

# PDF Combiner

A simple Windows Forms app to combine multiple PDF files in a single PDF file

# Table of Contents

<b>Symbol Reference</b>	<b>1</b>
<b>PdfCombiner Namespace</b>	<b>1</b>
PdfCombiner.Properties Namespace	1
Classes	1
Resources Class	1
Settings Class	3
Classes	4
BaseException Class	4
BaseException Constructor	5
Enums Class	5
Enums Enumerations	6
ExceptionExtensions Class	6
ExceptionExtensions Methods	6
FileInfo Class	7
FileInfo Properties	7
FrmMain Class	8
FrmMain.FrmMain Constructor	10
FrmMain Fields	10
FrmMain Methods	13
Program Class	33
Program Methods	34
<b>Files</b>	<b>34</b>
AssemblyInfo.cs	35
BaseException.cs	35
Enums.cs	36
ExceptionExtensions.cs	36
FileInfo.cs	37
FrmMain.cs	37
FrmMain.Designer.cs	49
PdfCombiner.csproj	54
PdfCombiner.sln	54
Program.cs	54
Resources.Designer.cs	55
Settings.Designer.cs	56
<b>Index</b>	<b>a</b>

# 1 Symbol Reference







## 1.1 PdfCombiner Namespace

This is namespace PdfCombiner.

### Namespaces

Name	Description
Properties ( <a href="#">see page 1</a> )	This is namespace PdfCombiner.Properties.



### Classes

	Name	Description
	BaseException ( <a href="#">see page 4</a> )	This function is used to take exceptions which can be thrown in PDF Combiner App
	Enums ( <a href="#">see page 5</a> )	This class contains enums for detailed information In usage
	ExceptionExtensions ( <a href="#">see page 6</a> )	This is class PdfCombiner.ExceptionExtensions.
	FileInfo ( <a href="#">see page 7</a> )	This class is used to have File's Full Path and Name
	FrmMain ( <a href="#">see page 8</a> )	This is class PdfCombiner.FrmMain.
	Program ( <a href="#">see page 33</a> )	This is class PdfCombiner.Program.

### 1.1.1 PdfCombiner.Properties Namespace

This is namespace PdfCombiner.Properties.



### Classes

	Name	Description
	Resources ( <a href="#">see page 1</a> )	A strongly-typed resource class, for looking up localized strings, etc.
	Settings ( <a href="#">see page 3</a> )	This is class PdfCombiner.Properties.Settings.

#### 1.1.1.1 Classes

The following table lists classes in this documentation.

### Classes

	Name	Description
	Resources ( <a href="#">see page 1</a> )	A strongly-typed resource class, for looking up localized strings, etc.
	Settings ( <a href="#">see page 3</a> )	This is class PdfCombiner.Properties.Settings.

#### 1.1.1.1.1 Resources Class

A strongly-typed resource class, for looking up localized strings, etc.

## Class Hierarchy

[PdfCombiner.Properties.Resources](#)

### C#

```
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.StronglyTypedResourceBuilder",
"15.0.0.0")]
[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
internal class Resources;
```


### File

Resources.Designer.cs ([see page 55](#))



### Description

This class was auto-generated by the StronglyTypedResourceBuilder class via a tool like ResGen or Visual Studio. To add or remove a member, edit your .ResX file then rerun ResGen with the /str option, or rebuild your VS project.




### Methods

	Name	Description
	Resources ( <a href="#">see page 2</a> )	This is Resources, a member of class Resources.

### Resources Fields

	Name	Description
	resourceCulture ( <a href="#">see page 3</a> )	This is resourceCulture, a member of class Resources.
	resourceMan ( <a href="#">see page 3</a> )	This is resourceMan, a member of class Resources.

### Resources Properties

	Name	Description
	AppTitle ( <a href="#">see page 3</a> )	Looks up a localized string similar to PDF Birlestirici.
	Culture ( <a href="#">see page 3</a> )	Overrides the current thread's CurrentUICulture property for all resource lookups using this strongly typed resource class.
	ResourceManager ( <a href="#">see page 3</a> )	Returns the cached ResourceManager instance used by this class.

#### 1.1.1.1.1 Resources.Resources Constructor

### C#

```
[global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1811:AvoidUncalledPrivateCode")]
internal Resources();
```

### Description

This is Resources, a member of class Resources.

### Body Source

```
1:
[global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1811:AvoidUncalledPrivateCode")]
2: internal Resources() {
3: }
```

#### 1.1.1.1.2 Resources Fields

#### 1.1.1.1.1.2.1 Resources.resourceCulture Field

**C#**

```
private static global::System.Globalization.CultureInfo resourceCulture;
```

##### Description

This is resourceCulture, a member of class Resources.

#### 1.1.1.1.1.2.2 Resources.resourceMan Field

**C#**

```
private static global::System.Resources.ResourceManager resourceMan;
```

##### Description

This is resourceMan, a member of class Resources.

### 1.1.1.1.1.3 Resources Properties

#### 1.1.1.1.1.3.1 Resources.AppTitle Property

Looks up a localized string similar to PDF Birlestirici.

**C#**

```
internal static string AppTitle;
```

#### 1.1.1.1.1.3.2 Resources.Culture Property

Overrides the current thread's CurrentUICulture property for all resource lookups using this strongly typed resource class.

**C#**

```
[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]
internal static global::System.Globalization.CultureInfo Culture;
```

#### 1.1.1.1.1.3.3 Resources.ResourceManager Property

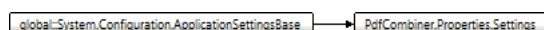
Returns the cached ResourceManager instance used by this class.

**C#**

```
[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]
internal static global::System.Resources.ResourceManager ResourceManager;
```

### 1.1.1.1.2 Settings Class

#### Class Hierarchy



**C#**

```
[global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.VisualStudio.Editors.SettingsDesigner.SettingsSingleFileGenerator",
"11.0.0.0")]
internal sealed class Settings : global::System.Configuration.ApplicationSettingsBase;
```


#### File

Settings.Designer.cs (🔗 see page 56)

### Description

This is class PdfCombiner.Properties.Settings.

### Settings Fields

	Name	Description
	defaultInstance ( <a href="#">see page 4</a> )	This is defaultInstance, a member of class Settings.

## 1.1.1.1.2.1 Settings Fields

### 1.1.1.1.2.1.1 Settings.defaultInstance Field

#### C#

```
private static Settings defaultInstance =  
((Settings)(global::System.Configuration.ApplicationSettingsBase.Synchronized(new  
Settings())));
```







### Description

This is defaultInstance, a member of class Settings.

## 1.1.2 Classes

The following table lists classes in this documentation.

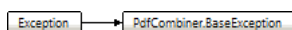
### Classes

	Name	Description
	BaseException ( <a href="#">see page 4</a> )	This function is used to take exceptions which can be thrown in PDF Combiner App
	Enums ( <a href="#">see page 5</a> )	This class contains enums for detailed information In usage
	ExceptionExtensions ( <a href="#">see page 6</a> )	This is class PdfCombiner.ExceptionExtensions.
	FileInfo ( <a href="#">see page 7</a> )	This class is used to have File's Full Path and Name
	FrmMain ( <a href="#">see page 8</a> )	This is class PdfCombiner.FrmMain.
	Program ( <a href="#">see page 33</a> )	This is class PdfCombiner.Program.

## 1.1.2.1 BaseException Class

This function is used to take exceptions which can be thrown in PDF Combiner App

### Class Hierarchy





#### C#

```
[Serializable]  
public abstract class BaseException : Exception;
```

#### File

BaseException.cs ([see page 35](#))

**Methods**

	Name	Description
	BaseException (  see page 5)	This is BaseException, a member of class BaseException.

## 1.1.2.1.1 BaseException Constructor

### 1.1.2.1.1.1 BaseException.BaseException Constructor ()

**C#**

```
protected BaseException();
```

**Description**

This is BaseException, a member of class BaseException.

**Body Source**

```
1: protected BaseException()  
2: {  
3: }
```

### 1.1.2.1.1.2 BaseException.BaseException Constructor (string)

**C#**

```
protected BaseException(string message);
```

**Description**

This is BaseException, a member of class BaseException.

**Body Source**

```
1: protected BaseException(string message) : base(message)  
2: {  
3: }
```

### 1.1.2.1.1.3 BaseException.BaseException Constructor (string, Exception)

**C#**

```
protected BaseException(string message, Exception innerException);
```

**Description**

This is BaseException, a member of class BaseException.


**Body Source**

```
1: protected BaseException(string message, Exception innerException) : base(message,  
innerException)  
2: {  
3: }
```

## 1.1.2.2 Enums Class

This class contains enums for detailed information In usage

**Class Hierarchy**

 PdfCombiner.Enums

**C#**

```
public static class Enums;
```

**File**

Enums.cs ([see page 36](#))

**Enums Enumerations**

	Name	Description
	OrderType ( <a href="#">see page 6</a> )	This is record PdfCombiner.Enums.OrderType.

## 1.1.2.2.1 Enums Enumerations

### 1.1.2.2.1.1 PdfCombiner.Enums.OrderType Enumeration

**C#**

```
[Flags]
public enum OrderType {
    Ascending,
    Descending
}
```

**File**

Enums.cs ([see page 36](#))

**Description**

This is record PdfCombiner.Enums.OrderType.

## 1.1.2.3 ExceptionExtensions Class

**Class Hierarchy**

[PdfCombiner.ExceptionExtensions](#)

**C#**

```
public static class ExceptionExtensions;
```


**File**

ExceptionExtensions.cs ([see page 36](#))

**Description**

This is class PdfCombiner.ExceptionExtensions.

**ExceptionExtensions Methods**

	Name	Description
	GetAllMessages ( <a href="#">see page 6</a> )	This method is used to take all messages with stack trace info in exception

## 1.1.2.3.1 ExceptionExtensions Methods

### 1.1.2.3.1.1 ExceptionExtensions.GetAllMessages Method

This method is used to take all messages with stack trace info in exception

**C#**

```
public static string GetAllMessages(this Exception exception);
```



**Parameters**

Parameters	Description
this Exception exception	Exception

**Returns**

Detailed Error Message

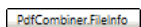
**Body Source**

```
1: public static string GetAllMessages(this Exception exception)
2: {
3:     var sb = new StringBuilder();
4:     sb.Append(exception.Message);
5:     if (exception.InnerException != null)
6:         sb.Append($" {exception.InnerException.GetAllMessages()}");
7:     if (!string.IsNullOrEmpty(exception.StackTrace))
8:         sb.Append($" {exception.StackTrace}");
9:     if (!string.IsNullOrEmpty(exception.Source))
10:        sb.Append($" {exception.Source}");
11:     return sb.ToString();
12: }
```

## 1.1.2.4 FileInfo Class

This class is used to have File's Full Path and Name

**Class Hierarchy**

 PdfCombiner.FileInfo





**C#**

```
public class FileInfo;
```

**File**

FileInfo.cs ( see page 37)

**FileInfo Properties**

	Name	Description
	FileName (  see page 7)	File Name Property
	FilePath (  see page 7)	File Path Property

### 1.1.2.4.1 FileInfo Properties

#### 1.1.2.4.1.1 FileInfo.FileName Property

File Name Property

**C#**

```
public string FileName;
```

#### 1.1.2.4.1.2 FileInfo.FilePath Property

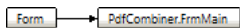
File Path Property

**C#**

```
public string FilePath;
```

## 1.1.2.5 FrmMain Class

### Class Hierarchy



### C#

```
public class FrmMain : Form;
```

### File

FrmMain.Designer.cs (see page 49)

### Description

This is class PdfCombiner.FrmMain.

### Methods







	Name	Description
	FrmMain (see page 10)	This is FrmMain, a member of class FrmMain.

### FrmMain Fields

	Name	Description
	_resource (see page 10)	This is _resource, a member of class FrmMain.
	btnAddFile (see page 11)	This is btnAddFile, a member of class FrmMain.
	btnAddFolder (see page 11)	This is btnAddFolder, a member of class FrmMain.
	btnClearList (see page 11)	This is btnClearList, a member of class FrmMain.
	btnCombineITextSharp (see page 11)	This is btnCombineITextSharp, a member of class FrmMain.
	btnCombinePdfSharp (see page 11)	This is btnCombinePdfSharp, a member of class FrmMain.
	cmbLanguage (see page 11)	This is cmbLanguage, a member of class FrmMain.
	components (see page 12)	Required designer variable.
	LanguageList (see page 12)	Culture lists whose resources have been added to project
	lbFiles (see page 12)	This is lbFiles, a member of class FrmMain.
	lblDetails (see page 12)	This is lblDetails, a member of class FrmMain.
	menuItemDelete (see page 12)	This is menuItemDelete, a member of class FrmMain.
	menuItemOrderByNameAscending (see page 12)	This is menuItemOrderByNameAscending, a member of class FrmMain.
	menuItemOrderByNameDescending (see page 12)	This is menuItemOrderByNameDescending, a member of class FrmMain.
	menuItemOrderByPathAscending (see page 13)	This is menuItemOrderByPathAscending, a member of class FrmMain.
	menuItemOrderByPathDescending (see page 13)	This is menuItemOrderByPathDescending, a member of class FrmMain.
	menuStrip (see page 13)	This is menuStrip, a member of class FrmMain.
	pbFiles (see page 13)	This is pbFiles, a member of class FrmMain.

### FrmMain Methods

	Name	Description
	AddPdfContentToTargetPdfFileITextSharp (see page 13)	This method is used to take source pdf file content And add to the target combined pdf file

	AddPdfContentToTargetPdfFilePdfSharp ( <a href="#">see page 14</a> )	This method is used to take source pdf file content And add to the target combined pdf file
	AddPdfFilesToList ( <a href="#">see page 14</a> )	This method is used both adding files or folders to list
	btnAddFile_Click ( <a href="#">see page 15</a> )	It is used to add single or multiple PDF files to combine When you choose file or files The listbox in the form will be filled with the name of the files Which you choose to combine
	btnAddFolder_Click ( <a href="#">see page 15</a> )	This function is used to add PDF files in a folder which you choose in Folder Dialog recursively When you use it the added files will be seen in listbox
	btnClearList_Click ( <a href="#">see page 16</a> )	This function is used to clear the file list in listbox
	btnCombineITextSharp_Click ( <a href="#">see page 16</a> )	This function is used to take list of PDF files in listbox, Combine them with a single file in location which you choose in folder dialog And you can see the progress in progress bar It uses iTextSharp Nuget Package
	btnCombinePdfSharp_Click ( <a href="#">see page 17</a> )	This function is used to take list of PDF files in listbox, Combine them with a single file in location which you choose in folder dialog And you can see the progress in progress bar It uses PdfSharp Nuget Package
	cmbLanguage_SelectedIndexChanged ( <a href="#">see page 18</a> )	This function is used to set the _resource ( <a href="#">see page 10</a> ) file And with this, you can see form elements, info messages etc. with the selected _resource ( <a href="#">see page 10</a> ) file contents. That is used to multi language support
	DeleteFilesFromListBox ( <a href="#">see page 18</a> )	This method is used to Delete selected files from ListBox This is used different ways in this app So we created this as a method
	Dispose ( <a href="#">see page 19</a> )	Clean up any resources being used.
	FillLanguageComboboxAndSetFirstLanguage ( <a href="#">see page 19</a> )	This method is used to Fill the Language combobox And set initial value with The language computer used in
	FormMembersNameInitialization ( <a href="#">see page 20</a> )	This method is used to change text of form members by the Resource file which is selected
	FrmMain_FormClosed ( <a href="#">see page 20</a> )	This function is used to terminate application
	FrmMain_Shown ( <a href="#">see page 20</a> )	This function is used to assign menu to listbox When form is shown
	GenerateAddFileMessage ( <a href="#">see page 21</a> )	This method is used to show info message after adding files or folders
	GenerateCombineFileMessage ( <a href="#">see page 21</a> )	This method is used to set successfull combining file message And set value of Form Caption etc.
	GenerateCombineWarningMessage ( <a href="#">see page 22</a> )	This method is used When user just want to combine 1 pdf file
	GenerateDeleteItemMessage ( <a href="#">see page 22</a> )	This method is used after deleting items from listbox
	GenerateExceptionMessage ( <a href="#">see page 22</a> )	This method is used to Show error message when exception occurs and set form title initial value
	GenerateNoFileInListBoxMessage ( <a href="#">see page 22</a> )	This method is used to generate message When there is no item in listbox
	GenerateNoSelectedFileInListBoxMessage ( <a href="#">see page 23</a> )	This method is used in When there is no selected file in listbox
	InitializeComponent ( <a href="#">see page 23</a> )	Required method for Designer support - do not modify the contents of this method with the code editor.
	InitializeFileDialog ( <a href="#">see page 26</a> )	This function is used to set options of file dialog like Filter, start path etc.
	lbFiles_DragDrop ( <a href="#">see page 27</a> )	This is used to drag and drop an item in listbox Which is used to order books and combine them with that order
	lbFiles_DragOver ( <a href="#">see page 27</a> )	This is used to give effect while dragging and dropping an item in listbox

	<code>IbFiles_KeyDown</code> ( <a href="#">see page 28</a> )	This function is used to operate some process in files in listbox which are chosen
	<code>IbFiles_MouseDown</code> ( <a href="#">see page 28</a> )	This is used to control mouse actions and if that is not right click Start to drag and drop item in listbox
	<code>menuItemDelete_Click</code> ( <a href="#">see page 28</a> )	This is used to delete selected items in listbox When menu item Delete is clicked and opened dialog
	<code>menuItemOrderByNameAscending_Click</code> ( <a href="#">see page 29</a> )	This method is used to order elements in listbox ascending by file name
	<code>menuItemOrderByNameDescending_Click</code> ( <a href="#">see page 29</a> )	This method is used to order elements in listbox descending by file name
	<code>menuItemOrderByPathAscending_Click</code> ( <a href="#">see page 30</a> )	This method is used to order elements in listbox ascending by full path
	<code>menuItemOrderByPathDescending_Click</code> ( <a href="#">see page 30</a> )	This method is used to order elements in listbox descending by full path
 	<code>SaveOutputPdfFile</code> ( <a href="#">see page 30</a> )	This method is used to Save Combined PDF File
	<code>SetCombinationRatio</code> ( <a href="#">see page 31</a> )	This method is used to set combination degree while combining PDF files And show this ratio on Form Caption
	<code>SetInitialValuesForCombiningFiles</code> ( <a href="#">see page 31</a> )	This method is used to set initial values of some properties Which are used in combining PDF files In both PdfSharp and ITextSharp
 	<code>SetOutputFilePropertiesPdfSharp</code> ( <a href="#">see page 31</a> )	This method is used to set combined PDF file Properties ( <a href="#">see page 1</a> ) like Compression etc.
	<code>SetProgramLanguage</code> ( <a href="#">see page 32</a> )	This method is used to Set Program ( <a href="#">see page 33</a> ) Language By the selection from Language Combobox Finally, with the selected language Form elements renamed with the selected language
	<code>ShowDeleteDialog</code> ( <a href="#">see page 32</a> )	This method is used to Show delete file dialog
 	<code>SortItemsByName</code> ( <a href="#">see page 32</a> )	This method is used to order items by file name in listbox And return them with the given order as parametre
 	<code>SortItemsByPath</code> ( <a href="#">see page 33</a> )	This method is used to order items by full path in listbox And return them with the given order as parametre

### 1.1.2.5.1 FrmMain.FrmMain Constructor

C#

```
public FrmMain();
```

#### Description

This is FrmMain, a member of class FrmMain.

