

# 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
Enums Class	4
Enums Enumerations	5
ExceptionExtensions Class	5
ExceptionExtensions Methods	5
FileInfo Class	6
FileInfo Properties	6
FrmMain Class	6
FrmMain.FrmMain Constructor	9
FrmMain Fields	9
FrmMain Methods	12
Program Class	32
Program Methods	32
<b>Files</b>	<b>33</b>
AssemblyInfo.cs	33
Enums.cs	34
ExceptionExtensions.cs	35
FileInfo.cs	35
FrmMain.cs	36
FrmMain.Designer.cs	48
PdfCombiner.csproj	52
PdfCombiner.sln	52
Program.cs	52
Resources.Designer.cs	53
Settings.Designer.cs	54
<b>Index</b>	<b>a</b>

# 1 Symbol Reference






## 1.1 PdfCombiner Namespace

This is namespace PdfCombiner.

### Namespaces

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



### Classes

	Name	Description
	Enums ( <a href="#">↗</a> see page 4)	This class contains enums for detailed information In usage
	ExceptionExtensions ( <a href="#">↗</a> see page 5)	This class is used to take generic exception messages With the inner exception, stack trace and source informations
	FileInfo ( <a href="#">↗</a> see page 6)	This class is used to have File's Full Path and Name
	FrmMain ( <a href="#">↗</a> see page 6)	This is class PdfCombiner.FrmMain.
	Program ( <a href="#">↗</a> see page 32)	This is class PdfCombiner.Program.

### 1.1.1 PdfCombiner.Properties Namespace

This is namespace PdfCombiner.Properties.



### Classes

	Name	Description
	Resources ( <a href="#">↗</a> see page 1)	A strongly-typed resource class, for looking up localized strings, etc.
	Settings ( <a href="#">↗</a> see page 3)	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="#">↗</a> see page 1)	A strongly-typed resource class, for looking up localized strings, etc.
	Settings ( <a href="#">↗</a> see page 3)	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 53](#))



## 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

#### 1.1.1.1.2.1 Resources.resourceCulture Field

**C#**

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

##### Description

This is resourceCulture, a member of class Resources.

#### 1.1.1.1.2.2 Resources.resourceMan Field

**C#**

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

##### Description

This is resourceMan, a member of class Resources.

### 1.1.1.1.3 Resources Properties

#### 1.1.1.1.3.1 Resources.AppTitle Property

Looks up a localized string similar to PDF Birleştirci.

**C#**

```
internal static string AppTitle;
```

#### 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;
```

#### 1.1.1.1.3.3 Resources.ResourceManager Property

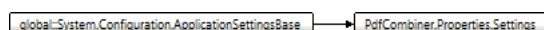
Returns the cached ResourceManager instance used by this class.

**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 54)

### 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.

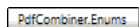
### Classes

	Name	Description
	Enums ( <a href="#">see page 4</a> )	This class contains enums for detailed information In usage
	ExceptionExtensions ( <a href="#">see page 5</a> )	This class is used to take generic exception messages With the inner exception, stack trace and source informations
	FileInfo ( <a href="#">see page 6</a> )	This class is used to have File's Full Path and Name
	FrmMain ( <a href="#">see page 6</a> )	This is class PdfCombiner.FrmMain.
	Program ( <a href="#">see page 32</a> )	This is class PdfCombiner.Program.

## 1.1.2.1 Enums Class

This class contains enums for detailed information In usage

### Class Hierarchy




#### C#

```
public static class Enums;
```

#### File

Enums.cs ([see page 34](#))

### Enums Enumerations

	Name	Description
	OrderType ( <a href="#">see page 5</a> )	This enum is used for Assigning the order type in listbox items Ascending or Descending

## 1.1.2.1.1 Enums Enumerations

### 1.1.2.1.1.1 PdfCombiner.Enums.OrderType Enumeration

This enum is used for Assigning the order type in listbox items Ascending or Descending

#### C#

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

#### File

Enums.cs (🔗 see page 34)

## 1.1.2.2 ExceptionExtensions Class

This class is used to take generic exception messages With the inner exception, stack trace and source informations

#### Class Hierarchy

PdfCombiner.ExceptionExtensions

#### C#

