# Практическая работа №17

**Тема: Разработка приложений с не визуальными компонентами**

**Цель работы:** формирование навыков создания приложений Windows Forms с использованием таймера. Изучение классов, реализующих задачу программирования печати и получение навыков по работе в программе с диалоговыми окнами

# Задачи:

* совершенствование приемов создания приложений Windows Forms;
* создание проектов с использованием диалоговых окон, средств печати.

# Материально-техническое обеспечение:

**Место проведения:** Компьютерный класс.

# Время на выполнение работы: 2 часа.

**Оборудование**: ПК

**Средства обучения:** операционная система, текстовый процессор MS Word, программные средства определенного вида

# Исходные данные:

1. Конспект занятия.
2. Задание для практической работы.

# Перечень справочной литературы:

1) Программирование на языке высокого уровня. Программирование на языке С++: учеб. пособие / Т. И. Немцова, С. Ю. Голова, А. И. Терентьев ; под ред. Л. Г. Гагариной. – М. : ИД

«ФОРУМ» : ИНФРА-М, 2018. – 512 с. – (Среднее профессиональное образование).

# Краткие теоретические сведения:

**Создание многооконных приложений**

Microsoft Visual Studio позволяет создавать две разновидности многооконных приложений: **SDI** - и **MDI**-приложения. **SDI-приложения** состоят из нескольких независимых форм (окон). По умолчанию будет создано SDI-приложение. В **MDI-приложении** имеется одна главная форма, остальные формы находятся в пределах главной; из главной формы можно управлять подчиненными формами. Единственное меню MDI-приложения находится в главном окне.

Перед созданием многооконного приложения его необходимо проектировать: продумать вопрос о том, какие окна нужны и что на них будет отображено. **Форма** – это разновидность класса. Экземпляры классов, как известно, необходимо создавать. Это правило распространяется и на формы: автоматически создается лишь одна форма – **главная**. Создание всех остальных форм лежит на программисте. Закрытие формы функцией **Close();** или нажатием на кнопку  вызывает уничтожение формы, и в случае необходимости она должна быть создана заново.

В принципе, любую задачу можно решить, как с помощью SDI-приложения, так и с помощью MDI-приложения. Пожалуй, создание SDI-приложения проще. MDI-приложение можно рекомендовать при необходимости создать и работать одновременно с несколькими одинаковыми формами.

# Создание файла PDF на C# .NET

Прямой поддержки для создания PDF-файлов на языке программирования C# нет, вместо этого можно использовать стороннюю библиотеку для создания PDF-файлов. Существует ряд платных и бесплатных библиотек для создания PDF-файлов.

Бесплатной библиотекой для создания файла PDF в C# .NET является PDFsharp.

**PDFsharp** — это библиотека .NET для обработки файлов PDF. Вы создаете страницы PDF, используя процедуры рисования, известные из GDI+. Почти все, что можно сделать с помощью GDI+, будет работать и с PDFsharp. PDFsharp поддерживает только базовую разметку текста, разрывы страниц не создаются автоматически. Одни и те же процедуры рисования можно использовать для файлов экрана, PDF или метафайлов.

# Создание документа для печати

Для обеспечения печати, в .NET Framework реализованы специальные классы, базовыми из которых есть:

* **PrintDocument** – представляет класс, объект которого посылает данные на принтер;
* **PageSetupDialog** – представляет диалоговое окно, которое позволяет пользователю изменять настройки страницы для печати (внутренние поля, ориентация печати, и т.п.);
* **PrintDialog** – представляет диалоговое окно, которое позволяет пользователю выбирать принтер и прочие настройки принтера, такие как количество копий, ориентация страницы и т.п.;
* **PrintPreviewDialog** – представляет диалоговое окно, которое показывает пользователю предыдущий просмотр документа в том виде как он будет отображен при печати.

**Ход работы:**

# Требования к содержанию отчета:

* Номер и название практической работы.
* Цель работы.
* По каждой заданию (задаче/примеру) экранные формы (при наличии) и листинг программного кода, показывающие порядок выполнения практической работы, и результаты, полученные в ходе её выполнения.
* Ответы на контрольные вопросы в тетради.