#### Body Source

```
1: public FrmMain()
2: {
3:     InitializeComponent();
4: }
```

### 1.1.2.5.2 FrmMain Fields

#### 1.1.2.5.2.1 FrmMain.\_resource Field

C#

```
private static ResourceManager _resource;
```

**Description**

This is \_resource, a member of class FrmMain.

**1.1.2.5.2.2 FrmMain.btnAddFile Field****C#**

```
private System.Windows.Forms.Button btnAddFile;
```

**Description**

This is btnAddFile, a member of class FrmMain.

**1.1.2.5.2.3 FrmMain.btnAddFolder Field****C#**

```
private System.Windows.Forms.Button btnAddFolder;
```

**Description**

This is btnAddFolder, a member of class FrmMain.

**1.1.2.5.2.4 FrmMain.btnClearList Field****C#**

```
private System.Windows.Forms.Button btnClearList;
```

**Description**

This is btnClearList, a member of class FrmMain.

**1.1.2.5.2.5 FrmMain.btnCombineITextSharp Field****C#**

```
private System.Windows.Forms.Button btnCombineITextSharp;
```

**Description**

This is btnCombineITextSharp, a member of class FrmMain.

**1.1.2.5.2.6 FrmMain.btnCombinePdfSharp Field****C#**

```
private System.Windows.Forms.Button btnCombinePdfSharp;
```

**Description**

This is btnCombinePdfSharp, a member of class FrmMain.

**1.1.2.5.2.7 FrmMain.cmbLanguage Field****C#**

```
public System.Windows.Forms.ComboBox cmbLanguage;
```

**Description**

This is cmbLanguage, a member of class FrmMain.

#### 1.1.2.5.2.8 FrmMain.components Field

Required designer variable.

**C#**

```
private System.ComponentModel.IContainer components;
```

#### 1.1.2.5.2.9 FrmMain.LanguageList Field

**C#**

```
private static readonly List<string> LanguageList = new List<string> { "EN", "TR", "DE",  
"FR", "RU", "ES", "IT", "ZH", "AR", "NL", "PT", "IN", "ID", "JP", "BG" };
```

##### Description

Culture lists whose resources have been added to project

#### 1.1.2.5.2.10 FrmMain.lbFiles Field

**C#**

```
private System.Windows.Forms.ListBox lbFiles;
```

##### Description

This is lbFiles, a member of class FrmMain.

#### 1.1.2.5.2.11 FrmMain.lblDetails Field

**C#**

```
private System.Windows.Forms.Label lblDetails;
```

##### Description

This is lblDetails, a member of class FrmMain.

#### 1.1.2.5.2.12 FrmMain.menuItemDelete Field

**C#**

```
private System.Windows.Forms.ToolStripItem menuItemDelete;
```

##### Description

This is menuItemDelete, a member of class FrmMain.

#### 1.1.2.5.2.13 FrmMain.menuItemOrderByNameAscending Field

**C#**

```
private System.Windows.Forms.ToolStripItem menuItemOrderByNameAscending;
```

##### Description

This is menuItemOrderByNameAscending, a member of class FrmMain.

#### 1.1.2.5.2.14 FrmMain.menuItemOrderByNameDescending Field

**C#**

```
private System.Windows.Forms.ToolStripItem menuItemOrderByNameDescending;
```

##### Description

This is menuItemOrderByNameDescending, a member of class FrmMain.

### 1.1.2.5.2.15 FrmMain.menuItemOrderByPathAscending Field

C#

```
private System.Windows.Forms.ToolStripItem menuItemOrderByPathAscending;
```

#### Description

This is menuItemOrderByPathAscending, a member of class FrmMain.

### 1.1.2.5.2.16 FrmMain.menuItemOrderByPathDescending Field

C#

```
private System.Windows.Forms.ToolStripItem menuItemOrderByPathDescending;
```

#### Description

This is menuItemOrderByPathDescending, a member of class FrmMain.

### 1.1.2.5.2.17 FrmMain.menuStrip Field

C#

```
private System.Windows.Forms.ContextMenuStrip menuStrip;
```

#### Description

This is menuStrip, a member of class FrmMain.

### 1.1.2.5.2.18 FrmMain.pbFiles Field

C#

```
private System.Windows.Forms.ProgressBar pbFiles;
```

#### Description

This is pbFiles, a member of class FrmMain.

## 1.1.2.5.3 FrmMain Methods

### 1.1.2.5.3.1 FrmMain.AddPdfContentToTargetPdfFileITextSharp Method

This method is used to take source pdf file content And add to the target combined pdf file

C#

```
private static void AddPdfContentToTargetPdfFileITextSharp(object t, PdfCopy pdfWriter, ref int combinedFiles);
```

#### Parameters

Parameters	Description
object t	Source PDF File Path
PdfCopy pdfWriter	PDF Writer
ref int combinedFiles	Combined File Count

#### Body Source

```
1: private static void AddPdfContentToTargetPdfFileITextSharp(object t, PdfCopy pdfWriter,
2: ref int combinedFiles)
3: {
4:     var pdfReader =
5:         new iTextSharp.text.pdf.PdfReader(t.ToString());
```

```
5: pdfReader.ConsolidateNamedDestinations();
6: for (var j = 1; j <= pdfReader.NumberOfPages; j++)
7: {
8:     var page = pdfWriter.GetImportedPage(pdfReader, j);
9:     pdfWriter.AddPage(page);
10: }
11:
12: pdfReader.Close();
13: combinedFiles++;
14: }
```

### 1.1.2.5.3.2 FrmMain.AddPdfContentToTargetPdfFilePdfSharp Method

This method is used to take source pdf file content And add to the target combined pdf file

**C#**

```
private static void AddPdfContentToTargetPdfFilePdfSharp(object t, PdfDocument outputFile,
ref int combinedFiles);
```

**Parameters**

Parameters	Description
object t	Source PDF File Path
PdfDocument outputFile	Combined PDF File
ref int combinedFiles	Combined File Count

**Body Source**

```
1: private static void AddPdfContentToTargetPdfFilePdfSharp(object t, PdfDocument
outputFile, ref int combinedFiles)
2: {
3:     var inputDocument = PdfReader.Open(t.ToString(),
4:     PdfDocumentOpenMode.Import);
5:     var count = inputDocument.PageCount;
6:     for (var idx = 0; idx < count; idx++)
7:     {
8:         var page = inputDocument.Pages[idx];
9:         outputFile.AddPage(page);
10:     }
11:
12:     inputDocument.Close();
13:     combinedFiles++;
14: }
```

### 1.1.2.5.3.3 FrmMain.AddPdfFilesToList Method

This method is used both adding files or folders to list

**C#**

```
private void AddPdfFilesToList(ICollection<string> fileNames, ref int addedFileCount);
```

**Parameters**

Parameters	Description
ICollection<string> fileNames	List Of File Names
ref int addedFileCount	Added File Count For Progress Bar

**Body Source**

```
1: private void AddPdfFilesToList(ICollection<string> fileNames, ref int addedFileCount)
2: {
3:     foreach (var file in fileNames)
4:     {
5:         if (lbFiles.Items.Contains(file)) continue;
6:         lbFiles.Items.Add(file);
7:         addedFileCount++;
8:     }
```



```

9:         pbFiles.Value = addedFileCount * 100 / fileNames.Count;
10:         if (pbFiles.Value > pbFiles.Maximum)
11:             pbFiles.Value = pbFiles.Maximum;
12:         if (ActiveForm != null) ActiveForm.Text = @"%" + pbFiles.Value;
13:     }
14: }

```

#### 1.1.2.5.3.4 FrmMain.btnAddFile\_Click Method

It is used to add single or multiple PDF files to combine When you choose file or files The listbox in the form will be filled with the name of the files Which you choose to combine

**C#**

```
private void btnAddFile_Click(object sender, EventArgs e);
```

##### Parameters

Parameters	Description
object sender	The sender info (For example Main Form)
EventArgs e	Event Arguments

##### Body Source

```

1: private void btnAddFile_Click(object sender, EventArgs e)
2: {
3:     using (var dialogAddFile = new OpenFileDialog())
4:     {
5:         InitializeFileDialog(dialogAddFile);
6:
7:         var result = dialogAddFile.ShowDialog();
8:         var addedFileCount = 0;
9:         if (result != DialogResult.OK || dialogAddFile.FileNames == null) return;
10:
11:         AddPdfFilesToList(dialogAddFile.FileNames, ref addedFileCount);
12:         GenerateAddFileMessage(addedFileCount);
13:     }
14: }

```

#### 1.1.2.5.3.5 FrmMain.btnAddFolder\_Click Method

This function is used to add PDF files in a folder which you choose in Folder Dialog recursively When you use it the added files will be seen in listbox

**C#**

```
private void btnAddFolder_Click(object sender, EventArgs e);
```

##### Parameters

Parameters	Description
object sender	The sender info (For example Main Form)
EventArgs e	Event Arguments

##### Body Source

```

1: private void btnAddFolder_Click(object sender, EventArgs e)
2: {
3:     using (var dialogAddFolder = new FolderBrowserDialog())
4:     {
5:         var result = dialogAddFolder.ShowDialog();
6:         if (result != DialogResult.OK ||
string.IsNullOrEmpty(dialogAddFolder.SelectedPath)) return;
7:         var fileNames = Directory.GetFiles(dialogAddFolder.SelectedPath, "*.pdf",
8:             SearchOption.AllDirectories);
9:         var addedFileCount = 0;
10:
11:         AddPdfFilesToList(fileNames, ref addedFileCount);

```

```

12:         GenerateAddFileMessage(addedFileCount);
13:     }
14: }

```

### 1.1.2.5.3.6 FrmMain.btnClearList\_Click Method

This function is used to clear the file list in listbox

**C#**

```
private void btnClearList_Click(object sender, EventArgs e);
```

**Parameters**

Parameters	Description
object sender	The sender info (For example Main Form)
EventArgs e	Event Arguments

**Body Source**

```

1: private void btnClearList_Click(object sender, EventArgs e)
2: {
3:     var fileCount = lbFiles.Items.Count;
4:     lbFiles.Items.Clear();
5:     GenerateDeleteItemMessage(fileCount);
6: }

```

### 1.1.2.5.3.7 FrmMain.btnCombineITextSharp\_Click Method

This function is used to take list of PDF files in listbox, Combine them with a single file in location which you choose in folder dialog And you can see the progress in progress bar It uses iTextSharp Nuget Package

**C#**

```
private void btnCombineITextSharp_Click(object sender, EventArgs e);
```

**Parameters**

Parameters	Description
object sender	The sender info (For example Main Form)
EventArgs e	Event Arguments

**Body Source**

```

1: private void btnCombineITextSharp_Click(object sender, EventArgs e)
2: {
3:     try
4:     {
5:         if (lbFiles.Items.Count < 2)
6:             GenerateCombineWarningMessage();
7:         else
8:         {
9:             using (var dialogExport = new FolderBrowserDialog())
10:            {
11:                var result = dialogExport.ShowDialog();
12:                if (result != DialogResult.OK ||
13:                    string.IsNullOrEmpty(dialogExport.SelectedPath)) return;
14:                SetInitialValuesForCombiningFiles(dialogExport, out var outputFileName,
15:                out var fileCount,
16:                out var combinedFiles);
17:                var outputFile = new Document();
18:                using (var outputFileStream = new FileStream(outputFileName,
19:                    FileMode.Create))
20:                {
21:                    using (var pdfWriter = new PdfCopy(outputFile, outputFileStream))
22:                    {
23:                        pdfWriter.SetFullCompression();

```

```

23:         outputFile.Open();
24:         foreach (var t in lbFiles.Items)
25:         {
26:             try
27:             {
28:                 AddPdfContentToTargetPdfFileITextSharp(t, pdfWriter,
ref combinedFiles);
29:             }
30:             catch (Exception)
31:             {
32:                 fileCount--;
33:             }
34:             SetCombinationRatio(combinedFiles, fileCount);
35:         }
36:         pdfWriter.Close();
37:     }
38:     outputFile.Close();
39: }
40: }
41: }
42: }
43: }
44: GenerateCombineFileMessage(fileCount, outputFileName);
45: }
46: }
47: }
48: catch (Exception ex)
49: {
50:     GenerateExceptionMessage(ex);
51: }
52: }

```

#### 1.1.2.5.3.8 FrmMain.btnCombinePdfSharp\_Click Method

This function is used to take list of PDF files in listbox, Combine them with a single file in location which you choose in folder dialog And you can see the progress in progress bar It uses PdfSharp Nuget Package

C#

```
private void btnCombinePdfSharp_Click(object sender, EventArgs e);
```

Parameters

Parameters	Description
object sender	The sender info (For example Main Form)
EventArgs e	Event Arguments

Body Source

```

1: private void btnCombinePdfSharp_Click(object sender, EventArgs e)
2: {
3:     try
4:     {
5:         if (lbFiles.Items.Count < 2)
6:             GenerateCombineWarningMessage();
7:         else
8:         {
9:             using (var dialogExport = new FolderBrowserDialog())
10:            {
11:                var result = dialogExport.ShowDialog();
12:                if (result != DialogResult.OK ||
string.IsNullOrEmpty(dialogExport.SelectedPath)) return;
13:            }
14:            SetInitialValuesForCombiningFiles(dialogExport, out var outputFileName,
out var fileCount,
15:                out var combinedFiles);
16:            using (var outputFile = new PdfDocument())
17:            {

```

```

18:         {
19:             SetOutputFilePropertiesPdfSharp(outputFile);
20:
21:             foreach (var t in lbFiles.Items)
22:             {
23:                 try
24:                 {
25:                     AddPdfContentToTargetPdfFilePdfSharp(t, outputFile, ref
combinedFiles);
26:                 }
27:                 catch (Exception)
28:                 {
29:                     fileCount--;
30:                 }
31:
32:                 SetCombinationRatio(combinedFiles, fileCount);
33:             }
34:
35:             SaveOutputPdfFile(outputFile, outputFileName);
36:         }
37:
38:         GenerateCombineFileMessage(fileCount, outputFileName);
39:     }
40: }
41: }
42: catch (Exception ex)
43: {
44:     GenerateExceptionMessage(ex);
45: }
46: }

```

#### 1.1.2.5.3.9 FrmMain.cmbLanguage\_SelectedIndexChanged Method

This function is used to set the \_resource (see page 10) file And with this, you can see form elements, info messages etc. with the selected \_resource (see page 10) file contents. That is used to multi language support

**C#**

```
private void cmbLanguage_SelectedIndexChanged(object sender, EventArgs e);
```

**Parameters**

Parameters	Description
object sender	The sender info (For example Main Form)
EventArgs e	Event Arguments

**Body Source**

```

1: private void cmbLanguage_SelectedIndexChanged(object sender, EventArgs e)
2: {
3:     if (cmbLanguage.SelectedIndex == -1) return;
4:     SetProgramLanguage();
5: }

```

#### 1.1.2.5.3.10 FrmMain.DeleteFilesFromListBox Method

This method is used to Delete selected files from ListBox This is used different ways in this app So we created this as a method

**C#**

```
private void DeleteFilesFromListBox();
```

**Body Source**

```

1: private void DeleteFilesFromListBox()
2: {
3:     var fileCount = lbFiles.SelectedItems.Count;
4:     var removedItems = (from object t in lbFiles.SelectedItems select

```

```

t.ToString()).ToList();
5:
6:     var allItems = (from object item in lbFiles.Items where removedItems.All(j => j !=
item.ToString()) select item.ToString()).ToList();
7:
8:     lbFiles.Items.Clear();
9:     foreach (var item in allItems)
10:    {
11:        lbFiles.Items.Add(item);
12:    }
13:
14:     GenerateDeleteItemMessage(fileCount);
15: }

```

### 1.1.2.5.3.11 FrmMain.Dispose Method

Clean up any resources being used.

