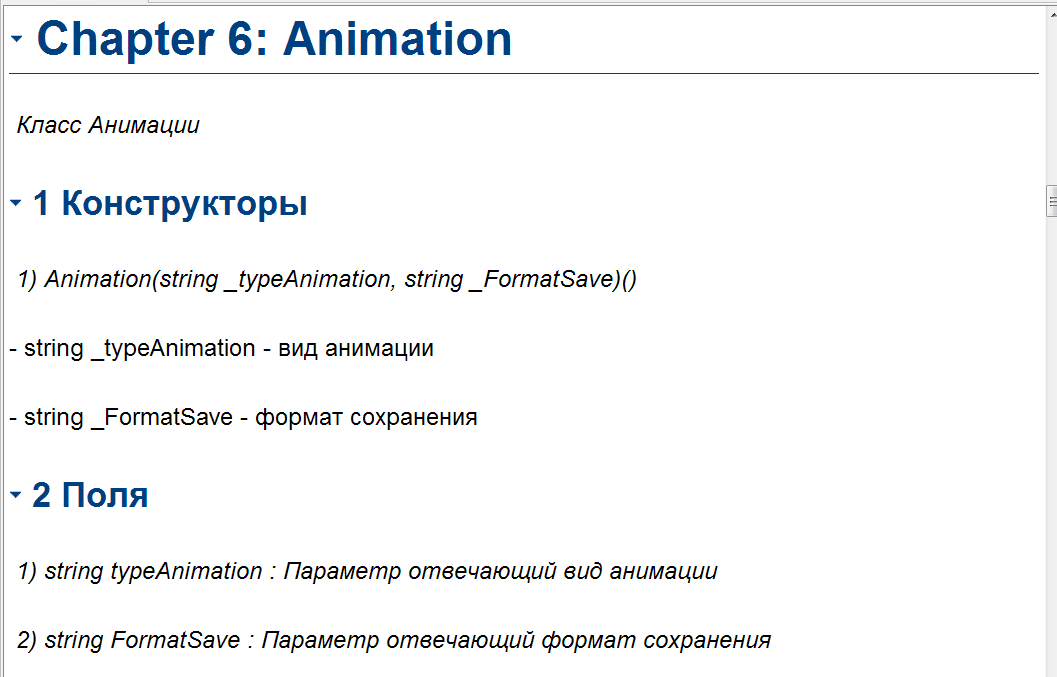
**Фрагмент DocBook(класса Shape)**

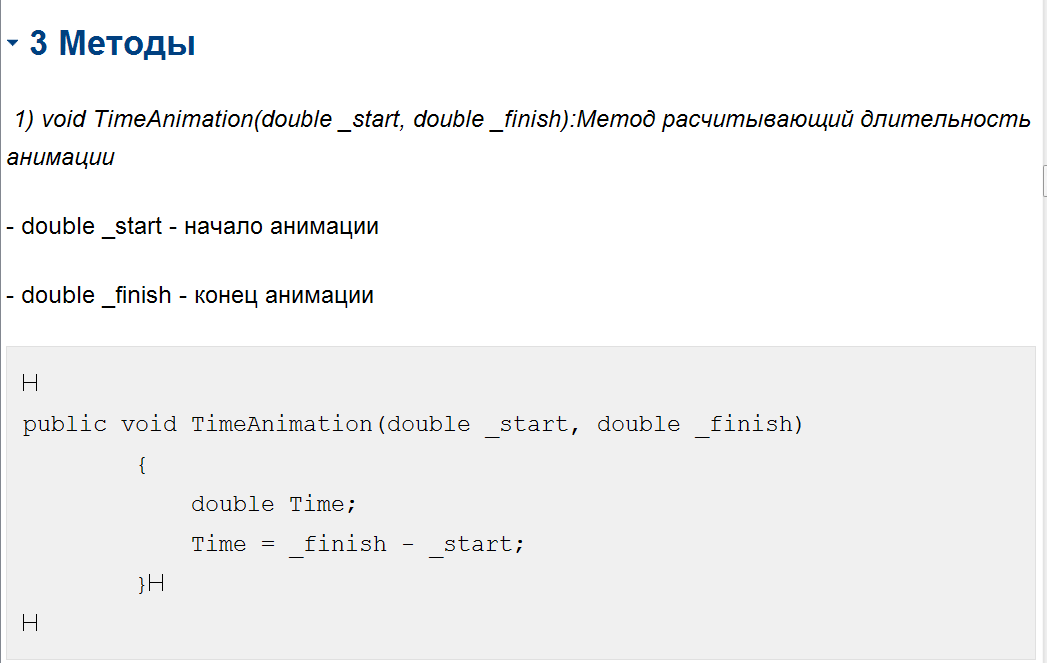


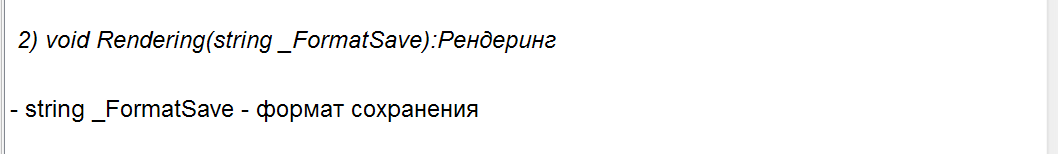




**Фрагмент DocBook(класса Shape)**







Т.к. окно ошибок редактора пустое, то валидация считается пройденной успешно (рисунок 9).