# Порядок выполнения работы:

Все проекты практической работы размещать в своей сетевой в новой папке **Пр17\_Фамилия В начале каждого файла проекта установить комментарии: пр.р.№** **(указать номер), свою Фамилию. Формулировку задания**

## Пример 1. Создание файла PDF на C# .NET

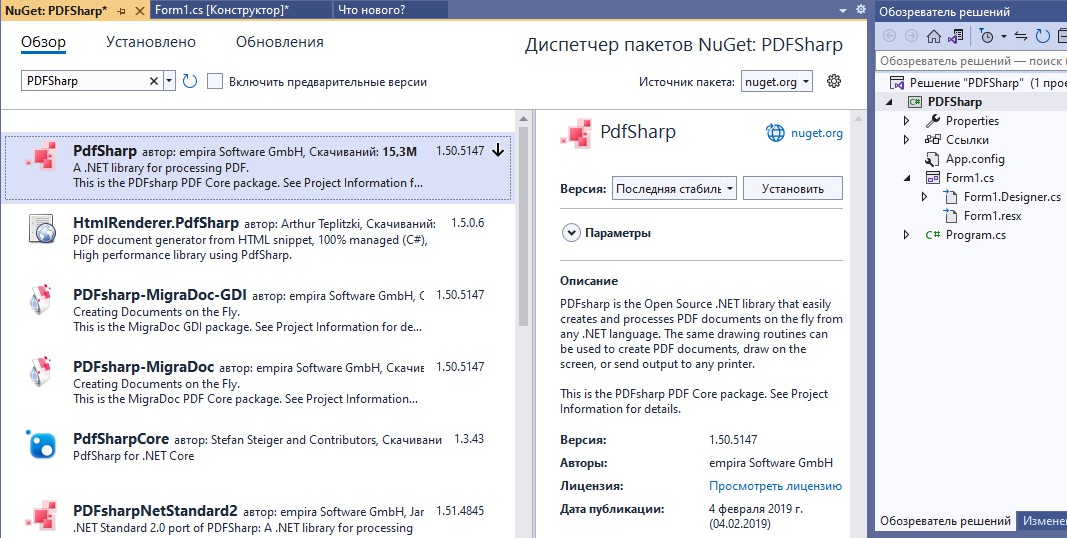
**Задание 1.1.** Создайте проект типа приложения Windows Forms C# .NET **(**имя проекта

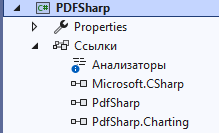
# пр15\_1\_Фамилия)

* Запустите **Visual Studio 2022** и выберите вариант создания нового проекта.
* Выберите шаблон проекта как приложение **Windows Form (.NET Framework)** для **C#** и нажмите кнопку «**Далее**».
* Укажите имя проекта, выберите место для сохранения проекта, выберите платформу как **.NET Framework 4.8** и нажмите кнопку «**Создать**».
* Проект должен быть создан и загружен в визуальную студию для модификации.

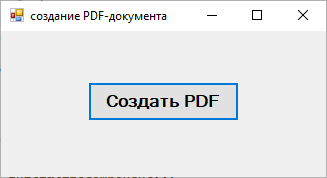
**Задание 1.2.** Добавьте ссылку на библиотеку **PDF Sharp**

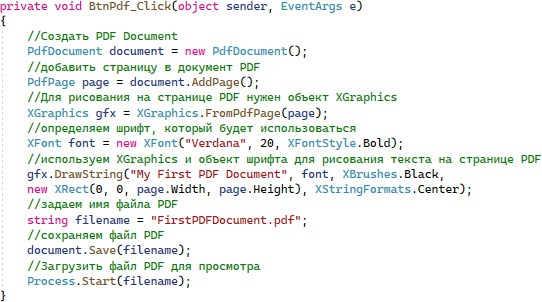
* Щелкните правой кнопкой мыши по названию файла проекта в обозревателе решений и выберите пункт меню «**Управление пакетами NuGet...**». Это должно открыть экран, показанный ниже, для управления NuGet.
* Затем найдите **PDF Sharp** на вкладке «**Обзор**» на экране «**Управление NuGet**» и после обнаружения необходимой библиотеки, т.е. **PdfSharp**, выберите и установите стабильную версию библиотеки PdfSharp.



* Поле выполнения установки должны добавиться ссылки на необходимые библиотеки PdfSharp в проекте.