**C#**

```
protected override void Dispose(bool disposing);
```

**Parameters**

Parameters	Description
bool disposing	true if managed resources should be disposed; otherwise, false.

**Body Source**

```

1: protected override void Dispose(bool disposing)
2: {
3:     if (disposing && (components != null))
4:     {
5:         components.Dispose();
6:     }
7:     base.Dispose(disposing);
8: }

```

### 1.1.2.5.3.12 FrmMain.FillLanguageComboboxAndSetFirstLanguage Method

This method is used to Fill the Language combobox And set initial value with The language computer used in

**C#**

```
private void FillLanguageComboboxAndSetFirstLanguage(out string firstLanguage);
```

**Parameters**

Parameters	Description
out string firstLanguage	First Language Info

**Body Source**

```

1: private void FillLanguageComboboxAndSetFirstLanguage(out string firstLanguage)
2: {
3:     cmbLanguage.DropDownStyle = ComboBoxStyle.DropDownList;
4:     cmbLanguage.Items.Clear();
5:     var cultureInfo = CultureInfo.InstalledUICulture;
6:     firstLanguage = string.Empty;
7:     foreach (var item in LanguageList)
8:     {
9:         if (cultureInfo.Name.Contains(item))
10:            firstLanguage = item.ToLower(new CultureInfo("en-US"));
11:         cmbLanguage.Items.Add(item);
12:     }
13:
14:     if (string.IsNullOrEmpty(firstLanguage))
15:         firstLanguage = "tr";
16: }

```

### 1.1.2.5.3.13 FrmMain.FormMembersNameInitialization Method

This method is used to change text of form members by the Resource file which is selected

**C#**

```
private void FormMembersNameInitialization();
```

**Body Source**

```
1: private void FormMembersNameInitialization()
2: {
3:     if (_resource == null) return;
4:     if (ActiveForm != null) ActiveForm.Text = _resource.GetString("AppTitle") ??
string.Empty;
5:     btnAddFile.Text = _resource.GetString("AddFile") ?? string.Empty;
6:     btnAddFolder.Text = _resource.GetString("AddFolder") ?? string.Empty;
7:     btnClearList.Text = _resource.GetString("ClearFileList") ?? string.Empty;
8:     btnCombineITextSharp.Text = _resource.GetString("CombineFilesITextSharp") ??
string.Empty;
9:     btnCombinePdfSharp.Text = _resource.GetString("CombineFilesPdfSharp") ??
string.Empty;
10:    menuItemDelete.Text = _resource.GetString("Delete") ?? string.Empty;
11:    menuItemOrderByPathAscending.Text = _resource.GetString("OrderByPathAscending") ??
string.Empty;
12:    menuItemOrderByPathDescending.Text = _resource.GetString("OrderByPathDescending")
?? string.Empty;
13:    menuItemOrderByNameAscending.Text = _resource.GetString("OrderByNameAscending") ??
string.Empty;
14:    menuItemOrderByNameDescending.Text = _resource.GetString("OrderByNameDescending")
?? string.Empty;
15:    cmbLanguage.Text = _resource.GetString("SelectLanguage") ?? string.Empty;
16: }
```

### 1.1.2.5.3.14 FrmMain.FrmMain\_FormClosed Method

This function is used to terminate application

**C#**

```
private void FrmMain_FormClosed(object sender, FormClosedEventArgs e);
```

**Parameters**

Parameters	Description
object sender	The sender info (For example Main Form)
FormClosedEventArgs e	Event Arguments

**Body Source**

```
1: private void FrmMain_FormClosed(object sender, FormClosedEventArgs e)
2: {
3:     MessageBox.Show(_resource.GetString("ThanksMessage"),
_resource.GetString("AppTitle"), MessageBoxButtons.OK,
4:     MessageBoxIcon.Information);
5:     Application.Exit();
6: }
```

### 1.1.2.5.3.15 FrmMain.FrmMain\_Shown Method

This function is used to assign menu to listbox When form is shown

**C#**

```
private void FrmMain_Shown(object sender, EventArgs e);
```

**Parameters**

Parameters	Description
object sender	The sender info (For example Main Form)
EventArgs e	Event Arguments

**Body Source**

```

1: private void FrmMain_Shown(object sender, EventArgs e)
2: {
3:     lbFiles.ContextMenuStrip = menuStrip;
4:
5:     FillLanguageComboboxAndSetFirstLanguage(out var firstLanguage);
6:
7:     _resource = new ResourceManager("PdfCombiner.Resources.AppResources-" +
firstLanguage,
8:     Assembly.GetExecutingAssembly());
9:     cmbLanguage.SelectedIndex = cmbLanguage.FindStringExact(firstLanguage.ToUpper(new
CultureInfo("en-US")));
10:
11:     FormMembersNameInitialization();
12: }

```

**1.1.2.5.3.16 FrmMain.GenerateAddFileMessage Method**

This method is used to show info message after adding files or folders

**C#**

```
private void GenerateAddFileMessage(int addedFileCount);
```

**Parameters**

Parameters	Description
int addedFileCount	Added File Count

**Body Source**

```

1: private void GenerateAddFileMessage(int addedFileCount)
2: {
3:     MessageBox.Show(addedFileCount + _resource.GetString("FileAddMessage"),
4:     _resource.GetString("AppTitle"), MessageBoxButtons.OK,
MessageBoxIcon.Information);
5:
6:     pbFiles.Value = pbFiles.Minimum;
7:     if (ActiveForm != null) ActiveForm.Text = _resource.GetString("AppTitle");
8: }

```

**1.1.2.5.3.17 FrmMain.GenerateCombineFileMessage Method**

This method is used to set successfull combining file message And set value of Form Caption etc.

**C#**

```
private void GenerateCombineFileMessage(int fileCount, string outputFileName);
```

**Parameters**

Parameters	Description
int fileCount	Total File Count
string outputFileName	Output File Name

**Body Source**

```

1: private void GenerateCombineFileMessage(int fileCount, string outputFileName)
2: {
3:     MessageBox.Show(fileCount + _resource.GetString("CombineFileMessage") +
outputFileName,

```

```

4:         _resource.GetString("AppTitle"), MessageBoxButtons.OK,
MessageBoxIcon.Information);
5:     pbFiles.Value = pbFiles.Minimum;
6:     if (ActiveForm != null) ActiveForm.Text = _resource.GetString("AppTitle");
7: }

```

#### 1.1.2.5.3.18 FrmMain.GenerateCombineWarningMessage Method

This method is used When user just want to combine 1 pdf file

**C#**

```
private static void GenerateCombineWarningMessage();
```

**Body Source**

```

1: private static void GenerateCombineWarningMessage()
2: {
3:     MessageBox.Show(_resource.GetString("CombineWarning"),
_resource.GetString("AppTitle"),
4:     MessageBoxButtons.OK, MessageBoxIcon.Error);
5: }

```

#### 1.1.2.5.3.19 FrmMain.GenerateDeleteItemMessage Method

This method is used after deleting items from listbox

**C#**

```
private static void GenerateDeleteItemMessage(int fileCount);
```

**Parameters**

Parameters	Description
int fileCount	Deleted File Count Information

**Body Source**

```

1: private static void GenerateDeleteItemMessage(int fileCount)
2: {
3:     MessageBox.Show(fileCount + _resource.GetString("DeleteFileMessage"),
_resource.GetString("AppTitle"),
4:     MessageBoxButtons.OK, MessageBoxIcon.Information);
5: }

```

#### 1.1.2.5.3.20 FrmMain.GenerateExceptionMessage Method

This method is used to Show error message when exception occurs and set form title initial value

**C#**

```
private void GenerateExceptionMessage(Exception ex);
```

**Parameters**

Parameters	Description
Exception ex	Exception information

**Body Source**

```

1: private void GenerateExceptionMessage(Exception ex)
2: {
3:     MessageBox.Show(_resource.GetString("CombineErrorMessage") + ex.GetAllMessages(),
4:     _resource.GetString("AppTitle"), MessageBoxButtons.OK, MessageBoxIcon.Error);
5:     pbFiles.Value = pbFiles.Minimum;
6:     if (ActiveForm != null) ActiveForm.Text = _resource.GetString("AppTitle");
7: }

```

#### 1.1.2.5.3.21 FrmMain.GenerateNoFileInListBoxMessage Method

This method is used to generate message When there is no item in listbox



**C#**

```
private static void GenerateNoFileInListBoxMessage();
```

**Body Source**

```
1: private static void GenerateNoFileInListBoxMessage()
2: {
3:     MessageBox.Show(_resource.GetString("NoFile"), _resource.GetString("AppTitle"),
4:     MessageBoxButtons.OK,
5:     MessageBoxIcon.Error);
6: }
```

**1.1.2.5.3.22 FrmMain.GenerateNoSelectedFileInListBoxMessage Method**

This method is used in When there is no selected file in listbox

**C#**

```
private static void GenerateNoSelectedFileInListBoxMessage();
```

**Body Source**

```
1: private static void GenerateNoSelectedFileInListBoxMessage()
2: {
3:     MessageBox.Show(_resource.GetString("NoSelectedFile"),
4:     _resource.GetString("AppTitle"),
5:     MessageBoxButtons.OK, MessageBoxIcon.Error);
6: }
```

**1.1.2.5.3.23 FrmMain.InitializeComponent Method**

Required method for Designer support - do not modify the contents of this method with the code editor.

**C#**

```
private void InitializeComponent();
```

**Body Source**

```
1: private void InitializeComponent()
2: {
3:     this.components = new System.ComponentModel.Container();
4:     System.ComponentModel.ComponentResourceManager resources = new
5: System.ComponentModel.ComponentResourceManager(typeof(FrmMain));
6:     this.btnCombinePdfSharp = new System.Windows.Forms.Button();
7:     this.btnAddFolder = new System.Windows.Forms.Button();
8:     this.btnAddFile = new System.Windows.Forms.Button();
9:     this.btnClearList = new System.Windows.Forms.Button();
10:    this.lbFiles = new System.Windows.Forms.ListBox();
11:    this.lblDetails = new System.Windows.Forms.Label();
12:    this.pbFiles = new System.Windows.Forms.ProgressBar();
13:    this.btnCombineITextSharp = new System.Windows.Forms.Button();
14:    this.menuStrip = new System.Windows.Forms.ContextMenuStrip(this.components);
15:    this.menuItemDelete = new System.Windows.Forms.ToolStripMenuItem();
16:    this.menuItemOrderByPathAscending = new System.Windows.Forms.ToolStripMenuItem();
17:    this.menuItemOrderByPathDescending = new System.Windows.Forms.ToolStripMenuItem();
18:    this.menuItemOrderByNameAscending = new System.Windows.Forms.ToolStripMenuItem();
19:    this.menuItemOrderByNameDescending = new System.Windows.Forms.ToolStripMenuItem();
20:    this.cmbLanguage = new System.Windows.Forms.ComboBox();
21:    this.menuStrip.SuspendLayout();
22:    this.SuspendLayout();
23:    //
24:    // btnCombinePdfSharp
25:    //
26:    this.btnCombinePdfSharp.Image = ((System.Drawing.Image)
27: (resources.GetObject("btnCombinePdfSharp.Image")));
28:    this.btnCombinePdfSharp.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
29:    this.btnCombinePdfSharp.Location = new System.Drawing.Point(442, 5);
30:    this.btnCombinePdfSharp.Name = "btnCombinePdfSharp";
31:    this.btnCombinePdfSharp.Size = new System.Drawing.Size(214, 37);
```

```

30:         this.btnCombinePdfSharp.TabIndex = 0;
31:         this.btnCombinePdfSharp.Text = "Combine Files (PdfSharp)";
32:         this.btnCombinePdfSharp.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
33:         this.btnCombinePdfSharp.UseVisualStyleBackColor = true;
34:         this.btnCombinePdfSharp.Click += new
System.EventHandler(this.btnCombinePdfSharp_Click);
35:         //
36:         // btnAddFolder
37:         //
38:         this.btnAddFolder.Image = ((System.Drawing.Image)
(resources.GetObject("btnAddFolder.Image")));
39:         this.btnAddFolder.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
40:         this.btnAddFolder.Location = new System.Drawing.Point(197, 5);
41:         this.btnAddFolder.Name = "btnAddFolder";
42:         this.btnAddFolder.Size = new System.Drawing.Size(171, 37);
43:         this.btnAddFolder.TabIndex = 2;
44:         this.btnAddFolder.Text = "Add Folder";
45:         this.btnAddFolder.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
46:         this.btnAddFolder.UseVisualStyleBackColor = true;
47:         this.btnAddFolder.Click += new System.EventHandler(this.btnAddFolder_Click);
48:         //
49:         // btnAddFile
50:         //
51:         this.btnAddFile.Image = ((System.Drawing.Image)
(resources.GetObject("btnAddFile.Image")));
52:         this.btnAddFile.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
53:         this.btnAddFile.Location = new System.Drawing.Point(12, 5);
54:         this.btnAddFile.Name = "btnAddFile";
55:         this.btnAddFile.Size = new System.Drawing.Size(179, 37);
56:         this.btnAddFile.TabIndex = 3;
57:         this.btnAddFile.Text = "Add Files";
58:         this.btnAddFile.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
59:         this.btnAddFile.UseVisualStyleBackColor = true;
60:         this.btnAddFile.Click += new System.EventHandler(this.btnAddFile_Click);
61:         //
62:         // btnClearList
63:         //
64:         this.btnClearList.Image = ((System.Drawing.Image)
(resources.GetObject("btnClearList.Image")));
65:         this.btnClearList.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
66:         this.btnClearList.Location = new System.Drawing.Point(12, 45);
67:         this.btnClearList.Name = "btnClearList";
68:         this.btnClearList.Size = new System.Drawing.Size(179, 37);
69:         this.btnClearList.TabIndex = 4;
70:         this.btnClearList.Text = "Clear File List";
71:         this.btnClearList.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
72:         this.btnClearList.UseVisualStyleBackColor = true;
73:         this.btnClearList.Click += new System.EventHandler(this.btnClearList_Click);
74:         //
75:         // lbFiles
76:         //
77:         this.lbFiles.AllowDrop = true;
78:         this.lbFiles.FormattingEnabled = true;
79:         this.lbFiles.Location = new System.Drawing.Point(12, 88);
80:         this.lbFiles.Name = "lbFiles";
81:         this.lbFiles.SelectionMode = System.Windows.Forms.SelectionMode.MultiExtended;
82:         this.lbFiles.Size = new System.Drawing.Size(644, 186);
83:         this.lbFiles.TabIndex = 5;
84:         this.lbFiles.DragDrop += new
System.Windows.Forms.DragEventHandler(this.lbFiles_DragDrop);
85:         this.lbFiles.DragOver += new
System.Windows.Forms.DragEventHandler(this.lbFiles_DragOver);
86:         this.lbFiles.KeyDown += new
System.Windows.Forms.KeyEventHandler(this.lbFiles_KeyDown);
87:         this.lbFiles.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.lbFiles_MouseDown);
88:         //
89:         // lblDetails
90:         //