```
public static class ExceptionExtensions;
```

#### File

ExceptionExtensions.cs (🔗 see page 35)

#### ExceptionExtensions Methods

	Name	Description
🔗	GetAllMessages (🔗 see page 5)	This method is used to take all messages with stack trace info in exception

### 1.1.2.2.1 ExceptionExtensions Methods

#### 1.1.2.2.1.1 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: {
```

```

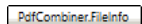
3:     var sb = new StringBuilder();
4:     sb.AppendLine(exception.Message);
5:     if (exception.InnerException != null)
6:         sb.AppendLine($" {exception.InnerException.GetAllMessages()}");
7:     if (!string.IsNullOrEmpty(exception.StackTrace))
8:         sb.AppendLine($" {exception.StackTrace}");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

```

### 1.1.2.3 FileInfo Class

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

#### Class Hierarchy







#### C#

```
public class FileInfo;
```

#### File

FileInfo.cs ( see page 35)

#### FileInfo Properties

	Name	Description
	FileName (  see page 6)	File Name Property
	FilePath (  see page 6)	File Path Property

### 1.1.2.3.1 FileInfo Properties

#### 1.1.2.3.1.1 FileInfo.FileName Property

File Name Property

#### C#

```
public string FileName;
```

#### 1.1.2.3.1.2 FileInfo.FilePath Property

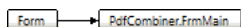
File Path Property

#### C#

```
public string FilePath;
```

### 1.1.2.4 FrmMain Class


#### Class Hierarchy



#### C#

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

#### File

FrmMain.Designer.cs ( see page 48)





















**Description**

This is class PdfCombiner.FrmMain.







**Methods**































	Name	Description
	FrmMain ( <a href="#">see page 9</a> )	This is FrmMain, a member of class FrmMain.











**FrmMain Fields**

	Name	Description
	_resource ( <a href="#">see page 9</a> )	This is _resource, a member of class FrmMain.
	btnAddFile ( <a href="#">see page 9</a> )	This is btnAddFile, a member of class FrmMain.
	btnAddFolder ( <a href="#">see page 10</a> )	This is btnAddFolder, a member of class FrmMain.
	btnClearList ( <a href="#">see page 10</a> )	This is btnClearList, a member of class FrmMain.
	btnCombineITextSharp ( <a href="#">see page 10</a> )	This is btnCombineITextSharp, a member of class FrmMain.
	btnCombinePdfSharp ( <a href="#">see page 10</a> )	This is btnCombinePdfSharp, a member of class FrmMain.
	cmbLanguage ( <a href="#">see page 10</a> )	This is cmbLanguage, a member of class FrmMain.
	components ( <a href="#">see page 10</a> )	Required designer variable.
	LanguageList ( <a href="#">see page 10</a> )	Culture lists whose resources have been added to project
	lbFiles ( <a href="#">see page 11</a> )	This is lbFiles, a member of class FrmMain.
	lblDetails ( <a href="#">see page 11</a> )	This is lblDetails, a member of class FrmMain.
	menuItemDelete ( <a href="#">see page 11</a> )	This is menuItemDelete, a member of class FrmMain.
	menuItemOrderByNameAscending ( <a href="#">see page 11</a> )	This is menuItemOrderByNameAscending, a member of class FrmMain.
	menuItemOrderByNameDescending ( <a href="#">see page 11</a> )	This is menuItemOrderByNameDescending, a member of class FrmMain.
	menuItemOrderByPathAscending ( <a href="#">see page 11</a> )	This is menuItemOrderByPathAscending, a member of class FrmMain.
	menuItemOrderByPathDescending ( <a href="#">see page 11</a> )	This is menuItemOrderByPathDescending, a member of class FrmMain.
	menuStrip ( <a href="#">see page 12</a> )	This is menuStrip, a member of class FrmMain.
	pbFiles ( <a href="#">see page 12</a> )	This is pbFiles, a member of class FrmMain.