# Задание 1.3. Добавьте код для создания PDF с помощью PDF Sharp.

* Создайте форму по образцу:
* Добавьте приведенный ниже код, чтобы создать файл Pdf на С# в событии нажатия кнопки:



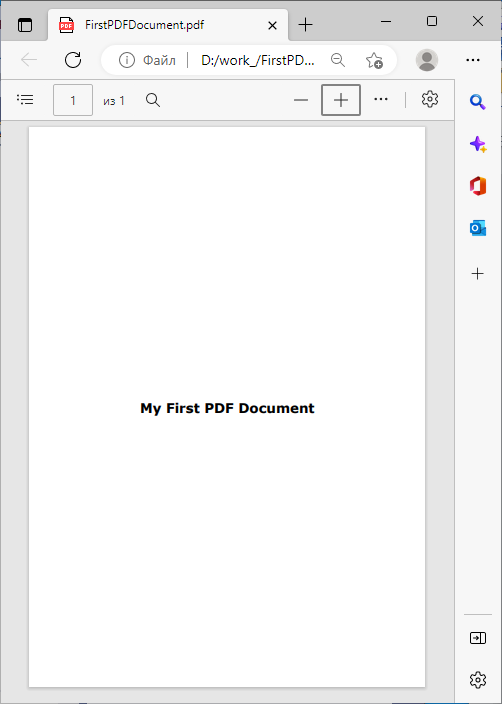
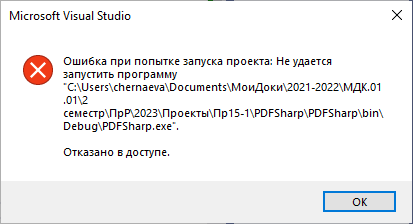
* Также нужно будет включить пространства имен для библиотеки и диагностики **PdfSharp**. Добавьте приведенный ниже код вверху в Form1.cs, чтобы импортировать необходимые пространства имен.

*using PdfSharp.Drawing; using PdfSharp.Pdf; using System.Diagnostics;*

# Задание 1.4 — Запустите и протестируйте код

* Нажмите **F5**, чтобы скомпилировать и запустить наш код. Это должно запустить форму с кнопкой «**Создать PDF**».
* Нажмите кнопку «**Создать PDF**». При успешном выполнении кода в событии нажатия кнопки в папке **bin/debug** будет создан файл **Pdf** с указанным именем **FirstPDFDocument.pdf**, который будет загружен в браузер (или программу просмотра) PDF.

***Примечание:*** *Если при запуске проекта, выводится ошибка:*

*, то в программном коде задайте путь для сохранения файла, например, папку с текущим проектом.*

# Задание 1.5 – Просмотр созданного PDF-файла

* Теперь, когда файл Pdf создан и загружен в средство просмотра PDF, можно просмотреть файл Pdf, который создается в соответствии с написанным кодом. Проверьте результат работы приложения.

***Пример 2. Организация печати в формах windows***

# Упражнение 1. Использование диалоговых окон для печати

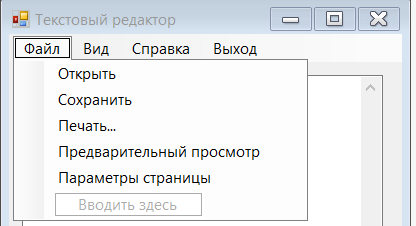
При печати различных документов пользователям часто приходится изменять параметры печати. Обычно они ограничиваются заданием таких параметров, как ориентация страницы, ширина полей и размер бумаги.

**.NET Framework** содержит классы, которые предоставляют пользователям возможность осуществлять и более сложные настройки

**Задание 2.1.** Откройте проект созданного текстового редактора (**пр14\_Прил1**)

* Откройте окно формы в режиме конструктора. В меню Файл добавьте новые подпункты

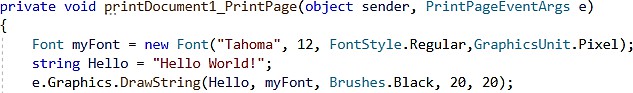
**Печать…, Предварительный просмотр, Параметры страницы**



# Задание 2.2. Добавление компонентов печати

* Добавьте в окно дизайнера форм элемент **PrintDocument** вкладки **Печать** панели элементов *Компонент* ***PrintDocument*** *предназначен для вывода данных документа на принтер. Свойства компонента* ***PrintDocument*** *описывают, как именно нужно распечатывать документ.*
* С помощью окна **Свойства** для компонента **printDocument1** добавьте обработчик события