```

```

91:         this.lblDetails.AutoSize = true;
92:         this.lblDetails.Location = new System.Drawing.Point(12, 65);
93:         this.lblDetails.Name = "lblDetails";
94:         this.lblDetails.Size = new System.Drawing.Size(0, 13);
95:         this.lblDetails.TabIndex = 6;
96:         //
97:         // pbFiles
98:         //
99:         this.pbFiles.Location = new System.Drawing.Point(197, 45);
100:        this.pbFiles.Name = "pbFiles";
101:        this.pbFiles.Size = new System.Drawing.Size(239, 37);
102:        this.pbFiles.TabIndex = 7;
103:        //
104:        // btnCombineITextSharp
105:        //
106:        this.btnCombineITextSharp.Image = ((System.Drawing.Image)
(resources.GetObject("btnCombineITextSharp.Image")));
107:        this.btnCombineITextSharp.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
108:        this.btnCombineITextSharp.Location = new System.Drawing.Point(442, 45);
109:        this.btnCombineITextSharp.Name = "btnCombineITextSharp";
110:        this.btnCombineITextSharp.Size = new System.Drawing.Size(214, 37);
111:        this.btnCombineITextSharp.TabIndex = 9;
112:        this.btnCombineITextSharp.Text = "Combine Files (iTextSharp)";
113:        this.btnCombineITextSharp.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
114:        this.btnCombineITextSharp.UseVisualStyleBackColor = true;
115:        this.btnCombineITextSharp.Click += new
System.EventHandler(this.btnCombineITextSharp_Click);
116:        //
117:        // menuStrip
118:        //
119:        this.menuStrip.Items.AddRange(new System.Windows.Forms.ToolStripItem[]
{this.menuItemDelete, this.menuItemOrderByPathAscending,
this.menuItemOrderByPathDescending, this.menuItemOrderByNameAscending,
this.menuItemOrderByNameDescending});
120:        this.menuStrip.Name = "menuStrip";
121:        this.menuStrip.Size = new System.Drawing.Size(229, 114);
122:        //
123:        // menuItemDelete
124:        //
125:        this.menuItemDelete.Image = ((System.Drawing.Image)
(resources.GetObject("menuItemDelete.Image")));
126:        this.menuItemDelete.Name = "menuItemDelete";
127:        this.menuItemDelete.Size = new System.Drawing.Size(228, 22);
128:        this.menuItemDelete.Text = "Delete";
129:        this.menuItemDelete.Click += new System.EventHandler(this.menuItemDelete_Click);
130:        //
131:        // menuItemOrderByPathAscending
132:        //
133:        this.menuItemOrderByPathAscending.Image = ((System.Drawing.Image)
(resources.GetObject("menuItemOrderByPathAscending.Image")));
134:        this.menuItemOrderByPathAscending.Name = "menuItemOrderByPathAscending";
135:        this.menuItemOrderByPathAscending.Size = new System.Drawing.Size(228, 22);
136:        this.menuItemOrderByPathAscending.Text = "Order By Path (Ascending)";
137:        this.menuItemOrderByPathAscending.Click += new
System.EventHandler(this.menuItemOrderByPathAscending_Click);
138:        //
139:        // menuItemOrderByPathDescending
140:        //
141:        this.menuItemOrderByPathDescending.Image = ((System.Drawing.Image)
(resources.GetObject("menuItemOrderByPathDescending.Image")));
142:        this.menuItemOrderByPathDescending.Name = "menuItemOrderByPathDescending";
143:        this.menuItemOrderByPathDescending.Size = new System.Drawing.Size(228, 22);
144:        this.menuItemOrderByPathDescending.Text = "Order By Path (Descending)";
145:        this.menuItemOrderByPathDescending.Click += new
System.EventHandler(this.menuItemOrderByPathDescending_Click);
146:        //
147:        // menuItemOrderByNameAscending
148:        //
149:        this.menuItemOrderByNameAscending.Image = ((System.Drawing.Image)

```

```

(resources.GetObject("menuItemOrderByNameAscending.Image"));
150:     this.menuItemOrderByNameAscending.Name = "menuItemOrderByNameAscending";
151:     this.menuItemOrderByNameAscending.Size = new System.Drawing.Size(228, 22);
152:     this.menuItemOrderByNameAscending.Text = "Order By Name (Ascending)";
153:     this.menuItemOrderByNameAscending.Click += new
System.EventHandler(this.menuItemOrderByNameAscending_Click);
154:     //
155:     // menuItemOrderByNameDescending
156:     //
157:     this.menuItemOrderByNameDescending.Image = ((System.Drawing.Image)
(resources.GetObject("menuItemOrderByNameDescending.Image")));
158:     this.menuItemOrderByNameDescending.Name = "menuItemOrderByNameDescending";
159:     this.menuItemOrderByNameDescending.Size = new System.Drawing.Size(228, 22);
160:     this.menuItemOrderByNameDescending.Text = "Order By Name (Descending)";
161:     this.menuItemOrderByNameDescending.Click += new
System.EventHandler(this.menuItemOrderByNameDescending_Click);
162:     //
163:     // cmbLanguage
164:     //
165:     this.cmbLanguage.Anchor = System.Windows.Forms.AnchorStyles.None;
166:     this.cmbLanguage.Font = new System.Drawing.Font("Microsoft Sans Serif", 15F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte) (162)));
167:     this.cmbLanguage.FormattingEnabled = true;
168:     this.cmbLanguage.Items.AddRange(new object[] { "TR", "EN", "DE", "FR", "RU", "ES" });
169:     this.cmbLanguage.Location = new System.Drawing.Point(374, 6);
170:     this.cmbLanguage.Name = "cmbLanguage";
171:     this.cmbLanguage.Size = new System.Drawing.Size(62, 33);
172:     this.cmbLanguage.TabIndex = 10;
173:     this.cmbLanguage.Text = "Select Language";
174:     this.cmbLanguage.SelectedIndexChanged += new
System.EventHandler(this.cmbLanguage_SelectedIndexChanged);
175:     //
176:     // FrmMain
177:     //
178:     this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
179:     this.AutoScaleMode = System.Windows.Forms.AutoScaleModeMode.Font;
180:     this.ClientSize = new System.Drawing.Size(668, 287);
181:     this.Controls.Add(this.cmbLanguage);
182:     this.Controls.Add(this.btnCombineITextSharp);
183:     this.Controls.Add(this.pbFiles);
184:     this.Controls.Add(this.lblDetails);
185:     this.Controls.Add(this.lbFiles);
186:     this.Controls.Add(this.btnClearList);
187:     this.Controls.Add(this.btnAddFile);
188:     this.Controls.Add(this.btnAddFolder);
189:     this.Controls.Add(this.btnCombinePdfSharp);
190:     this.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte) (162)));
191:     this.Name = "FrmMain";
192:     this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
193:     this.Text = "PDF Combiner";
194:     this.FormClosed += new
System.Windows.Forms.FormClosedEventHandler(this.FrmMain_FormClosed);
195:     this.Shown += new System.EventHandler(this.FrmMain_Shown);
196:     this.menuStrip.ResumeLayout(false);
197:     this.ResumeLayout(false);
198:     this.PerformLayout();
199: }

```

#### 1.1.2.5.3.24 FrmMain.InitializeFileDialog Method

This function is used to set options of file dialog like Filter, start path etc.

C#

```
private static void InitializeFileDialog(OpenFileDialog dialogAddFile);
```

**Parameters**

Parameters	Description
OpenFileDialog dialogAddFile	File Dialog

**Body Source**

```
1: private static void InitializeFileDialog(OpenFileDialog dialogAddFile)
2: {
3:     dialogAddFile.Multiselect = true;
4:     dialogAddFile.Filter = _resource.GetString("PdfFiles") + @" (*.pdf)|*.pdf";
5:     dialogAddFile.InitialDirectory = Application.StartupPath;
6:     dialogAddFile.Title = _resource.GetString("SelectPdfFile");
7:     dialogAddFile.DefaultExt = "PDF";
8: }
```

**1.1.2.5.3.25 FrmMain.lbFiles\_DragDrop Method**

This is used to drag and drop an item in listbox Which is used to order books and combine them with that order

**C#**

```
private void lbFiles_DragDrop(object sender, DragEventArgs e);
```

**Parameters**

Parameters	Description
object sender	The sender info (For example Main Form)
DragEventArgs e	Event Arguments

**Description**

m>

**Body Source**

```
1: private void lbFiles_DragDrop(object sender, DragEventArgs e)
2: {
3:     var point = lbFiles.PointToClient(new Point(e.X, e.Y));
4:     var index = lbFiles.IndexFromPoint(point);
5:     if (index < 0) index = lbFiles.Items.Count - 1;
6:     // type of string because file names stored in string in listbox
7:     var data = e.Data.GetData(typeof(string));
8:     lbFiles.Items.Remove(data);
9:     lbFiles.Items.Insert(index, data);
10: }
```

**1.1.2.5.3.26 FrmMain.lbFiles\_DragOver Method**

This is used to give effect while dragging and dropping an item in listbox

**C#**

```
private void lbFiles_DragOver(object sender, DragEventArgs e);
```

**Parameters**

Parameters	Description
object sender	The sender info (For example Main Form)
DragEventArgs e	Event Arguments

**Body Source**

```
1: private void lbFiles_DragOver(object sender, DragEventArgs e)
2: {
3:     e.Effect = DragDropEffects.Move;
4: }
```

### 1.1.2.5.3.27 FrmMain.lbFiles\_KeyDown Method

This function is used to operate some process in files in listbox which are chosen

**C#**

```
private void lbFiles_KeyDown(object sender, KeyEventArgs e);
```

**Parameters**

Parameters	Description
object sender	The sender info (For example Main Form)
KeyEventArgs e	Event Arguments

**Body Source**

```
1: private void lbFiles_KeyDown(object sender, KeyEventArgs e)
2: {
3:     // this is used to delete selected files in listbox when you press Delete key
4:     if (e.KeyValue == Keys.Delete.GetHashCode())
5:         DeleteFilesFromListBox();
6:     // This is used to select all items in listbox when you press Ctrl + A key
7:     combination
8:     if (!e.Control || e.KeyCode != Keys.A) return;
9:     for (var i = 0; i < lbFiles.Items.Count; i++)
10:    {
11:        lbFiles.SetSelected(i, true);
12:    }
```

### 1.1.2.5.3.28 FrmMain.lbFiles\_MouseDown Method

This is used to control mouse actions and if that is not right click Start to drag and drop item in listbox

**C#**

```
private void lbFiles_MouseDown(object sender, MouseEventArgs e);
```

**Parameters**

Parameters	Description
object sender	The sender info (For example Main Form)
MouseEventArgs e	Event Arguments

**Body Source**

```
1: private void lbFiles_MouseDown(object sender, MouseEventArgs e)
2: {
3:     if (e.Button.Equals(MouseButtons.Right)) return;
4:     if (lbFiles.SelectedItem == null) return;
5:     lbFiles.DoDragDrop(lbFiles.SelectedItem, DragDropEffects.Move);
6: }
```

### 1.1.2.5.3.29 FrmMain.menuItemDelete\_Click Method

This is used to delete selected items in listbox When menu item Delete is clicked and opened dialog

**C#**

```
private void menuItemDelete_Click(object sender, EventArgs e);
```

**Parameters**

Parameters	Description
object sender	The sender info (For example Main Form)
EventArgs e	Event Arguments



**Body Source**

```

1: private void menuItemDelete_Click(object sender, EventArgs e)
2: {
3:     if (lbFiles.Items.Count > 0)
4:     {
5:         if (lbFiles.SelectedItems.Count > 0)
6:         {
7:             ShowDeleteDialog(out var deleteDialog);
8:             if (deleteDialog == DialogResult.Yes)
9:                 DeleteFilesFromListBox();
10:        }
11:        else
12:            GenerateNoSelectedFileInListBoxMessage();
13:    }
14:    else
15:        GenerateNoFileInListBoxMessage();
16: }

```

**1.1.2.5.3.30 FrmMain.menuItemOrderByNameAscending\_Click Method**

This method is used to order elements in listbox ascending by file name

**C#**

```
private void menuItemOrderByNameAscending_Click(object sender, EventArgs e);
```

**Parameters**

Parameters	Description
object sender	The sender info (For example Main Form)
EventArgs e	Event Arguments

**Body Source**

```

1: private void menuItemOrderByNameAscending_Click(object sender, EventArgs e)
2: {
3:     if (lbFiles.Items.Count > 0)
4:         lbFiles = SortItemsByName(lbFiles, OrderType.Ascending);
5:     else
6:         GenerateNoFileInListBoxMessage();
7: }

```

**1.1.2.5.3.31 FrmMain.menuItemOrderByNameDescending\_Click Method**

This method is used to order elements in listbox descending by file name

**C#**

```
private void menuItemOrderByNameDescending_Click(object sender, EventArgs e);
```

**Parameters**

Parameters	Description
object sender	The sender info (For example Main Form)
EventArgs e	Event Arguments

**Body Source**

```

1: private void menuItemOrderByNameDescending_Click(object sender, EventArgs e)
2: {
3:     if (lbFiles.Items.Count > 0)
4:         lbFiles = SortItemsByName(lbFiles, OrderType.Descending);
5:     else
6:         GenerateNoFileInListBoxMessage();
7: }

```

### 1.1.2.5.3.32 FrmMain.menuItemOrderByPathAscending\_Click Method

This method is used to order elements in listbox ascending by full path

**C#**

```
private void menuItemOrderByPathAscending_Click(object sender, EventArgs e);
```

**Parameters**

Parameters	Description
object sender	The sender info (For example Main Form)
EventArgs e	Event Arguments

**Body Source**

```
1: private void menuItemOrderByPathAscending_Click(object sender, EventArgs e)
2: {
3:     if (lbFiles.Items.Count > 0)
4:         lbFiles = SortItemsByPath(lbFiles, OrderType.Ascending);
5:     else
6:         GenerateNoFileInListBoxMessage();
7: }
```

### 1.1.2.5.3.33 FrmMain.menuItemOrderByPathDescending\_Click Method

This method is used to order elements in listbox descending by full path

**C#**

```
private void menuItemOrderByPathDescending_Click(object sender, EventArgs e);
```

**Parameters**

Parameters	Description
object sender	The sender info (For example Main Form)
EventArgs e	Event Arguments

**Body Source**

```
1: private void menuItemOrderByPathDescending_Click(object sender, EventArgs e)
2: {
3:     if (lbFiles.Items.Count > 0)
4:         lbFiles = SortItemsByPath(lbFiles, OrderType.Descending);
5:     else
6:         GenerateNoFileInListBoxMessage();
7: }
```

### 1.1.2.5.3.34 FrmMain.SaveOutputPdfFile Method

This method is used to Save Combined PDF File

**C#**

```
private static void SaveOutputPdfFile(PdfDocument outputFile, string outputFileName);
```

**Parameters**

Parameters	Description
PdfDocument outputFile	Combined PDF File
string outputFileName	Combined PDF File Path

**Body Source**

```
1: private static void SaveOutputPdfFile(PdfDocument outputFile, string outputFileName)
2: {
3:     outputFile.Save(outputFileName);
4:     outputFile.Close();
}
```



```
5: }
```

### 1.1.2.5.3.35 FrmMain.SetCombinationRatio Method

This method is used to set combination degree while combining PDF files And show this ratio on Form Caption

**C#**

```
private void SetCombinationRatio(int combinedFiles, int fileCount);
```

**Parameters**

Parameters	Description
int combinedFiles	Combined File Count
int fileCount	Total File Count

**Body Source**

```
1: private void SetCombinationRatio(int combinedFiles, int fileCount)
2: {
3:     pbFiles.Value = combinedFiles * 100 / fileCount;
4:     if (pbFiles.Value > pbFiles.Maximum)
5:         pbFiles.Value = pbFiles.Maximum;
6:     if (ActiveForm != null) ActiveForm.Text = @"%" + pbFiles.Value;
7: }
```

### 1.1.2.5.3.36 FrmMain.SetInitialValuesForCombiningFiles Method

This method is used to set initial values of some properties Which are used in combining PDF files In both PdfSharp and ITextSharp

**C#**

```
private void SetInitialValuesForCombiningFiles(FolderBrowserDialog dialogExport, out string
outputFileName, out int fileCount, out int combinedFiles);
```

**Parameters**

Parameters	Description
FolderBrowserDialog dialogExport	Export Dialog Info
out string outputFileName	Output File Name
out int fileCount	Total File Count
out int combinedFiles	Combined Files Count

**Body Source**

```
1: private void SetInitialValuesForCombiningFiles(FolderBrowserDialog dialogExport, out
string outputFileName,
2:     out int fileCount, out int combinedFiles)
3: {
4:     outputFileName = dialogExport.SelectedPath + "/" + Guid.NewGuid() + ".pdf";
5:     fileCount = lbFiles.Items.Count;
6:     combinedFiles = 0;
7: }
```

### 1.1.2.5.3.37 FrmMain.SetOutputFilePropertiesPdfSharp Method

This method is used to set combined PDF file Properties (see page 1) like Compression etc.