**FrmMain Methods**

	Name	Description
	AddPdfContentToTargetPdfFileITextSharp ( <a href="#">see page 12</a> )	This method is used to take source pdf file content And add to the target combined pdf file
	AddPdfContentToTargetPdfFilePdfSharp ( <a href="#">see page 12</a> )	This method is used to take source pdf file content And add to the target combined pdf file
	AddPdfFilesToList ( <a href="#">see page 13</a> )	This method is used both adding files or folders to list
	btnAddFile_Click ( <a href="#">see page 13</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 14</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 14</a> )	This function is used to clear the file list in listbox

	<code>btnCombineITextSharp_Click</code> ( <a href="#">see page 15</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
	<code>btnCombinePdfSharp_Click</code> ( <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 PdfSharp Nuget Package
	<code>cmbLanguage_SelectedIndexChanged</code> ( <a href="#">see page 17</a> )	This function is used to set the <code>_resource</code> ( <a href="#">see page 9</a> ) file And with this, you can see form elements, info messages etc. with the selected <code>_resource</code> ( <a href="#">see page 9</a> ) file contents. That is used to multi language support
	<code>DeleteFilesFromListBox</code> ( <a href="#">see page 17</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
	<code>Dispose</code> ( <a href="#">see page 18</a> )	Clean up any resources being used.
	<code>FillLanguageComboboxAndSetFirstLanguage</code> ( <a href="#">see page 18</a> )	This method is used to Fill the Language combobox And set initial value with The language computer used in
	<code>FormMembersNameInitialization</code> ( <a href="#">see page 18</a> )	This method is used to change text of form members by the Resource file which is selected
	<code>FrmMain_FormClosed</code> ( <a href="#">see page 19</a> )	This function is used to terminate application
	<code>FrmMain_Shown</code> ( <a href="#">see page 19</a> )	This function is used to assign menu to listbox When form is shown
	<code>GenerateAddFileMessage</code> ( <a href="#">see page 20</a> )	This method is used to show info message after adding files or folders
	<code>GenerateCombineFileMessage</code> ( <a href="#">see page 20</a> )	This method is used to set successfull combining file message And set value of Form Caption etc.
 	<code>GenerateCombineWarningMessage</code> ( <a href="#">see page 20</a> )	This method is used When user just want to combine 1 pdf file
 	<code>GenerateDeleteItemMessage</code> ( <a href="#">see page 21</a> )	This method is used after deleting items from listbox
	<code>GenerateExceptionMessage</code> ( <a href="#">see page 21</a> )	This method is used to Show error message when exception occurs and set form title initial value
 	<code>GenerateNoFileInListBoxMessage</code> ( <a href="#">see page 21</a> )	This method is used to generate message When there is no item in listbox
 	<code>GenerateNoSelectedFileInListBoxMessage</code> ( <a href="#">see page 22</a> )	This method is used in When there is no selected file in listbox
	<code>InitializeComponent</code> ( <a href="#">see page 22</a> )	Required method for Designer support - do not modify the contents of this method with the code editor.
 	<code>InitializeFileDialog</code> ( <a href="#">see page 25</a> )	This function is used to set options of file dialog like Filter, start path etc.
	<code>lbFiles_DragDrop</code> ( <a href="#">see page 26</a> )	This is used to drag and drop an item in listbox Which is used to order books and combine them with that order
	<code>lbFiles_DragOver</code> ( <a href="#">see page 26</a> )	This is used to give effect while dragging and dropping an item in listbox
	<code>lbFiles_KeyDown</code> ( <a href="#">see page 26</a> )	This function is used to operate some process in files in listbox which are chosen
	<code>lbFiles_MouseDown</code> ( <a href="#">see page 27</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 27</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 28</a> )	This method is used to order elements in listbox ascending by file name
	<code>menuItemOrderByNameDescending_Click</code> ( <a href="#">see page 28</a> )	This method is used to order elements in listbox descending by file name