**PrintPage** и внутри него добавьте следующий код:



* Добавьте в окно дизайнера форм компоненты **PrintDialog. PageSetupDialog** и

**PrintPreviewDialog** вкладки **Печать** панели элементов

*С помощью компонента* ***PrintDialog*** *приложение выведет на экран стандартное диалоговое окно печати документа.*

*Компоненты* ***PageSetupDialog*** *и* ***PrintPreviewDialog*** *предназначены для управления печатью*

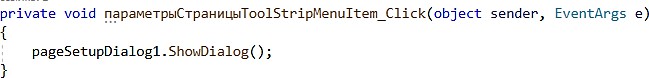
*пробного документа: заданием таких параметров, как ориентация страницы, ширина полей и размер бумаги, и предварительный просмотр документа.*

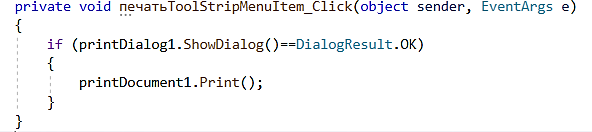
* Задайте для всех компонентов значение свойства **Document** равным **printDocument1**. *Этим обеспечивается связь компонента* ***printDialog*** *с компонентом* ***PrintDocument****.*
* Для элемента **PrintDialog** присвойте свойству **AllowSomePages** в значение **True**.

# Задание 2.3. Реализация вызова диалоговых окон

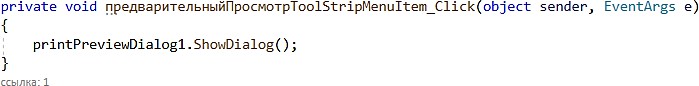
* Для работы с классами, предназначенными для выполнения операций с потоками и печати, добавьте в начало программы следующие строки:

using System.Drawing.Printing;

* должна быть добавлена на предыдущем занятии, проверьте ее наличие: using System.IO;
* Добавьте обработчик события для подпункта меню **Параметры страницы**: в конструкторе дважды щелкните подменю **Параметры страницы** и добавьте следующий код:
* Добавьте обработчик события для подпункта меню **Печать**:



* Добавьте обработчик события для подпункта меню **Предварительный просмотр**:



* Протестируйте приложение. Выберите каждый из подпунктов меню, чтобы проверить открытие различных диалоговых окон печати.

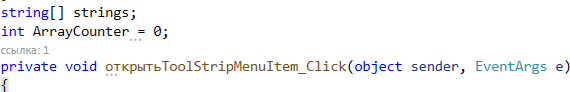
# Упражнение 2. Создание документа печати

**Задание 2.4.** Создайте обработчик приложения, разрешающий пользователю печатать содержимое открытого текстового файла:

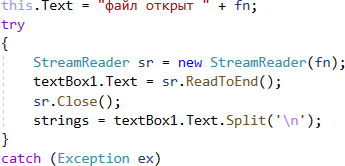
1. Измените свойства элемента **OpenFileDialog**: присвойте свойству **Filter** значение **Text Files |**

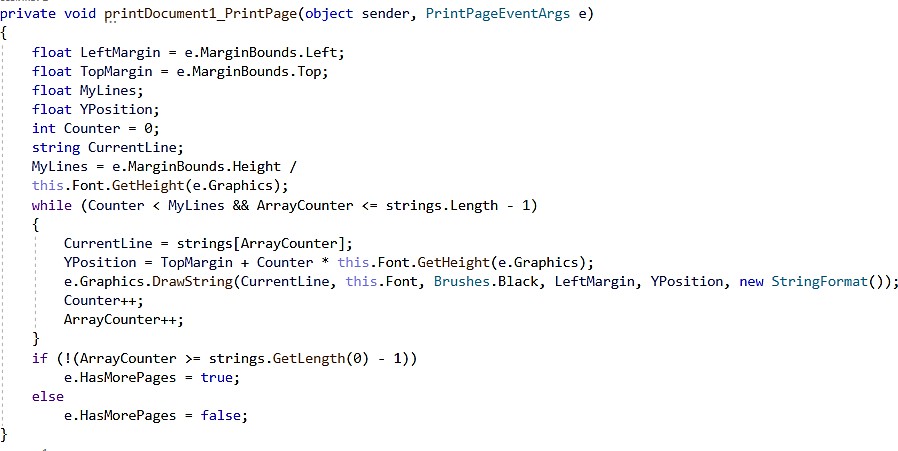
**\*.txt**, и очистите поле свойства **FileName**.