**C#**

```
private static void SetOutputFilePropertiesPdfSharp(PdfDocument outputFile);
```

**Parameters**

Parameters	Description
PdfDocument outputFile	Combined PDF File

**Body Source**

```

1: private static void SetOutputFilePropertiesPdfSharp(PdfDocument outputFile)
2: {
3:     outputFile.Options.CompressContentStreams = true;
4:     outputFile.Options.EnableCcittCompressionForBilevelImages = true;
5:     outputFile.Options.FlateEncodeMode = PdfFlateEncodeMode.BestCompression;
6: }

```

**1.1.2.5.38 FrmMain.SetProgramLanguage Method**

This method is used to Set Program (see page 33) Language By the selection from Language Combobox Finally, with the selected language Form elements renamed with the selected language

**C#**

```
private void SetProgramLanguage();
```

**Body Source**

```

1: private void SetProgramLanguage()
2: {
3:     try
4:     {
5:         _resource = new ResourceManager(
6:             "PdfCombiner.Resources.AppResources-" +
7:             cmbLanguage.SelectedItem.ToString().ToLower(new CultureInfo("en-US")),
8:             Assembly.GetExecutingAssembly());
9:     }
10:    catch (Exception)
11:    {
12:        _resource = new ResourceManager("PdfCombiner.Resources.AppResources-en",
13:            Assembly.GetExecutingAssembly());
14:    }
15:    finally
16:    {
17:        FormMembersNameInitialization();
18:    }
19: }

```

**1.1.2.5.39 FrmMain.ShowDeleteDialog Method**

This method is used to Show delete file dialog

**C#**

```
private void ShowDeleteDialog(out DialogResult deleteDialog);
```

**Parameters**

Parameters	Description
out DialogResult deleteDialog	Delete File Dialog

**Body Source**

```

1: private void ShowDeleteDialog(out DialogResult deleteDialog)
2: {
3:     deleteDialog = MessageBox.Show(
4:         _resource.GetString("DeleteWarning1") + lbFiles.SelectedItems.Count +
5:         _resource.GetString("DeleteWarning2"), _resource.GetString("AppTitle"),
6:         MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
7: }

```

**1.1.2.5.3.40 FrmMain.SortItemsByName Method**

This method is used to order items by file name in listbox And return them with the given order as parametre

**C#**

```
private static ListBox SortItemsByName(ListBox listBox, OrderType orderType);
```

**Parameters**

Parameters	Description
ListBox listBox	ListBox Info
OrderType orderType	Order Type is Ascending Or Not

**Body Source**

```

1: private static ListBox SortItemsByName(ListBox listBox, OrderType orderType)
2: {
3:     var items = listBox.Items.OfType<object>().ToList();
4:     const char c = '\u005c';
5:     var fileList = (from item in items select item.ToString() into fullPath let list =
fullPath.Split(c).ToList() where list.Count > 0 select new FileInfo {FilePath = fullPath,
FileName = list[list.Count - 1]}).ToList();
6:
7:     fileList = orderType == OrderType.Ascending
8:         ? fileList.OrderBy(j => j.FileName).ThenBy(j => j.FilePath).ToList()
9:         : fileList.OrderByDescending(j => j.FileName).ThenByDescending(j =>
j.FilePath).ToList();
10:
11:     listBox.Items.Clear();
12:     foreach (var item in fileList)
13:         listBox.Items.Add(item.FilePath);
14:
15:     return listBox;
16: }

```

**1.1.2.5.3.41 FrmMain.SortItemsByPath Method**

This method is used to order items by full path in listbox And return them with the given order as parametre

**C#**

```
private static ListBox SortItemsByPath(ListBox listBox, OrderType orderType);
```

**Parameters**

Parameters	Description
ListBox listBox	ListBox Info
OrderType orderType	Order Type is Ascending Or Not

**Body Source**

```

1: private static ListBox SortItemsByPath(ListBox listBox, OrderType orderType)
2: {
3:     var items = listBox.Items.OfType<object>().ToList();
4:     listBox.Items.Clear();
5:     listBox.Items.AddRange(orderType == OrderType.Ascending
6:         ? items.OrderBy(i => i).ToArray()
7:         : items.OrderByDescending(i => i).ToArray());
8:     return listBox;
9: }

```

**1.1.2.6 Program Class****Class Hierarchy**

```
PdfCombiner.Program
```

**C#**

```
internal static class Program;
```


**File**

Program.cs (see page 54)

**Description**

This is class PdfCombiner.Program.

**Program Methods**

	Name	Description
	Main ( <a href="#">see page 34</a> )	The main entry point for the application.

**1.1.2.6.1 Program Methods****1.1.2.6.1.1 Program.Main Method**

The main entry point for the application.

**C#**

```
[STAThread]
private static void Main();
```

**Body Source**

```
1: [STAThread]
2: private static void Main()
3: {
4:     Application.EnableVisualStyles();
5:     Application.SetCompatibleTextRenderingDefault(false);
6:     Application.Run(new FrmMain());
7: }
```

**1.2 Files**

The following table lists files in this documentation.

**Files**

Name	Description
AssemblyInfo.cs ( <a href="#">see page 35</a> )	This is file AssemblyInfo.cs.
BaseException.cs ( <a href="#">see page 35</a> )	This is file BaseException.cs.
Enums.cs ( <a href="#">see page 36</a> )	This is file Enums.cs.
ExceptionExtensions.cs ( <a href="#">see page 36</a> )	This is file ExceptionExtensions.cs.
FileInfo.cs ( <a href="#">see page 37</a> )	This is file FileInfo.cs.
FrmMain.cs ( <a href="#">see page 37</a> )	This is file FrmMain.cs.
FrmMain.Designer.cs ( <a href="#">see page 49</a> )	This is file FrmMain.Designer.cs.
PdfCombiner.csproj ( <a href="#">see page 54</a> )	This is file PdfCombiner.csproj.
PdfCombiner.sln ( <a href="#">see page 54</a> )	This is file PdfCombiner.sln.
Program.cs ( <a href="#">see page 54</a> )	This is file Program.cs.
Resources.Designer.cs ( <a href="#">see page 55</a> )	This code was generated by a tool. Runtime Version:4.0.30319.42000 Changes to this file may cause incorrect behavior and will be lost if the code is regenerated.
Settings.Designer.cs ( <a href="#">see page 56</a> )	This code was generated by a tool. Runtime Version:4.0.30319.42000 Changes to this file may cause incorrect behavior and will be lost if the code is regenerated.

## 1.2.1 AssemblyInfo.cs

This is file AssemblyInfo.cs.

### Body Source

```
1: ?using System.Reflection;
2: using System.Runtime.InteropServices;
3:
4: // General Information about an assembly is controlled through the following
5: // set of attributes. Change these attribute values to modify the information
6: // associated with an assembly.
7: [assembly: AssemblyTitle("PdfCombiner")]
8: [assembly: AssemblyDescription("")]
9: [assembly: AssemblyConfiguration("")]
10: [assembly: AssemblyCompany("")]
11: [assembly: AssemblyProduct("PdfCombiner")]
12: [assembly: AssemblyCopyright("Copyright © 2022")]
13: [assembly: AssemblyTrademark("")]
14: [assembly: AssemblyCulture("")]
15:
16: // Setting ComVisible to false makes the types in this assembly not visible
17: // to COM components. If you need to access a type in this assembly from
18: // COM, set the ComVisible attribute to true on that type.
19: [assembly: ComVisible(false)]
20:
21: // The following GUID is for the ID of the typelib if this project is exposed to COM
22: [assembly: Guid("48e34ee8-dfaa-48db-acfd-70573cee9c00")]
23:
24: // Version information for an assembly consists of the following four values:
25: //
26: //      Major Version
27: //      Minor Version
28: //      Build Number
29: //      Revision
30: //
31: // You can specify all the values or you can default the Build and Revision Numbers
32: // by using the '*' as shown below:
33: // [assembly: AssemblyVersion("1.0.*")]
34: [assembly: AssemblyVersion("1.0.0.0")]
35: [assembly: AssemblyFileVersion("1.0.0.0")]
```

## 1.2.2 BaseException.cs

This is file BaseException.cs.

### Body Source

```
1: ?using System;
2:
3: namespace PdfCombiner
4: {
5:     /// <summary>
6:     /// This function is used to take exceptions which can be thrown in
7:     /// PDF Combiner App
8:     /// </summary>
9:     [Serializable]
10:    public abstract class BaseException : Exception
11:    {
12:        protected BaseException()
13:        {
```

```
14:         }
15:
16:         protected BaseException(string message) : base(message)
17:         {
18:         }
19:
20:         protected BaseException(string message, Exception innerException) :
base(message, innerException)
21:         {
22:         }
23:     }
24: }
```

#### Namespaces

Name	Description
PdfCombiner (see page 1)	This is namespace PdfCombiner.

## 1.2.3 Enums.cs

This is file Enums.cs.

#### Body Source

```
1: ?using System;
2:
3: namespace PdfCombiner
4: {
5:     /// <summary>
6:     /// This class contains enums for detailed information
7:     /// In usage
8:     /// </summary>
9:     public static class Enums
10:    {
11:        [Flags]
12:        public enum OrderType
13:        {
14:            Ascending,
15:            Descending
16:        }
17:    }
18: }
```

#### Namespaces

Name	Description
PdfCombiner (see page 1)	This is namespace PdfCombiner.

## 1.2.4 ExceptionExtensions.cs

This is file ExceptionExtensions.cs.

#### Body Source

```
1: ?using System;
2: using System.Text;
3:
4: namespace PdfCombiner
5: {
6:     public static class ExceptionExtensions
7:     {
```

```

8:      /// <summary>
9:      /// This method is used to take all messages with stack trace info in exception
10:     /// </summary>
11:     /// <param name="exception">Exception</param>
12:     /// <returns>Detailed Error Message</returns>
13:     public static string GetAllMessages(this Exception exception)
14:     {
15:         var sb = new StringBuilder();
16:         sb.Append(exception.Message);
17:         if (exception.InnerException != null)
18:             sb.Append($" {exception.InnerException.GetAllMessages()}");
19:         if (!string.IsNullOrEmpty(exception.StackTrace))
20:             sb.Append($" {exception.StackTrace}");
21:         if (!string.IsNullOrEmpty(exception.Source))
22:             sb.Append($" {exception.Source}");
23:         return sb.ToString();
24:     }
25: }
26: }

```

#### Namespaces

Name	Description
PdfCombiner (see page 1)	This is namespace PdfCombiner.

## 1.2.5 FileInfo.cs

This is file FileInfo.cs.

#### Body Source

```

1: ?namespace PdfCombiner
2: {
3:     /// <summary>
4:     /// This class is used to have File's Full Path and Name
5:     /// </summary>
6:     public class FileInfo
7:     {
8:         /// <summary>
9:         /// File Path Property
10:        /// </summary>
11:        public string FilePath { get; set; }
12:        /// <summary>
13:        /// File Name Property
14:        /// </summary>
15:        public string FileName { get; set; }
16:    }
17: }

```

#### Namespaces

Name	Description
PdfCombiner (see page 1)	This is namespace PdfCombiner.

## 1.2.6 FrmMain.cs

This is file FrmMain.cs.

#### Body Source

```

1: ?using iTextSharp.text;

```

```

2: using PdfSharp.Pdf;
3: using PdfSharp.Pdf.IO;
4: using System;
5: using System.Collections.Generic;
6: using System.Drawing;
7: using System.Globalization;
8: using System.IO;
9: using System.Linq;
10: using System.Reflection;
11: using System.Resources;
12: using System.Windows.Forms;
13: using iTextSharp.text.pdf;
14: using PdfDocument = PdfSharp.Pdf.PdfDocument;
15: using PdfReader = PdfSharp.Pdf.IO.PdfReader;
16: using static PdfCombiner.Enums;
17:
18: namespace PdfCombiner
19: {
20:     public partial class FrmMain : Form
21:     {
22:         private static ResourceManager _resource;
23:
24:         // Culture lists whose resources have been added to project
25:         private static readonly List<string> LanguageList = new List<string>
26:             { "EN", "TR", "DE", "FR", "RU", "ES", "IT", "ZH", "AR", "NL", "PT", "IN",
27: "ID", "JP", "BG" };
28:
29:         public FrmMain()
30:         {
31:             InitializeComponent();
32:
33:             #region Form Init, Language Settings and Finalize Methods
34:
35:             /// <summary>
36:             /// This function is used to terminate application
37:             /// </summary>
38:             /// <param name="sender">The sender info (For example Main Form)</param>
39:             /// <param name="e">Event Arguments</param>
40:             private void FrmMain_FormClosed(object sender, FormClosedEventArgs e)
41:             {
42:                 MessageBox.Show(_resource.GetString("ThanksMessage"),
43: _resource.GetString("AppTitle"), MessageBoxButtons.OK,
44:                 MessageBoxIcon.Information);
45:                 Application.Exit();
46:
47:                 /// <summary>
48:                 /// This function is used to assign menu to listbox
49:                 /// When form is shown
50:                 /// </summary>
51:                 /// <param name="sender">The sender info (For example Main Form)</param>
52:                 /// <param name="e">Event Arguments</param>
53:                 private void FrmMain_Shown(object sender, EventArgs e)
54:                 {
55:                     lbFiles.ContextMenuStrip = menuStrip;
56:
57:                     FillLanguageComboboxAndSetFirstLanguage(out var firstLanguage);
58:
59:                     _resource = new ResourceManager("PdfCombiner.Resources.AppResources-" +
60: firstLanguage,
61:                     Assembly.GetExecutingAssembly());
62:                     cmbLanguage.SelectedIndex =
63:                     cmbLanguage.FindStringExact(firstLanguage.ToUpper(new CultureInfo("en-US")));
64:
65:                     FormMembersNameInitialization();
66:                 }
67:
68:                 /// <summary>

```



```

67:      /// This method is used to Fill the Language combobox
68:      /// And set initial value with
69:      /// The language computer used in
70:      /// </summary>
71:      /// <param name="firstLanguage">First Language Info</param>
72:      private void FillLanguageComboboxAndSetFirstLanguage(out string firstLanguage)
73:      {
74:          cmbLanguage.DropDownStyle = ComboBoxStyle.DropDownList;
75:          cmbLanguage.Items.Clear();
76:          var cultureInfo = CultureInfo.InstalledUICulture;
77:          firstLanguage = string.Empty;
78:          foreach (var item in LanguageList)
79:          {
80:              if (cultureInfo.Name.Contains(item))
81:                  firstLanguage = item.ToLower(new CultureInfo("en-US"));
82:              cmbLanguage.Items.Add(item);
83:          }
84:
85:          if (string.IsNullOrEmpty(firstLanguage))
86:              firstLanguage = "tr";
87:      }
88:
89:      /// <summary>
90:      /// This function is used to set the _resource file
91:      /// And with this, you can see form elements, info messages etc. with the
selected
92:      /// _resource file contents. That is used to multi language support
93:      /// </summary>
94:      /// <param name="sender">The sender info (For example Main Form)</param>
95:      /// <param name="e">Event Arguments</param>
96:      private void cmbLanguage_SelectedIndexChanged(object sender, EventArgs e)
97:      {
98:          if (cmbLanguage.SelectedIndex == -1) return;
99:          SetProgramLanguage();
100:      }
101:
102:      /// <summary>
103:      /// This method is used to Set Program Language
104:      /// By the selection from Language Combobox
105:      /// Finally, with the selected language
106:      /// Form elements renamed with the selected language
107:      /// </summary>
108:      private void SetProgramLanguage()
109:      {
110:          try
111:          {
112:              _resource = new ResourceManager(
113:                  "PdfCombiner.Resources.AppResources-" +
114:                  cmbLanguage.SelectedItem.ToString().ToLower(new
CultureInfo("en-US")),
115:                  Assembly.GetExecutingAssembly());
116:          }
117:          catch (Exception)
118:          {
119:              _resource = new
ResourceManager("PdfCombiner.Resources.AppResources-en",
120:                  Assembly.GetExecutingAssembly());
121:          }
122:          finally
123:          {
124:              FormMembersNameInitialization();
125:          }
126:      }
127:
128:      /// <summary>
129:      /// This method is used to change text of form members by the
130:      /// Resource file which is selected
131:      /// </summary>
132:      private void FormMembersNameInitialization()

```

```

133:         {
134:             if (_resource == null) return;
135:             if (ActiveForm != null) ActiveForm.Text = _resource.GetString("AppTitle")
?? string.Empty;
136:             btnAddFile.Text = _resource.GetString("AddFile") ?? string.Empty;
137:             btnAddFolder.Text = _resource.GetString("AddFolder") ?? string.Empty;
138:             btnClearList.Text = _resource.GetString("ClearFileList") ?? string.Empty;
139:             btnCombineITextSharp.Text = _resource.GetString("CombineFilesITextSharp")
?? string.Empty;
140:             btnCombinePdfSharp.Text = _resource.GetString("CombineFilesPdfSharp") ??
string.Empty;
141:             menuItemDelete.Text = _resource.GetString("Delete") ?? string.Empty;
142:             menuItemOrderByPathAscending.Text =
_resource.GetString("OrderByPathAscending") ?? string.Empty;
143:             menuItemOrderByPathDescending.Text =
_resource.GetString("OrderByPathDescending") ?? string.Empty;
144:             menuItemOrderByNameAscending.Text =
_resource.GetString("OrderByNameAscending") ?? string.Empty;
145:             menuItemOrderByNameDescending.Text =
_resource.GetString("OrderByNameDescending") ?? string.Empty;
146:             cmbLanguage.Text = _resource.GetString("SelectLanguage") ?? string.Empty;
147:         }
148:
149:         #endregion
150:
151:         #region Add Item Methods
152:
153:         /// <summary>
154:         /// It is used to add single or multiple PDF files to combine
155:         /// When you choose file or files
156:         /// The listbox in the form will be filled with the name of the files
157:         /// Which you choose to combine
158:         /// </summary>
159:         /// <param name="sender">The sender info (For example Main Form)</param>
160:         /// <param name="e">Event Arguments</param>
161:         private void btnAddFile_Click(object sender, EventArgs e)
162:         {
163:             using (var dialogAddFile = new OpenFileDialog())
164:             {
165:                 InitializeFileDialog(dialogAddFile);
166:
167:                 var result = dialogAddFile.ShowDialog();
168:                 var addedFileCount = 0;
169:                 if (result != DialogResult.OK || dialogAddFile.FileNames == null)
return;
170:
171:                 AddPdfFilesToList(dialogAddFile.FileNames, ref addedFileCount);
172:                 GenerateAddFileMessage(addedFileCount);
173:             }
174:         }
175:
176:         /// <summary>
177:         /// This function is used to add PDF files in a folder which you choose in
178:         /// Folder Dialog recursively
179:         /// When you use it the added files will be seen in listbox
180:         /// </summary>
181:         /// <param name="sender">The sender info (For example Main Form)</param>
182:         /// <param name="e">Event Arguments</param>
183:         private void btnAddFolder_Click(object sender, EventArgs e)
184:         {
185:             using (var dialogAddFolder = new FolderBrowserDialog())
186:             {
187:                 var result = dialogAddFolder.ShowDialog();
188:                 if (result != DialogResult.OK ||
string.IsNullOrEmpty(dialogAddFolder.SelectedPath)) return;
189:                 var fileNames = Directory.GetFiles(dialogAddFolder.SelectedPath,
"*.pdf",
190:                     SearchOption.AllDirectories);
191:                 var addedFileCount = 0;