	menuItemOrderByPathAscending_Click (see page 28)	This method is used to order elements in listbox ascending by full path
	menuItemOrderByPathDescending_Click (see page 29)	This method is used to order elements in listbox descending by full path
	SaveOutputPdfFile (see page 29)	This method is used to Save Combined PDF File
	SetCombinationRatio (see page 29)	This method is used to set combination degree while combining PDF files And show this ratio on Form Caption
	SetInitialValuesForCombiningFiles (see page 30)	This method is used to set initial values of some properties Which are used in combining PDF files In both PdfSharp and ITextSharp
	SetOutputFilePropertiesPdfSharp (see page 30)	This method is used to set combined PDF file Properties (see page 1) like Compression etc.
	SetProgramLanguage (see page 30)	This method is used to Set Program (see page 32) Language By the selection from Language Combobox Finally, with the selected language Form elements renamed with the selected language
	ShowDeleteDialog (see page 31)	This method is used to Show delete file dialog
	SortItemsByName (see page 31)	This method is used to order items by file name in listbox And return them with the given order as parametre
	SortItemsByPath (see page 32)	This method is used to order items by full path in listbox And return them with the given order as parametre

### 1.1.2.4.1 FrmMain.FrmMain Constructor

C#

```
public FrmMain();
```

Description

This is FrmMain, a member of class FrmMain.

Body Source

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

### 1.1.2.4.2 FrmMain Fields

#### 1.1.2.4.2.1 FrmMain.\_resource Field

C#

```
private static ResourceManager _resource;
```

Description

This is \_resource, a member of class FrmMain.

#### 1.1.2.4.2.2 FrmMain.btnAddFile Field

C#

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

Description

This is btnAddFile, a member of class FrmMain.

#### 1.1.2.4.2.3 FrmMain.btnAddFolder Field

**C#**

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

**Description**

This is btnAddFolder, a member of class FrmMain.

#### 1.1.2.4.2.4 FrmMain.btnClearList Field

**C#**

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

**Description**

This is btnClearList, a member of class FrmMain.

#### 1.1.2.4.2.5 FrmMain.btnCombineITextSharp Field

**C#**

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

**Description**

This is btnCombineITextSharp, a member of class FrmMain.

#### 1.1.2.4.2.6 FrmMain.btnCombinePdfSharp Field

**C#**

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

**Description**

This is btnCombinePdfSharp, a member of class FrmMain.

#### 1.1.2.4.2.7 FrmMain.cmbLanguage Field

**C#**

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

**Description**

This is cmbLanguage, a member of class FrmMain.

#### 1.1.2.4.2.8 FrmMain.components Field

Required designer variable.

**C#**

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

#### 1.1.2.4.2.9 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

#### 1.1.2.4.2.10 FrmMain.lbFiles Field

C#

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

##### Description

This is lbFiles, a member of class FrmMain.

#### 1.1.2.4.2.11 FrmMain.lblDetails Field

C#

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

##### Description

This is lblDetails, a member of class FrmMain.

#### 1.1.2.4.2.12 FrmMain.menuItemDelete Field

C#

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

##### Description

This is menuItemDelete, a member of class FrmMain.

#### 1.1.2.4.2.13 FrmMain.menuItemOrderByNameAscending Field

C#

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

##### Description

This is menuItemOrderByNameAscending, a member of class FrmMain.

#### 1.1.2.4.2.14 FrmMain.menuItemOrderByNameDescending Field

C#

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

##### Description

This is menuItemOrderByNameDescending, a member of class FrmMain.

#### 1.1.2.4.2.15 FrmMain.menuItemOrderByPathAscending Field

C#

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

##### Description

This is menuItemOrderByPathAscending, a member of class FrmMain.

#### 1.1.2.4.2.16 FrmMain.menuItemOrderByPathDescending Field

C#

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

**Description**

This is menuItemOrderByPathDescending, a member of class FrmMain.

**1.1.2.4.2.17 FrmMain.menuStrip Field****C#**

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

**Description**

This is menuStrip, a member of class FrmMain.

**1.1.2.4.2.18 FrmMain.pbFiles Field****C#**

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

**Description**

This is pbFiles, a member of class FrmMain.