1. выше кода обработчика события **Click** открытия редактора **OpenFile** добавьте код:



1. И в коде обработчика события добавьте строку

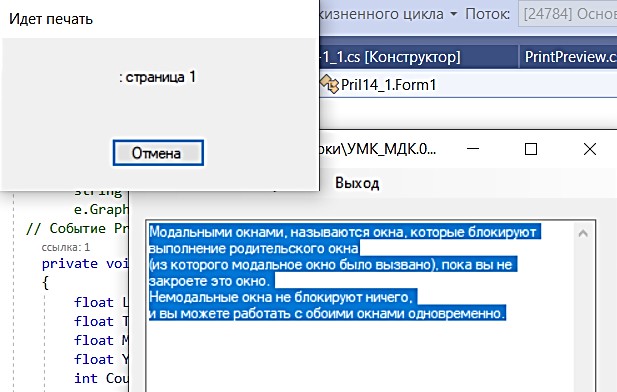


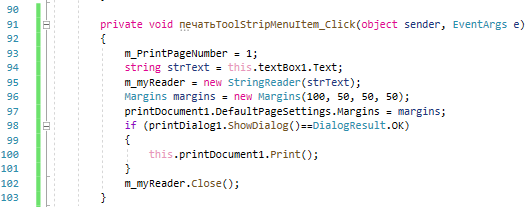
1. В обработчике события **printDocument1\_PrintPage** замените существующий код следующим:

## Примечание:

*Для полного понимания действий, выполняемых нашим обработчиком событий, требуется предварительное знакомство с графической подсистемой Graphics Device Interface Plus (GDI+), реализованной компанией Microsoft в рамках библиотеки классов .NET Framework. Пока же нужно отметить, что приложение распечатывает текст построчно в цикле. После завершения печати всех строк текущей страницы обработчик событий PrintPage печатает верхний и нижний колонтитулы, а также рисует горизонтальные линии, отделяющие текст колонтитулов от текста документа.*

1. Протестируйте приложение. Откройте текстовый файл на своем компьютере. Выберите **Предварительный просмотр** для просмотра файла в диалоговом окне. Напечатайте файл (если принтер подключен).



1. Создайте обработчик события печати документа.

*Печать документа будет начинаться с первой страницы, поэтому в поле m\_PrintPageNumber записывается значение 1. Далее выполняется чтение текущего содержимого окна редактирования текста в поток m\_myReader (при необходимости задайте для него тип* ***var*** *) класса StringReader. Далее задаются границы отступов на распечатываемой странице и отображается диалоговое окно печати документа. Если пользователь щелкает в этом окне кнопку OK, документ printDocument1 отправляется на печать методом Print. Далее ненужный более поток m\_myReader закрывается методом Close.*

На данном этапе приложение еще не в состоянии распечатать документ. Причина этого в том, что приложение пока еще не знает, каким именно образом нужно печатать документ.

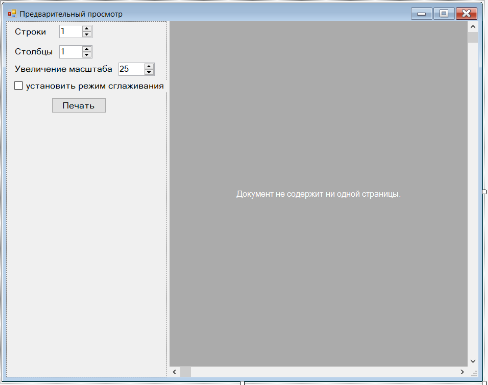
# Упражнение 3. Создание специализированной формы предварительного просмотра.

Хотя компонент **PrintPreviewDialog** является простым, удобным в работе способом предоставить в ваших приложениях функциональность предварительного просмотра, его трудно настроить. Для приложений со специализированным предварительным просмотром можно для создания специализированного компонента предварительного просмотра использовать элемент управления **PrintPreviewControl**.