```

```

192:
193:         AddPdfFilesToList(fileNames, ref addedFileCount);
194:         GenerateAddFileMessage(addedFileCount);
195:     }
196: }
197:
198: /// <summary>
199: /// This function is used to set options of file dialog like
200: /// Filter, start path etc.
201: /// </summary>
202: /// <param name="dialogAddFile">File Dialog</param>
203: private static void InitializeFileDialog(OpenFileDialog dialogAddFile)
204: {
205:     dialogAddFile.Multiselect = true;
206:     dialogAddFile.Filter = _resource.GetString("PdfFiles") + @" (*.pdf)|*.pdf";
207:     dialogAddFile.InitialDirectory = Application.StartupPath;
208:     dialogAddFile.Title = _resource.GetString("SelectPdfFile");
209:     dialogAddFile.DefaultExt = "PDF";
210: }
211:
212: /// <summary>
213: /// This method is used both adding files or folders to list
214: /// </summary>
215: /// <param name="fileNames">List Of File Names</param>
216: /// <param name="addedFileCount">Added File Count For Progress Bar</param>
217: private void AddPdfFilesToList(ICollection<string> fileNames, ref int
addedFileCount)
218: {
219:     foreach (var file in fileNames)
220:     {
221:         if (lbFiles.Items.Contains(file)) continue;
222:         lbFiles.Items.Add(file);
223:         addedFileCount++;
224:
225:         pbFiles.Value = addedFileCount * 100 / fileNames.Count;
226:         if (pbFiles.Value > pbFiles.Maximum)
227:             pbFiles.Value = pbFiles.Maximum;
228:         if (ActiveForm != null) ActiveForm.Text = @"%" + pbFiles.Value;
229:     }
230: }
231:
232: /// <summary>
233: /// This method is used to show info message after adding files or folders
234: /// </summary>
235: /// <param name="addedFileCount">Added File Count</param>
236: private void GenerateAddFileMessage(int addedFileCount)
237: {
238:     MessageBox.Show(addedFileCount + _resource.GetString("FileAddMessage"),
239:         _resource.GetString("AppTitle"), MessageBoxButtons.OK,
240:         MessageBoxIcon.Information);
241:
242:     pbFiles.Value = pbFiles.Minimum;
243:     if (ActiveForm != null) ActiveForm.Text = _resource.GetString("AppTitle");
244: }
245: #endregion
246:
247: #region Combine PDF Methods
248:
249: /// <summary>
250: /// This function is used to take list of PDF files in listbox,
251: /// Combine them with a single file in location which you choose in folder
dialog
252: /// And you can see the progress in progress bar
253: /// It uses PdfSharp Nuget Package
254: /// </summary>
255: /// <param name="sender">The sender info (For example Main Form)</param>
256: /// <param name="e">Event Arguments</param>
257: private void btnCombinePdfSharp_Click(object sender, EventArgs e)

```

```

258:         {
259:             try
260:             {
261:                 if (lbFiles.Items.Count < 2)
262:                     GenerateCombineWarningMessage();
263:                 else
264:                 {
265:                     using (var dialogExport = new FolderBrowserDialog())
266:                     {
267:                         var result = dialogExport.ShowDialog();
268:                         if (result != DialogResult.OK ||
string.IsNullOrEmpty(dialogExport.SelectedPath)) return;
269:
270:                         SetInitialValuesForCombiningFiles(dialogExport, out var
outputFileName, out var fileCount,
271:                             out var combinedFiles);
272:
273:                         using (var outputFile = new PdfDocument())
274:                         {
275:                             SetOutputFilePropertiesPdfSharp(outputFile);
276:
277:                             foreach (var t in lbFiles.Items)
278:                             {
279:                                 try
280:                                 {
281:                                     AddPdfContentToTargetPdfFilePdfSharp(t,
outputFile, ref combinedFiles);
282:                                 }
283:                                 catch (Exception)
284:                                 {
285:                                     fileCount--;
286:                                 }
287:
288:                                 SetCombinationRatio(combinedFiles, fileCount);
289:                             }
290:
291:                             SaveOutputPdfFile(outputFile, outputFileName);
292:                         }
293:
294:                         GenerateCombineFileMessage(fileCount, outputFileName);
295:                     }
296:                 }
297:             }
298:             catch (Exception ex)
299:             {
300:                 GenerateExceptionMessage(ex);
301:             }
302:         }
303:
304:         /// <summary>
305:         /// This function is used to take list of PDF files in listbox,
306:         /// Combine them with a single file in location which you choose in folder
dialog
307:         /// And you can see the progress in progress bar
308:         /// It uses iTextSharp Nuget Package
309:         /// </summary>
310:         /// <param name="sender">The sender info (For example Main Form)</param>
311:         /// <param name="e">Event Arguments</param>
312:         private void btnCombineITextSharp_Click(object sender, EventArgs e)
313:         {
314:             try
315:             {
316:                 if (lbFiles.Items.Count < 2)
317:                     GenerateCombineWarningMessage();
318:                 else
319:                 {
320:                     using (var dialogExport = new FolderBrowserDialog())
321:                     {
322:                         var result = dialogExport.ShowDialog();

```

```

323:                                     if (result != DialogResult.OK ||
string.IsNullOrEmpty(dialogExport.SelectedPath)) return;
324:
325:                                     SetInitialValuesForCombiningFiles(dialogExport, out var
outputFileName, out var fileCount,
326:                                     out var combinedFiles);
327:
328:                                     var outputFile = new Document();
329:                                     using (var outputFileStream = new FileStream(outputFileName,
FileMode.Create))
330:                                     {
331:                                         using (var pdfWriter = new PdfCopy(outputFile,
outputFileStream))
332:                                         {
333:                                             pdfWriter.SetFullCompression();
334:                                             outputFile.Open();
335:                                             foreach (var t in lbFiles.Items)
336:                                             {
337:                                                 try
338:                                                 {
339:                                                     AddPdfContentToTargetPdfFileITextSharp(t,
pdfWriter, ref combinedFiles);
340:                                                 }
341:                                                 catch (Exception)
342:                                                 {
343:                                                     fileCount--;
344:                                                 }
345:
346:                                             SetCombinationRatio(combinedFiles, fileCount);
347:                                         }
348:
349:                                         pdfWriter.Close();
350:                                     }
351:
352:                                     outputFile.Close();
353:                                 }
354:
355:                                     GenerateCombineFileMessage(fileCount, outputFileName);
356:                                 }
357:                            }
358:
359:                            catch (Exception ex)
360:                            {
361:                                GenerateExceptionMessage(ex);
362:                            }
363:                        }
364:
365:                        /// <summary>
366:                        /// This method is used to
367:                        /// Save Combined PDF File
368:                        /// </summary>
369:                        /// <param name="outputFile">Combined PDF File</param>
370:                        /// <param name="outputFileName">Combined PDF File Path</param>
371:                        private static void SaveOutputPdfFile(PdfDocument outputFile, string
outputFileName)
372:                        {
373:                            outputFile.Save(outputFileName);
374:                            outputFile.Close();
375:                        }
376:
377:                        /// <summary>
378:                        /// This method is used to take source pdf file content
379:                        /// And add to the target combined pdf file
380:                        /// </summary>
381:                        /// <param name="t">Source PDF File Path</param>
382:                        /// <param name="outputFile">Combined PDF File</param>
383:                        /// <param name="combinedFiles">Combined File Count</param>
384:                        private static void AddPdfContentToTargetPdfFilePdfSharp(object t, PdfDocument
outputFile, ref int combinedFiles)

```

```

385:         {
386:             var inputDocument = PdfReader.Open(t.ToString(),
387:                 PdfDocumentOpenMode.Import);
388:             var count = inputDocument.PageCount;
389:             for (var idx = 0; idx < count; idx++)
390:             {
391:                 var page = inputDocument.Pages[idx];
392:                 outputFile.AddPage(page);
393:             }
394:
395:             inputDocument.Close();
396:             combinedFiles++;
397:         }
398:
399:         /// <summary>
400:         /// This method is used to set combined PDF file
401:         /// Properties like Compression etc.
402:         /// </summary>
403:         /// <param name="outputFile">Combined PDF File</param>
404:         private static void SetOutputFilePropertiesPdfSharp(PdfDocument outputFile)
405:         {
406:             outputFile.Options.CompressContentStreams = true;
407:             outputFile.Options.EnableCcittCompressionForBilevelImages = true;
408:             outputFile.Options.FlateEncodeMode = PdfFlateEncodeMode.BestCompression;
409:         }
410:
411:         /// <summary>
412:         /// This method is used to take source pdf file content
413:         /// And add to the target combined pdf file
414:         /// </summary>
415:         /// <param name="t">Source PDF File Path</param>
416:         /// <param name="pdfWriter">PDF Writer</param>
417:         /// <param name="combinedFiles">Combined File Count</param>
418:         private static void AddPdfContentToTargetPdfFileITextSharp(object t, PdfCopy
pdfWriter, ref int combinedFiles)
419:         {
420:             var pdfReader =
421:                 new iTextSharp.text.pdf.PdfReader(t.ToString());
422:             pdfReader.ConsolidateNamedDestinations();
423:             for (var j = 1; j <= pdfReader.NumberOfPages; j++)
424:             {
425:                 var page = pdfWriter.GetImportedPage(pdfReader, j);
426:                 pdfWriter.AddPage(page);
427:             }
428:
429:             pdfReader.Close();
430:             combinedFiles++;
431:         }
432:
433:         /// <summary>
434:         /// This method is used
435:         /// When user just want to combine 1 pdf file
436:         /// </summary>
437:         private static void GenerateCombineWarningMessage()
438:         {
439:             MessageBox.Show(_resource.GetString("CombineWarning"),
_resource.GetString("AppTitle"),
440:                 MessageBoxButtons.OK, MessageBoxIcon.Error);
441:         }
442:
443:         /// <summary>
444:         /// This method is used to
445:         /// Show error message when exception occurs and set form title initial value
446:         /// </summary>
447:         /// <param name="ex">Exception information</param>
448:         private void GenerateExceptionMessage(Exception ex)
449:         {
450:             MessageBox.Show(_resource.GetString("CombineErrorMessage") +
ex.GetAllMessages(),

```

```

451:         _resource.GetString("AppTitle"), MessageBoxButtons.OK,
MessageBoxIcon.Error);
452:         pbFiles.Value = pbFiles.Minimum;
453:         if (ActiveForm != null) ActiveForm.Text = _resource.GetString("AppTitle");
454:     }
455:
456:     /// <summary>
457:     /// This method is used to set initial values of some properties
458:     /// Which are used in combining PDF files
459:     /// In both PdfSharp and ITextSharp
460:     /// </summary>
461:     /// <param name="dialogExport">Export Dialog Info</param>
462:     /// <param name="outputFileName">Output File Name</param>
463:     /// <param name="fileCount">Total File Count</param>
464:     /// <param name="combinedFiles">Combined Files Count</param>
465:     private void SetInitialValuesForCombiningFiles(FolderBrowserDialog
dialogExport, out string outputFileName,
466:         out int fileCount, out int combinedFiles)
467:     {
468:         outputFileName = dialogExport.SelectedPath + "/" + Guid.NewGuid() + ".pdf";
469:         fileCount = lbFiles.Items.Count;
470:         combinedFiles = 0;
471:     }
472:
473:     /// <summary>
474:     /// This method is used to set combination degree while combining PDF files
475:     /// And show this ratio on Form Caption
476:     /// </summary>
477:     /// <param name="combinedFiles">Combined File Count</param>
478:     /// <param name="fileCount">Total File Count</param>
479:     private void SetCombinationRatio(int combinedFiles, int fileCount)
480:     {
481:         pbFiles.Value = combinedFiles * 100 / fileCount;
482:         if (pbFiles.Value > pbFiles.Maximum)
483:             pbFiles.Value = pbFiles.Maximum;
484:         if (ActiveForm != null) ActiveForm.Text = @"%" + pbFiles.Value;
485:     }
486:
487:     /// <summary>
488:     /// This method is used to set successfull combining file message
489:     /// And set value of Form Caption etc.
490:     /// </summary>
491:     /// <param name="fileCount">Total File Count</param>
492:     /// <param name="outputFileName">Output File Name</param>
493:     private void GenerateCombineFileMessage(int fileCount, string outputFileName)
494:     {
495:         MessageBox.Show(fileCount + _resource.GetString("CombineFileMessage") +
outputFileName,
496:             _resource.GetString("AppTitle"), MessageBoxButtons.OK,
MessageBoxIcon.Information);
497:         pbFiles.Value = pbFiles.Minimum;
498:         if (ActiveForm != null) ActiveForm.Text = _resource.GetString("AppTitle");
499:     }
500:
501:     #endregion
502:
503:     #region Delete Item Methods
504:
505:     /// <summary>
506:     /// This function is used to operate some process in files in listbox which
are chosen
507:     /// </summary>
508:     /// <param name="sender">The sender info (For example Main Form)</param>
509:     /// <param name="e">Event Arguments</param>
510:     private void lbFiles_KeyDown(object sender, KeyEventArgs e)
511:     {
512:         // this is used to delete selected files in listbox when you press Delete
key
513:         if (e.KeyValue == Keys.Delete.GetHashCode())

```



```

514:         DeleteFilesFromListBox();
515:         // This is used to select all items in listbox when you press Ctrl + A key
combination
516:         if (!e.Control || e.KeyCode != Keys.A) return;
517:         for (var i = 0; i < lbFiles.Items.Count; i++)
518:         {
519:             lbFiles.SetSelected(i, true);
520:         }
521:     }
522:
523:     /// <summary>
524:     /// This is used to delete selected items in listbox
525:     /// When menu item Delete is clicked and opened dialog
526:     /// </summary>
527:     /// <param name="sender">The sender info (For example Main Form)</param>
528:     /// <param name="e">Event Arguments</param>
529:     private void menuItemDelete_Click(object sender, EventArgs e)
530:     {
531:         if (lbFiles.Items.Count > 0)
532:         {
533:             if (lbFiles.SelectedItems.Count > 0)
534:             {
535:                 ShowDeleteDialog(out var deleteDialog);
536:                 if (deleteDialog == DialogResult.Yes)
537:                     DeleteFilesFromListBox();
538:             }
539:             else
540:                 GenerateNoSelectedFileInListBoxMessage();
541:         }
542:         else
543:             GenerateNoFileInListBoxMessage();
544:     }
545:
546:     /// <summary>
547:     /// This method is used in
548:     /// When there is no selected file in listbox
549:     /// </summary>
550:     private static void GenerateNoSelectedFileInListBoxMessage()
551:     {
552:         MessageBox.Show(_resource.GetString("NoSelectedFile"),
553:             _resource.GetString("AppTitle"),
554:             MessageBoxButtons.OK, MessageBoxIcon.Error);
555:     }
556:
557:     /// <summary>
558:     /// This method is used to
559:     /// Show delete file dialog
560:     /// </summary>
561:     /// <param name="deleteDialog">Delete File Dialog</param>
562:     private void ShowDeleteDialog(out DialogResult deleteDialog)
563:     {
564:         deleteDialog = MessageBox.Show(
565:             _resource.GetString("DeleteWarning1") + lbFiles.SelectedItems.Count +
566:             _resource.GetString("DeleteWarning2"), _resource.GetString("AppTitle"),
567:             MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
568:     }
569:
570:     /// <summary>
571:     /// This method is used to Delete selected files from ListBox
572:     /// This is used different ways in this app
573:     /// So we created this as a method
574:     /// </summary>
575:     private void DeleteFilesFromListBox()
576:     {
577:         var fileCount = lbFiles.SelectedItems.Count;
578:         var removedItems = (from object t in lbFiles.SelectedItems select
579:             t.ToString()).ToList();
580:         var allItems = (from object item in lbFiles.Items where removedItems.All(j

```



```

=> j != item.ToString()) select item.ToString()).ToList();
580:
581:         lbFiles.Items.Clear();
582:         foreach (var item in allItems)
583:         {
584:             lbFiles.Items.Add(item);
585:         }
586:
587:         GenerateDeleteItemMessage(fileCount);
588:     }
589:
590:     /// <summary>
591:     /// This function is used to clear the file list in listbox
592:     /// </summary>
593:     /// <param name="sender">The sender info (For example Main Form)</param>
594:     /// <param name="e">Event Arguments</param>
595:     private void btnClearList_Click(object sender, EventArgs e)
596:     {
597:         var fileCount = lbFiles.Items.Count;
598:         lbFiles.Items.Clear();
599:         GenerateDeleteItemMessage(fileCount);
600:     }
601:
602:     /// <summary>
603:     /// This method is used after deleting items from listbox
604:     /// </summary>
605:     /// <param name="fileCount">Deleted File Count Information</param>
606:     private static void GenerateDeleteItemMessage(int fileCount)
607:     {
608:         MessageBox.Show(fileCount + _resource.GetString("DeleteFileMessage"),
609:             _resource.GetString("AppTitle"),
610:             MessageBoxButtons.OK, MessageBoxIcon.Information);
611:     }
612:     #endregion
613:
614:     #region Order Item Methods
615:
616:     /// <summary>
617:     /// This method is used to order elements in listbox ascending by full path
618:     /// </summary>
619:     /// <param name="sender">The sender info (For example Main Form)</param>
620:     /// <param name="e">Event Arguments</param>
621:     private void menuItemOrderByPathAscending_Click(object sender, EventArgs e)
622:     {
623:         if (lbFiles.Items.Count > 0)
624:             lbFiles = SortItemsByPath(lbFiles, OrderType.Ascending);
625:         else
626:             GenerateNoFileInListBoxMessage();
627:     }
628:
629:     /// <summary>
630:     /// This method is used to order elements in listbox descending by full path
631:     /// </summary>
632:     /// <param name="sender">The sender info (For example Main Form)</param>
633:     /// <param name="e">Event Arguments</param>
634:     private void menuItemOrderByPathDescending_Click(object sender, EventArgs e)
635:     {
636:         if (lbFiles.Items.Count > 0)
637:             lbFiles = SortItemsByPath(lbFiles, OrderType.Descending);
638:         else
639:             GenerateNoFileInListBoxMessage();
640:     }
641:
642:     /// <summary>
643:     /// This method is used to order elements in listbox ascending by file name
644:     /// </summary>
645:     /// <param name="sender">The sender info (For example Main Form)</param>
646:     /// <param name="e">Event Arguments</param>