**1.1.2.4.3 FrmMain Methods****1.1.2.4.3.1 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());
6:     pdfReader.ConsolidateNamedDestinations();
7:     for (var j = 1; j <= pdfReader.NumberOfPages; j++)
8:     {
9:         var page = pdfWriter.GetImportedPage(pdfReader, j);
10:        pdfWriter.AddPage(page);
11:    }
12:    pdfReader.Close();
13:    combinedFiles++;
14: }
```

**1.1.2.4.3.2 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 index = 0; index < count; index++)
7:     {
8:         var page = inputDocument.Pages[index];
9:         outputFile.AddPage(page);
10:    }
11:
12:    inputDocument.Close();
13:    combinedFiles++;
14: }
```

**1.1.2.4.3.3 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: }
```

**1.1.2.4.3.4 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.FileName == null) return;
10:
11:         AddPdfFilesToList(dialogAddFile.FileName, ref addedFileCount);
12:         GenerateAddFileMessage(addedFileCount);
13:     }
14: }
```

**1.1.2.4.3.5 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: }
```

**1.1.2.4.3.6 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: }

```

**1.1.2.4.3.7 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();
24:                        outputFile.Open();
25:                        foreach (var t in lbFiles.Items)
26:                        {
27:                            try
28:                            {
29:                                AddPdfContentToTargetPdfFileITextSharp(t, pdfWriter,
30:                                ref combinedFiles);
31:                            }
32:                            catch (Exception)
33:                            {
34:                                //
35:                            }
36:                        }
37:                    }
38:                }
39:            }
40:        }
41:    }
42:    catch (Exception)
43:    {
44:        //
45:    }
46: }

```

```

32:                fileCount--;
33:            }
34:
35:            SetCombinationRatio(combinedFiles, fileCount);
36:        }
37:
38:        pdfWriter.Close();
39:    }
40:
41:    outputFile.Close();
42:    }
43:
44:    GenerateCombineFileMessage(fileCount, outputFileName);
45:    }
46:    }
47:    }
48:    catch (Exception ex)
49:    {
50:        GenerateExceptionMessage(ex);
51:    }
52: }

```

#### 1.1.2.4.3.8 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 ||
13:                    string.IsNullOrEmpty(dialogExport.SelectedPath)) return;
14:                SetInitialValuesForCombiningFiles(dialogExport, out var outputFileName,
15:                out var fileCount,
16:                out var combinedFiles);
17:                using (var outputFile = new PdfDocument())
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: }

```

#### 1.1.2.4.3.9 FrmMain.cmbLanguage\_SelectedIndexChanged Method

This function is used to set the \_resource (see page 9) file And with this, you can see form elements, info messages etc. with the selected \_resource (see page 9) 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: }

```

#### 1.1.2.4.3.10 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
5:         t.ToString()).ToList();
6:     var allItems = (from object item in lbFiles.Items where removedItems.All(j => j !=
7:         item.ToString()) select item.ToString()).ToList();
8:     lbFiles.Items.Clear();
9:     foreach (var item in allItems)
10:     {
11:         lbFiles.Items.Add(item);
12:     }

```

```

13:
14:     GenerateDeleteItemMessage(fileCount);
15: }

```

#### 1.1.2.4.3.11 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: }

```

#### 1.1.2.4.3.12 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: }

```

#### 1.1.2.4.3.13 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: }

```

**1.1.2.4.3.14 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, MessageBoxIcon.Information);
4:     Application.Exit();
5: }

```

**1.1.2.4.3.15 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: }

```

#### 1.1.2.4.3.16 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: }

```

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

```

#### 1.1.2.4.3.18 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"), MessageBoxButtons.OK, MessageBoxIcon.Error);
4: }

```

**1.1.2.4.3.19 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"), MessageBoxButtons.OK, MessageBoxIcon.Information);
4: }

```

**1.1.2.4.3.20 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(),
    _resource.GetString("AppTitle"), MessageBoxButtons.OK, MessageBoxIcon.Error);
4:     pbFiles.Value = pbFiles.Minimum;
5:     if (ActiveForm != null) ActiveForm.Text = _resource.GetString("AppTitle");
6: }

```

**1.1.2.4.3.21 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"),
    MessageBoxButtons.OK, MessageBoxIcon.Error);
4: }

```

### 1.1.2.4.3.22 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: }
```