**Задание 2.5** Создайте специализированную форму предварительного просмотра, дающую пользователю возможность указывать масштаб, количество строк и столбцов, а также

переключать режим сглаживания, добавив ее к решению, созданному в предыдущем упражнении:

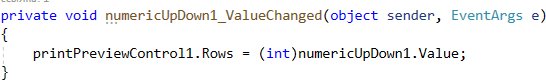
* Добавьте к проекту новую форму под именем **PrintPreview**.
* Из панели элементов перетащите **SplitContainer** в форму. В свойстве **Orientation** должна быть задана вертикаль.
* Из панели элементов перетащите **PrintPreviewControl** в **Раnе12** и присвойте свойству **Dock**

значение **Fill**.

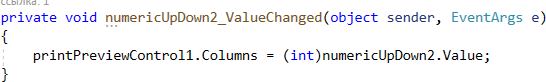
* Для **printPreviewControl1** присвойте свойству **Modifiers** значение **Internal**.
* Из панели элементов добавьте в **Раnеl1** три элемента управления **Label**, три **NumericUpDown** (надпись **Label** определяет назначение соответствующего элемента **NumericUpDown**), один **Checkbox** и один **Button**. Свяжите надписи с элементами управления **NumericUpDown** и установите свойства, как показано в следующей таблице:
* И дать соответствующие имена элементам

|  |  |  |
| --- | --- | --- |
| **Элемент управления** | **Свойство** | **Значение** |
| Label1 | Text | Строки |
| Label2 | Text | Столбцы |
| Label3 | Text | Увеличение масштаба |
| Label1 | Name | RowsLabel |
| Label2 | Name | ColumsLabel |
| Label3 | Name | ScaleLabel |
| NumericUpDown 1 | Minimum | 1 |
| NumericUpDown 2 | Minimum | 1 |
| NumericUpDown 3 | Minimum | 25 |
| NumericUpDown 1 | Maximum | 8 |
| NumericUpDown 2 | Maximum | 8 |
| NumericUpDown 3 | Maximum | 500 |
| NumericUpDown 3 | Increment | 25 |
| CheckBox1 | Text | Установить режим сглаживания |
| Buttonl | Text | Печать |
| Button1 | Name | PrintButton |

* Дважды щелкните **NumericUpDown1** и добавьте к обработчику события

**numericUpDown1\_ValueChanged** следующий код:

* В конструкторе дважды щелкните **NumericUpDown2** и добавьте к обработчику события

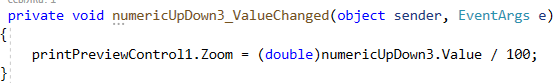
**numericUpDown2\_ValueChanged** следующий код:

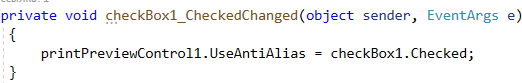
Свойство **Columns** указывает количество отображаемых страниц по горизонтали, а свойство **Rows**

– по вертикали.

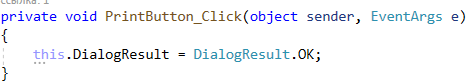
* В конструкторе дважды щелкните **NumericUpDown3** и добавьте к обработчику события

**numericUpDown3\_ValueChanged** следующий код

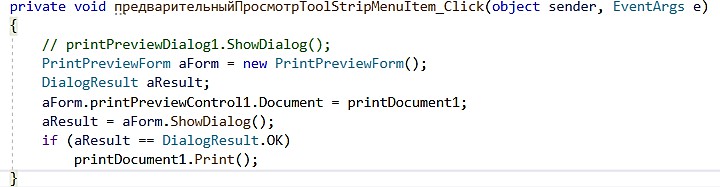


* В конструкторе дважды щелкните **CheckBox1** и добавьте к обработчику события checkBox1\_CheckedChanged следующий код:
* В конструкторе дважды щелкните **Печать** и добавьте к обработчику события

**PrintButton\_Click** следующий код



* В редакторе кода формы **Form1** закомментируйте код, существующий в обработчике события

**предварительныйПросмотрToolStripMenuItemClick**, и добавьте следующий:

* Протестируйте приложение. С помощью команды **Открыть** меню **Файл** откройте текстовый файл и затем щелкните **Предварительный просмотр** для проверки вашей новой формы предварительного просмотра.
* Сохраните все изменения в приложении.

# Контрольные вопросы:

* 1. PDFsharp: понятие и назначение.
  2. Опишите процесс создания файла PDF на C# .NET
  3. Опишите классы для обеспечения печати.