```

```

647:         private void menuItemOrderByNameAscending_Click(object sender, EventArgs e)
648:         {
649:             if (lbFiles.Items.Count > 0)
650:                 lbFiles = SortItemsByName(lbFiles, OrderType.Ascending);
651:             else
652:                 GenerateNoFileInListBoxMessage();
653:         }
654:
655:         /// <summary>
656:         /// This method is used to order elements in listbox descending by file name
657:         /// </summary>
658:         /// <param name="sender">The sender info (For example Main Form)</param>
659:         /// <param name="e">Event Arguments</param>
660:         private void menuItemOrderByNameDescending_Click(object sender, EventArgs e)
661:         {
662:             if (lbFiles.Items.Count > 0)
663:                 lbFiles = SortItemsByName(lbFiles, OrderType.Descending);
664:             else
665:                 GenerateNoFileInListBoxMessage();
666:         }
667:
668:         /// <summary>
669:         /// This method is used to generate message
670:         /// When there is no item in listbox
671:         /// </summary>
672:         private static void GenerateNoFileInListBoxMessage()
673:         {
674:             MessageBox.Show(_resource.GetString("NoFile"),
675:                 _resource.GetString("AppTitle"), MessageBoxButtons.OK,
676:                 MessageBoxIcon.Error);
677:         }
678:
679:         /// <summary>
680:         /// This method is used to order items by full path in listbox
681:         /// And return them with the given order as parametre
682:         /// </summary>
683:         /// <param name="listBox">ListBox Info</param>
684:         /// <param name="orderType">Order Type is Ascending Or Not</param>
685:         private static ListBox SortItemsByPath(ListBox listBox, OrderType orderType)
686:         {
687:             var items = listBox.Items.OfType<object>().ToList();
688:             listBox.Items.Clear();
689:             listBox.Items.AddRange(orderType == OrderType.Ascending
690:                 ? items.OrderBy(i => i).ToArray()
691:                 : items.OrderByDescending(i => i).ToArray());
692:             return listBox;
693:         }
694:
695:         /// <summary>
696:         /// This method is used to order items by file name in listbox
697:         /// And return them with the given order as parametre
698:         /// </summary>
699:         /// <param name="listBox">ListBox Info</param>
700:         /// <param name="orderType">Order Type is Ascending Or Not</param>
701:         private static ListBox SortItemsByName(ListBox listBox, OrderType orderType)
702:         {
703:             var items = listBox.Items.OfType<object>().ToList();
704:             const char c = '\u005c';
705:             var fileList = (from item in items select item.ToString() into fullPath
706:                 let list = fullPath.Split(c).ToList() where list.Count > 0 select new FileInfo {FilePath =
707:                 fullPath, FileName = list[list.Count - 1]}).ToList();
708:
709:             fileList = orderType == OrderType.Ascending
710:                 ? fileList.OrderBy(j => j.FileName).ThenBy(j => j.FilePath).ToList()
711:                 : fileList.OrderByDescending(j => j.FileName).ThenByDescending(j =>
712:                 j.FilePath).ToList();
713:
714:             listBox.Items.Clear();
715:             foreach (var item in fileList)

```

```

712:         listBox.Items.Add(item.FilePath);
713:
714:         return listBox;
715:     }
716:
717: #endregion
718:
719: #region Item Drag and Drop Methods
720:
721:     /// <summary>
722:     /// This is used to control mouse actions and if that is not right click
723:     /// Start to drag and drop item in listbox
724:     /// </summary>
725:     /// <param name="sender">The sender info (For example Main Form)</param>
726:     /// <param name="e">Event Arguments</param>
727:     private void lbFiles_MouseDown(object sender, MouseEventArgs e)
728:     {
729:         if (e.Button.Equals(MouseButtons.Right)) return;
730:         if (lbFiles.SelectedItem == null) return;
731:         lbFiles.DoDragDrop(lbFiles.SelectedItem, DragDropEffects.Move);
732:     }
733:
734:     /// <summary>
735:     /// This is used to give effect while dragging and dropping an item in listbox
736:     /// </summary>
737:     /// <param name="sender">The sender info (For example Main Form)</param>
738:     /// <param name="e">Event Arguments</param>
739:     private void lbFiles_DragOver(object sender, DragEventArgs e)
740:     {
741:         e.Effect = DragDropEffects.Move;
742:     }
743:
744:     /// <summary>
745:     /// This is used to drag and drop an item in listbox
746:     /// Which is used to order books and combine them with that order
747:     /// </summary>
748:     /// <param name="sender">The sender info (For example Main Form)</param>
749:     /// <param name="e">Event Arguments</param>
750:     private void lbFiles_DragDrop(object sender, DragEventArgs e)
751:     {
752:         var point = lbFiles.PointToClient(new Point(e.X, e.Y));
753:         var index = lbFiles.IndexFromPoint(point);
754:         if (index < 0) index = lbFiles.Items.Count - 1;
755:         // type of string because file names stored in string in listbox
756:         var data = e.Data.GetData(typeof(string));
757:         lbFiles.Items.Remove(data);
758:         lbFiles.Items.Insert(index, data);
759:     }
760:
761: #endregion
762: }
763: }

```

### Namespaces

Name	Description
PdfCombiner (see page 1)	This is namespace PdfCombiner.

## 1.2.7 FrmMain.Designer.cs

This is file FrmMain.Designer.cs.

### Body Source

```
1: ?namespace PdfCombiner
```

```

2: {
3:     partial class FrmMain
4:     {
5:         /// <summary>
6:         /// Required designer variable.
7:         /// </summary>
8:         private System.ComponentModel.IContainer components;
9:
10:        /// <summary>
11:        /// Clean up any resources being used.
12:        /// </summary>
13:        /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
14:        protected override void Dispose(bool disposing)
15:        {
16:            if (disposing && (components != null))
17:            {
18:                components.Dispose();
19:            }
20:            base.Dispose(disposing);
21:        }
22:
23:        #region Windows Form Designer generated code
24:
25:        /// <summary>
26:        /// Required method for Designer support - do not modify
27:        /// the contents of this method with the code editor.
28:        /// </summary>
29:        private void InitializeComponent()
30:        {
31:            this.components = new System.ComponentModel.Container();
32:            System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(FrmMain));
33:            this.btnCombinePdfSharp = new System.Windows.Forms.Button();
34:            this.btnAddFolder = new System.Windows.Forms.Button();
35:            this.btnAddFile = new System.Windows.Forms.Button();
36:            this.btnClearList = new System.Windows.Forms.Button();
37:            this.lbFiles = new System.Windows.Forms.ListBox();
38:            this.lblDetails = new System.Windows.Forms.Label();
39:            this.pbFiles = new System.Windows.Forms.ProgressBar();
40:            this.btnCombineITextSharp = new System.Windows.Forms.Button();
41:            this.menuStrip = new
System.Windows.Forms.ContextMenuStrip(this.components);
42:            this.menuItemDelete = new System.Windows.Forms.ToolStripMenuItem();
43:            this.menuItemOrderByPathAscending = new
System.Windows.Forms.ToolStripMenuItem();
44:            this.menuItemOrderByPathDescending = new
System.Windows.Forms.ToolStripMenuItem();
45:            this.menuItemOrderByNameAscending = new
System.Windows.Forms.ToolStripMenuItem();
46:            this.menuItemOrderByNameDescending = new
System.Windows.Forms.ToolStripMenuItem();
47:            this.cmbLanguage = new System.Windows.Forms.ComboBox();
48:            this.menuStrip.SuspendLayout();
49:            this.SuspendLayout();
50:            //
51:            // btnCombinePdfSharp
52:            //
53:            this.btnCombinePdfSharp.Image = ((System.Drawing.Image)
(resources.GetObject("btnCombinePdfSharp.Image")));
54:            this.btnCombinePdfSharp.ImageAlign =
System.Drawing.ContentAlignment.MiddleLeft;
55:            this.btnCombinePdfSharp.Location = new System.Drawing.Point(442, 5);
56:            this.btnCombinePdfSharp.Name = "btnCombinePdfSharp";
57:            this.btnCombinePdfSharp.Size = new System.Drawing.Size(214, 37);
58:            this.btnCombinePdfSharp.TabIndex = 0;
59:            this.btnCombinePdfSharp.Text = "Combine Files (PdfSharp)";
60:            this.btnCombinePdfSharp.TextAlign =
System.Drawing.ContentAlignment.MiddleRight;

```

```

61:         this.btnCombinePdfSharp.UseVisualStyleBackColor = true;
62:         this.btnCombinePdfSharp.Click += new
System.EventHandler(this.btnCombinePdfSharp_Click);
63:         //
64:         // btnAddFolder
65:         //
66:         this.btnAddFolder.Image = ((System.Drawing.Image)
(resources.GetObject("btnAddFolder.Image")));
67:         this.btnAddFolder.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
68:         this.btnAddFolder.Location = new System.Drawing.Point(197, 5);
69:         this.btnAddFolder.Name = "btnAddFolder";
70:         this.btnAddFolder.Size = new System.Drawing.Size(171, 37);
71:         this.btnAddFolder.TabIndex = 2;
72:         this.btnAddFolder.Text = "Add Folder";
73:         this.btnAddFolder.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
74:         this.btnAddFolder.UseVisualStyleBackColor = true;
75:         this.btnAddFolder.Click += new
System.EventHandler(this.btnAddFolder_Click);
76:         //
77:         // btnAddFile
78:         //
79:         this.btnAddFile.Image = ((System.Drawing.Image)
(resources.GetObject("btnAddFile.Image")));
80:         this.btnAddFile.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
81:         this.btnAddFile.Location = new System.Drawing.Point(12, 5);
82:         this.btnAddFile.Name = "btnAddFile";
83:         this.btnAddFile.Size = new System.Drawing.Size(179, 37);
84:         this.btnAddFile.TabIndex = 3;
85:         this.btnAddFile.Text = "Add Files";
86:         this.btnAddFile.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
87:         this.btnAddFile.UseVisualStyleBackColor = true;
88:         this.btnAddFile.Click += new System.EventHandler(this.btnAddFile_Click);
89:         //
90:         // btnClearList
91:         //
92:         this.btnClearList.Image = ((System.Drawing.Image)
(resources.GetObject("btnClearList.Image")));
93:         this.btnClearList.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
94:         this.btnClearList.Location = new System.Drawing.Point(12, 45);
95:         this.btnClearList.Name = "btnClearList";
96:         this.btnClearList.Size = new System.Drawing.Size(179, 37);
97:         this.btnClearList.TabIndex = 4;
98:         this.btnClearList.Text = "Clear File List";
99:         this.btnClearList.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
100:        this.btnClearList.UseVisualStyleBackColor = true;
101:        this.btnClearList.Click += new
System.EventHandler(this.btnClearList_Click);
102:        //
103:        // lbFiles
104:        //
105:        this.lbFiles.AllowDrop = true;
106:        this.lbFiles.FormattingEnabled = true;
107:        this.lbFiles.Location = new System.Drawing.Point(12, 88);
108:        this.lbFiles.Name = "lbFiles";
109:        this.lbFiles.SelectionMode =
System.Windows.Forms.SelectionMode.MultiExtended;
110:        this.lbFiles.Size = new System.Drawing.Size(644, 186);
111:        this.lbFiles.TabIndex = 5;
112:        this.lbFiles.DragDrop += new
System.Windows.Forms.DragEventHandler(this.lbFiles_DragDrop);
113:        this.lbFiles.DragOver += new
System.Windows.Forms.DragEventHandler(this.lbFiles_DragOver);
114:        this.lbFiles.KeyDown += new
System.Windows.Forms.KeyEventHandler(this.lbFiles_KeyDown);
115:        this.lbFiles.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.lbFiles_MouseDown);
116:        //
117:        // lblDetails
118:        //

```

```
119:         this.lblDetails.AutoSize = true;
120:         this.lblDetails.Location = new System.Drawing.Point(12, 65);
121:         this.lblDetails.Name = "lblDetails";
122:         this.lblDetails.Size = new System.Drawing.Size(0, 13);
123:         this.lblDetails.TabIndex = 6;
124:         //
125:         // pbFiles
126:         //
127:         this.pbFiles.Location = new System.Drawing.Point(197, 45);
128:         this.pbFiles.Name = "pbFiles";
129:         this.pbFiles.Size = new System.Drawing.Size(239, 37);
130:         this.pbFiles.TabIndex = 7;
131:         //
132:         // btnCombineITextSharp
133:         //
134:         this.btnCombineITextSharp.Image = ((System.Drawing.Image)
(resources.GetObject("btnCombineITextSharp.Image")));
135:         this.btnCombineITextSharp.ImageAlign =
System.Drawing.ContentAlignment.MiddleLeft;
136:         this.btnCombineITextSharp.Location = new System.Drawing.Point(442, 45);
137:         this.btnCombineITextSharp.Name = "btnCombineITextSharp";
138:         this.btnCombineITextSharp.Size = new System.Drawing.Size(214, 37);
139:         this.btnCombineITextSharp.TabIndex = 9;
140:         this.btnCombineITextSharp.Text = "Combine Files (iTextSharp)";
141:         this.btnCombineITextSharp.TextAlign =
System.Drawing.ContentAlignment.MiddleRight;
142:         this.btnCombineITextSharp.UseVisualStyleBackColor = true;
143:         this.btnCombineITextSharp.Click += new
System.EventHandler(this.btnCombineITextSharp_Click);
144:         //
145:         // menuStrip
146:         //
147:         this.menuStrip.Items.AddRange(new System.Windows.Forms.ToolStripItem[]
{this.menuItemDelete, this.menuItemOrderByPathAscending,
this.menuItemOrderByPathDescending, this.menuItemOrderByNameAscending,
this.menuItemOrderByNameDescending});
148:         this.menuStrip.Name = "menuStrip";
149:         this.menuStrip.Size = new System.Drawing.Size(229, 114);
150:         //
151:         // menuItemDelete
152:         //
153:         this.menuItemDelete.Image = ((System.Drawing.Image)
(resources.GetObject("menuItemDelete.Image")));
154:         this.menuItemDelete.Name = "menuItemDelete";
155:         this.menuItemDelete.Size = new System.Drawing.Size(228, 22);
156:         this.menuItemDelete.Text = "Delete";
157:         this.menuItemDelete.Click += new
System.EventHandler(this.menuItemDelete_Click);
158:         //
159:         // menuItemOrderByPathAscending
160:         //
161:         this.menuItemOrderByPathAscending.Image = ((System.Drawing.Image)
(resources.GetObject("menuItemOrderByPathAscending.Image")));
162:         this.menuItemOrderByPathAscending.Name = "menuItemOrderByPathAscending";
163:         this.menuItemOrderByPathAscending.Size = new System.Drawing.Size(228, 22);
164:         this.menuItemOrderByPathAscending.Text = "Order By Path (Ascending)";
165:         this.menuItemOrderByPathAscending.Click += new
System.EventHandler(this.menuItemOrderByPathAscending_Click);
166:         //
167:         // menuItemOrderByPathDescending
168:         //
169:         this.menuItemOrderByPathDescending.Image = ((System.Drawing.Image)
(resources.GetObject("menuItemOrderByPathDescending.Image")));
170:         this.menuItemOrderByPathDescending.Name = "menuItemOrderByPathDescending";
171:         this.menuItemOrderByPathDescending.Size = new System.Drawing.Size(228, 22);
172:         this.menuItemOrderByPathDescending.Text = "Order By Path (Descending)";
173:         this.menuItemOrderByPathDescending.Click += new
System.EventHandler(this.menuItemOrderByPathDescending_Click);
174:         //
```



```

175:         // menuItemOrderByNameAscending
176:         //
177:         this.menuItemOrderByNameAscending.Image = ((System.Drawing.Image)
(resources.GetObject("menuItemOrderByNameAscending.Image")));
178:         this.menuItemOrderByNameAscending.Name = "menuItemOrderByNameAscending";
179:         this.menuItemOrderByNameAscending.Size = new System.Drawing.Size(228, 22);
180:         this.menuItemOrderByNameAscending.Text = "Order By Name (Ascending)";
181:         this.menuItemOrderByNameAscending.Click += new
System.EventHandler(this.menuItemOrderByNameAscending_Click);
182:         //
183:         // menuItemOrderByNameDescending
184:         //
185:         this.menuItemOrderByNameDescending.Image = ((System.Drawing.Image)
(resources.GetObject("menuItemOrderByNameDescending.Image")));
186:         this.menuItemOrderByNameDescending.Name = "menuItemOrderByNameDescending";
187:         this.menuItemOrderByNameDescending.Size = new System.Drawing.Size(228, 22);
188:         this.menuItemOrderByNameDescending.Text = "Order By Name (Descending)";
189:         this.menuItemOrderByNameDescending.Click += new
System.EventHandler(this.menuItemOrderByNameDescending_Click);
190:         //
191:         // cmbLanguage
192:         //
193:         this.cmbLanguage.Anchor = System.Windows.Forms.AnchorStyles.None;
194:         this.cmbLanguage.Font = new System.Drawing.Font("Microsoft Sans Serif",
15F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte) (162)));
195:         this.cmbLanguage.FormattingEnabled = true;
196:         this.cmbLanguage.Items.AddRange(new object[] { "TR", "EN", "DE", "FR",
"RU", "ES" });
197:         this.cmbLanguage.Location = new System.Drawing.Point(374, 6);
198:         this.cmbLanguage.Name = "cmbLanguage";
199:         this.cmbLanguage.Size = new System.Drawing.Size(62, 33);
200:         this.cmbLanguage.TabIndex = 10;
201:         this.cmbLanguage.Text = "Select Language";
202:         this.cmbLanguage.SelectedIndexChanged += new
System.EventHandler(this.cmbLanguage_SelectedIndexChanged);
203:         //
204:         // FrmMain
205:         //
206:         this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
207:         this.AutoScaleMode = System.Windows.Forms.AutoScaleModeMode.Font;
208:         this.ClientSize = new System.Drawing.Size(668, 287);
209:         this.Controls.Add(this.cmbLanguage);
210:         this.Controls.Add(this.btnCombineITextSharp);
211:         this.Controls.Add(this.pbFiles);
212:         this.Controls.Add(this.lblDetails);
213:         this.Controls.Add(this.lbFiles);
214:         this.Controls.Add(this.btnClearList);
215:         this.Controls.Add(this.btnAddFile);
216:         this.Controls.Add(this.btnAddFolder);
217:         this.Controls.Add(this.btnCombinePdfSharp);
218:         this.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte) (162)));
219:         this.Name = "FrmMain";
220:         this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
221:         this.Text = "PDF Combiner";
222:         this.FormClosed += new
System.Windows.Forms.FormClosedEventHandler(this.FrmMain_FormClosed);
223:         this.Shown += new System.EventHandler(this.FrmMain_Shown);
224:         this.menuStrip.ResumeLayout(false);
225:         this.ResumeLayout(false);
226:         this.PerformLayout();
227:     }
228:
229:     public System.Windows.Forms.ComboBox cmbLanguage;
230:
231:     #endregion
232:
233:     private System.Windows.Forms.Button btnCombinePdfSharp;
234:     private System.Windows.Forms.Button btnAddFolder;