### 1.1.2.4.3.23 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
System.ComponentModel.ComponentResourceManager(typeof(FrmMain));
5:     this.btnCombinePdfSharp = new System.Windows.Forms.Button();
6:     this.btnAddFolder = new System.Windows.Forms.Button();
7:     this.btnAddFile = new System.Windows.Forms.Button();
8:     this.btnClearList = new System.Windows.Forms.Button();
9:     this.lbFiles = new System.Windows.Forms.ListBox();
10:    this.lblDetails = new System.Windows.Forms.Label();
11:    this.pbFiles = new System.Windows.Forms.ProgressBar();
12:    this.btnCombineITextSharp = new System.Windows.Forms.Button();
13:    this.menuStrip = new System.Windows.Forms.ContextMenuStrip(this.components);
14:    this.menuItemDelete = new System.Windows.Forms.ToolStripMenuItem();
15:    this.menuItemOrderByPathAscending = new System.Windows.Forms.ToolStripMenuItem();
16:    this.menuItemOrderByPathDescending = new System.Windows.Forms.ToolStripMenuItem();
17:    this.menuItemOrderByNameAscending = new System.Windows.Forms.ToolStripMenuItem();
18:    this.menuItemOrderByNameDescending = new System.Windows.Forms.ToolStripMenuItem();
19:    this.cmbLanguage = new System.Windows.Forms.ComboBox();
20:    this.menuStrip.SuspendLayout();
21:    this.SuspendLayout();
22:    //
23:    // btnCombinePdfSharp
24:    //
25:    this.btnCombinePdfSharp.Image = ((System.Drawing.Image)
(resources.GetObject("btnCombinePdfSharp.Image")));
26:    this.btnCombinePdfSharp.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
27:    this.btnCombinePdfSharp.Location = new System.Drawing.Point(442, 5);
28:    this.btnCombinePdfSharp.Name = "btnCombinePdfSharp";
29:    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.AutoScaleMode.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: }

```

### 1.1.2.4.3.24 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: }
```

#### 1.1.2.4.3.25 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: }
```

#### 1.1.2.4.3.26 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: }
```

#### 1.1.2.4.3.27 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
combination
7:     if (!e.Control || e.KeyCode != Keys.A) return;
8:     for (var i = 0; i < lbFiles.Items.Count; i++)
9:     {
10:         lbFiles.SetSelected(i, true);
11:     }
12: }

```

**1.1.2.4.3.28 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: }

```

**1.1.2.4.3.29 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:     }
```

#### 1.1.2.4.3.30 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: }
```

#### 1.1.2.4.3.31 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: }
```

#### 1.1.2.4.3.32 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: }
```

### 1.1.2.4.3.33 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: }
```

### 1.1.2.4.3.34 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: }
```

### 1.1.2.4.3.35 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: }

```

#### 1.1.2.4.3.36 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: }

```

#### 1.1.2.4.3.37 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: }

```

#### 1.1.2.4.3.38 FrmMain.SetProgramLanguage Method

This method is used to Set Program (see page 32) 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: }

```

**1.1.2.4.3.39 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( _resource.GetString("DeleteWarning1") +
4:     lbFiles.SelectedItems.Count + _resource.GetString("DeleteWarning2"),
5:     _resource.GetString("AppTitle"), MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
6: }

```

**1.1.2.4.3.40 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 =
6:     fullPath.Split(c).ToList() where list.Count > 0 select new FileInfo {FilePath = fullPath,
7:     FileName = list[list.Count - 1]}).ToList();
8:     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: }

```

#### 1.1.2.4.3.41 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: }

```

### 1.1.2.5 Program Class

**Class Hierarchy**

PdfCombiner.Program

**C#**