```

```
235:         private System.Windows.Forms.Button btnAddFile;
236:         private System.Windows.Forms.Button btnClearList;
237:         private System.Windows.Forms.ListBox lbFiles;
238:         private System.Windows.Forms.Label lblDetails;
239:         private System.Windows.Forms.ProgressBar pbFiles;
240:         private System.Windows.Forms.Button btnCombineITextSharp;
241:         private System.Windows.Forms.ContextMenuStrip menuStrip;
242:         private System.Windows.Forms.ToolStripMenuItem menuItemDelete;
243:         private System.Windows.Forms.ToolStripMenuItem menuItemOrderByPathAscending;
244:         private System.Windows.Forms.ToolStripMenuItem menuItemOrderByPathDescending;
245:         private System.Windows.Forms.ToolStripMenuItem menuItemOrderByNameAscending;
246:         private System.Windows.Forms.ToolStripMenuItem menuItemOrderByNameDescending;
247:     }
248: }
249:
```

Namespaces

Name	Description
PdfCombiner ( <a href="#">see page 1</a> )	This is namespace PdfCombiner.

## 1.2.8 PdfCombiner.csproj

This is file PdfCombiner.csproj.

## 1.2.9 PdfCombiner.sln

This is file PdfCombiner.sln.

## 1.2.10 Program.cs

This is file Program.cs.

Body Source

```
1: using System;
2: using System.Windows.Forms;
3:
4: namespace PdfCombiner
5: {
6:     internal static class Program
7:     {
8:         /// <summary>
9:         /// The main entry point for the application.
10:        /// </summary>
11:        [STAThread]
12:        private static void Main()
13:        {
14:            Application.EnableVisualStyles();
15:            Application.SetCompatibleTextRenderingDefault(false);
16:            Application.Run(new FrmMain());
17:        }
18:    }
19: }
```



Namespaces

Name	Description
PdfCombiner (see page 1)	This is namespace PdfCombiner.

1.2.11 Resources.Designer.cs

This code was generated by a tool. Runtime Version:4.0.30319.42000

Changes to this file may cause incorrect behavior and will be lost if the code is regenerated.

Body Source

```
1: ?//-----
2: // <auto-generated>
3: //   This code was generated by a tool.
4: //   Runtime Version:4.0.30319.42000
5: //
6: //   Changes to this file may cause incorrect behavior and will be lost if
7: //   the code is regenerated.
8: // </auto-generated>
9: //-----
10:
11: namespace PdfCombiner.Properties {
12:     using System;
13:
14:
15:     /// <summary>
16:     ///   A strongly-typed resource class, for looking up localized strings, etc.
17:     /// </summary>
18:     // This class was auto-generated by the StronglyTypedResourceBuilder
19:     // class via a tool like ResGen or Visual Studio.
20:     // To add or remove a member, edit your .ResX file then rerun ResGen
21:     // with the /str option, or rebuild your VS project.
22:
23: [global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.StronglyTypedResourceBuilder",
24: "15.0.0.0")]
25: [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
26: [global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
27:     internal class Resources {
28:
29:         private static global::System.Resources.ResourceManager resourceMan;
30:
31:         private static global::System.Globalization.CultureInfo resourceCulture;
32:
33: [global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
34: "CA1811:AvoidUncalledPrivateCode")]
35:         internal Resources() {
36:
37:             /// <summary>
38:             ///   Returns the cached ResourceManager instance used by this class.
39:             /// </summary>
40:
41: [global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]
42:             internal static global::System.Resources.ResourceManager ResourceManager {
43:                 get {
44:                     if (object.ReferenceEquals(resourceMan, null)) {
45:                         global::System.Resources.ResourceManager temp = new
46:                             global::System.Resources.ResourceManager("PdfCombiner.Properties.Resources",
47:                             typeof(Resources).Assembly);
```

```
43:         resourceMan = temp;
44:     }
45:     return resourceMan;
46: }
47:
48:
49:     /// <summary>
50:     ///     Overrides the current thread's CurrentUICulture property for all
51:     ///     resource lookups using this strongly typed resource class.
52:     /// </summary>
53:
54: [global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]
55:     internal static global::System.Globalization.CultureInfo Culture {
56:     get {
57:         return resourceCulture;
58:     }
59:     set {
60:         resourceCulture = value;
61:     }
62:
63:     /// <summary>
64:     ///     Looks up a localized string similar to PDF Birlestirici.
65:     /// </summary>
66:     internal static string AppTitle {
67:     get {
68:         return ResourceManager.GetString("AppTitle", resourceCulture);
69:     }
70: }
71: }
72: }
```

Namespaces

Name	Description
PdfCombiner (see page 1)	This is namespace PdfCombiner.

# 1.2.12 Settings.Designer.cs

This code was generated by a tool. Runtime Version:4.0.30319.42000

Changes to this file may cause incorrect behavior and will be lost if the code is regenerated.

Body Source

```
1: ?//-----
2: // <auto-generated>
3: //     This code was generated by a tool.
4: //     Runtime Version:4.0.30319.42000
5: //
6: //     Changes to this file may cause incorrect behavior and will be lost if
7: //     the code is regenerated.
8: // </auto-generated>
9: //-----
10:
11: namespace PdfCombiner.Properties
12: {
13:
14:     [global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
15:
16: [global::System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.VisualStudio.Editors.SettingsDesigner.SettingsSingleFileGenerator",
17: "11.0.0.0")]
18:     internal sealed partial class Settings :
```

```
global::System.Configuration.ApplicationSettingsBase
17:     {
18:
19:         private static Settings defaultInstance =
20:         ((Settings)(global::System.Configuration.ApplicationSettingsBase.Synchronized(new
21:         Settings())));
22:     }
```

Namespaces

Name	Description
PdfCombiner (🔗 see page 1)	This is namespace PdfCombiner.

# Index

## A

Ascending enumeration member 6  
AssemblyInfo.cs 35

## B

BaseException class 4  
    about BaseException class 4  
    BaseException 5  
BaseException.cs 35

## D

Descending enumeration member 6

## E

Enums class 5  
    about Enums class 5  
Enums.cs 36  
ExceptionExtensions class 6  
    about ExceptionExtensions class 6  
    GetAllMessages 6  
ExceptionExtensions.cs 36

## F

FileInfo class 7  
    about FileInfo class 7  
    FileName 7  
    FilePath 7  
FileInfo.cs 37  
Files 34  
FrmMain class 8  
    \_resource 10  
    about FrmMain class 8  
    AddPdfContentToTargetPdfFileITextSharp 13  
    AddPdfContentToTargetPdfFilePdfSharp 14  
    AddPdfFilesToList 14  
    btnAddFile 11  
    btnAddFile\_Click 15

btnAddFolder 11  
btnAddFolder\_Click 15  
btnClearList 11  
btnClearList\_Click 16  
btnCombineITextSharp 11  
btnCombineITextSharp\_Click 16  
btnCombinePdfSharp 11  
btnCombinePdfSharp\_Click 17  
cmbLanguage 11  
cmbLanguage\_SelectedIndexChanged 18  
components 12  
DeleteFilesFromListBox 18  
Dispose 19  
FillLanguageComboboxAndSetFirstLanguage 19  
FormMembersNameInitialization 20  
FrmMain 10  
FrmMain\_FormClosed 20  
FrmMain\_Shown 20  
GenerateAddFileMessage 21  
GenerateCombineFileMessage 21  
GenerateCombineWarningMessage 22  
GenerateDeleteItemMessage 22  
GenerateExceptionMessage 22  
GenerateNoFileInListBoxMessage 22  
GenerateNoSelectedFileInListBoxMessage 23  
InitializeComponent 23  
InitializeFileDialog 26  
LanguageList 12  
lbFiles 12  
lbFiles\_DragDrop 27  
lbFiles\_DragOver 27  
lbFiles\_KeyDown 28  
lbFiles\_MouseDown 28  
lbIDetails 12  
menuItemDelete 12  
menuItemDelete\_Click 28  
menuItemOrderByNameAscending 12  
menuItemOrderByNameAscending\_Click 29  
menuItemOrderByNameDescending 12  
menuItemOrderByNameDescending\_Click 29  
menuItemOrderByPathAscending 13

menuItemOrderByPathAscending_Click 30	PdfCombiner.FrmMain.btnAddFolder 11
menuItemOrderByPathDescending 13	PdfCombiner.FrmMain.btnAddFolder_Click 15
menuItemOrderByPathDescending_Click 30	PdfCombiner.FrmMain.btnClearList 11
menuStrip 13	PdfCombiner.FrmMain.btnClearList_Click 16
pbFiles 13	PdfCombiner.FrmMain.btnCombineTextSharp 11
SaveOutputPdfFile 30	PdfCombiner.FrmMain.btnCombineTextSharp_Click 16
SetCombinationRatio 31	PdfCombiner.FrmMain.btnCombinePdfSharp 11
SetInitialValuesForCombiningFiles 31	PdfCombiner.FrmMain.btnCombinePdfSharp_Click 17
SetOutputFilePropertiesPdfSharp 31	PdfCombiner.FrmMain.cmbLanguage 11
SetProgramLanguage 32	PdfCombiner.FrmMain.cmbLanguage_SelectedIndexChanged 18
ShowDeleteDialog 32	PdfCombiner.FrmMain.components 12
SortItemsByName 32	PdfCombiner.FrmMain.DeleteFilesFromListBox 18
SortItemsByPath 33	PdfCombiner.FrmMain.Dispose 19
FrmMain.cs 37	PdfCombiner.FrmMain.FillLanguageComboboxAndSetFirstLanguage 19
FrmMain.Designer.cs 49	PdfCombiner.FrmMain.FormMembersNameInitialization 20
<b>P</b>	PdfCombiner.FrmMain.FrmMain 10
PdfCombiner 1	PdfCombiner.FrmMain.FrmMain_FormClosed 20
PdfCombiner namespace 1	PdfCombiner.FrmMain.FrmMain_Shown 20
Classes 4	PdfCombiner.FrmMain.GenerateAddFileMessage 21
PdfCombiner.BaseException 4	PdfCombiner.FrmMain.GenerateCombineFileMessage 21
PdfCombiner.BaseException.BaseException 5	PdfCombiner.FrmMain.GenerateCombineWarningMessage 22
PdfCombiner.csproj 54	PdfCombiner.FrmMain.GenerateDeleteItemMessage 22
PdfCombiner.Enums 5	PdfCombiner.FrmMain.GenerateExceptionMessage 22
PdfCombiner.Enums.OrderType 6	PdfCombiner.FrmMain.GenerateNoFileInListBoxMessage 22
PdfCombiner.Enums.OrderType enumeration 6	PdfCombiner.FrmMain.GenerateNoSelectedFileInListBoxMessage 23
PdfCombiner.ExceptionExtensions 6	PdfCombiner.FrmMain.InitializeComponent 23
PdfCombiner.ExceptionExtensions.GetAllMessages 6	PdfCombiner.FrmMain.InitializeFileDialog 26
PdfCombiner.FileInfo 7	PdfCombiner.FrmMain.LanguageList 12
PdfCombiner.FileInfo.FileName 7	PdfCombiner.FrmMain.lbFiles 12
PdfCombiner.FileInfo.FilePath 7	PdfCombiner.FrmMain.lbFiles_DragDrop 27
PdfCombiner.FrmMain 8	PdfCombiner.FrmMain.lbFiles_DragOver 27
PdfCombiner.FrmMain._resource 10	PdfCombiner.FrmMain.lbFiles_KeyDown 28
PdfCombiner.FrmMain.AddPdfContentToTargetPdfFileTextSharp 13	PdfCombiner.FrmMain.lbFiles_MouseDown 28
PdfCombiner.FrmMain.AddPdfContentToTargetPdfFilePdfSharp 14	PdfCombiner.FrmMain.lblDetails 12
PdfCombiner.FrmMain.AddPdfFilesToList 14	PdfCombiner.FrmMain.menuItemDelete 12
PdfCombiner.FrmMain.btnAddFile 11	PdfCombiner.FrmMain.menuItemDelete_Click 28
PdfCombiner.FrmMain.btnAddFile_Click 15	PdfCombiner.FrmMain.menuItemOrderByNameAscending 12

PdfCombiner.FrmMain.menuItemOrderByNameAscending\_Click 29  
 PdfCombiner.FrmMain.menuItemOrderByNameDescending 12  
 PdfCombiner.FrmMain.menuItemOrderByNameDescending\_Click 29  
 PdfCombiner.FrmMain.menuItemOrderByPathAscending 13  
 PdfCombiner.FrmMain.menuItemOrderByPathAscending\_Click 30  
 PdfCombiner.FrmMain.menuItemOrderByPathDescending 13  
 PdfCombiner.FrmMain.menuItemOrderByPathDescending\_Click 30  
 PdfCombiner.FrmMain.menuStrip 13  
 PdfCombiner.FrmMain.pbFiles 13  
 PdfCombiner.FrmMain.SaveOutputPdfFile 30  
 PdfCombiner.FrmMain.SetCombinationRatio 31  
 PdfCombiner.FrmMain.SetInitialValuesForCombiningFiles 31  
 PdfCombiner.FrmMain.SetOutputFilePropertiesPdfSharp 31  
 PdfCombiner.FrmMain.SetProgramLanguage 32  
 PdfCombiner.FrmMain.ShowDeleteDialog 32  
 PdfCombiner.FrmMain.SortItemsByName 32  
 PdfCombiner.FrmMain.SortItemsByPath 33  
 PdfCombiner.Program 33  
 PdfCombiner.Program.Main 34  
 PdfCombiner.Properties 1  
 PdfCombiner.Properties namespace 1  
     Classes 1  
 PdfCombiner.Properties.Resources 1  
 PdfCombiner.Properties.Resources.AppTitle 3  
 PdfCombiner.Properties.Resources.Culture 3  
 PdfCombiner.Properties.Resources.resourceCulture 3  
 PdfCombiner.Properties.Resources.resourceMan 3  
 PdfCombiner.Properties.Resources.ResourceManager 3  
 PdfCombiner.Properties.Resources.Resources 2  
 PdfCombiner.Properties.Settings 3  
 PdfCombiner.Properties.Settings.defaultInstance 4  
 PdfCombiner.sln 54  
 Program class 33  
     about Program class 33  
     Main 34

## R

Resources class 1  
     about Resources class 1  
     AppTitle 3  
     Culture 3  
     resourceCulture 3  
     resourceMan 3  
     ResourceManager 3  
     Resources 2  
 Resources.Designer.cs 55

## S

Settings class 3  
     about Settings class 3  
     defaultInstance 4  
 Settings.Designer.cs 56