```
internal static class Program;
```

**File**

Program.cs (🔗 see page 52)

**Description**

This is class PdfCombiner.Program.

**Program Methods**

	Name	Description
🔗	Main (🔗 see page 32)	The main entry point for the application.

#### 1.1.2.5.1 Program Methods

##### 1.1.2.5.1.1 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 33</a> )	This is file AssemblyInfo.cs.
Enums.cs ( <a href="#">see page 34</a> )	This is file Enums.cs.
ExceptionExtensions.cs ( <a href="#">see page 35</a> )	This is file ExceptionExtensions.cs.
FileInfo.cs ( <a href="#">see page 35</a> )	This is file FileInfo.cs.
FrmMain.cs ( <a href="#">see page 36</a> )	This is file FrmMain.cs.
FrmMain.Designer.cs ( <a href="#">see page 48</a> )	This is file FrmMain.Designer.cs.
PdfCombiner.csproj ( <a href="#">see page 52</a> )	This is file PdfCombiner.csproj.
PdfCombiner.sln ( <a href="#">see page 52</a> )	This is file PdfCombiner.sln.
Program.cs ( <a href="#">see page 52</a> )	This is file Program.cs.
Resources.Designer.cs ( <a href="#">see page 53</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 54</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 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:         /// <summary>
12:         /// This enum is used for
13:         /// Assigning the order type in listbox items
14:         /// Ascending or Descending
15:         /// </summary>
16:         [Flags]
17:         public enum OrderType
18:         {
19:             Ascending,
20:             Descending
21:         }
22:     }
23: }
```

### Namespaces

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

## 1.2.3 ExceptionExtensions.cs

This is file ExceptionExtensions.cs.

### Body Source

```
1: ?using System;
2: using System.Text;
3:
4: namespace PdfCombiner
5: {
6:     /// <summary>
7:     /// This class is used to take generic exception messages
8:     /// With the inner exception, stack trace and source informations
9:     /// </summary>
10:    public static class ExceptionExtensions
11:    {
12:        /// <summary>
13:        /// This method is used to take all messages with stack trace info in exception
14:        /// </summary>
15:        /// <param name="exception">Exception</param>
16:        /// <returns>Detailed Error Message</returns>
17:        public static string GetAllMessages(this Exception exception)
18:        {
19:            var sb = new StringBuilder();
20:            sb.AppendLine(exception.Message);
21:            if (exception.InnerException != null)
22:                sb.AppendLine($" {exception.InnerException.GetAllMessages()}");
23:            if (!string.IsNullOrEmpty(exception.StackTrace))
24:                sb.AppendLine($" {exception.StackTrace}");
25:            if (!string.IsNullOrEmpty(exception.Source))
26:                sb.AppendLine($" {exception.Source}");
27:            return sb.ToString();
28:        }
29:    }
30: }
```

### Namespaces

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

## 1.2.4 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.

# 1.2.5 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:
34:         #region Form Init, Language Settings and Finalize Methods
35:
36:         /// <summary>
37:         /// This function is used to terminate application
38:         /// </summary>
39:         /// <param name="sender">The sender info (For example Main Form)</param>
40:         /// <param name="e">Event Arguments</param>
41:         private void FrmMain_FormClosed(object sender, FormClosedEventArgs e)
42:         {
43:             MessageBox.Show(_resource.GetString("ThanksMessage"),
44: _resource.GetString("AppTitle"), MessageBoxButtons.OK, MessageBoxIcon.Information);
45:             Application.Exit();
46:         }
47:
48:         /// <summary>
49:         /// This function is used to assign menu to listbox
```

```

48:         /// When form is shown
49:         /// </summary>
50:         /// <param name="sender">The sender info (For example Main Form)</param>
51:         /// <param name="e">Event Arguments</param>
52:         private void FrmMain_Shown(object sender, EventArgs e)
53:         {
54:             lbFiles.ContextMenuStrip = menuStrip;
55:
56:             FillLanguageComboboxAndSetFirstLanguage(out var firstLanguage);
57:
58:             _resource = new ResourceManager("PdfCombiner.Resources.AppResources-" +
firstLanguage,
59:             Assembly.GetExecutingAssembly());
60:             cmbLanguage.SelectedIndex =
cmbLanguage.FindStringExact(firstLanguage.ToUpper(new CultureInfo("en-US")));
61:
62:             FormMembersNameInitialization();
63:         }
64:
65:         /// <summary>
66:         /// This method is used to Fill the Language combobox
67:         /// And set initial value with
68:         /// The language computer used in
69:         /// </summary>
70:         /// <param name="firstLanguage">First Language Info</param>
71:         private void FillLanguageComboboxAndSetFirstLanguage(out string firstLanguage)
72:         {
73:             cmbLanguage.DropDownStyle = ComboBoxStyle.DropDownList;
74:             cmbLanguage.Items.Clear();
75:             var cultureInfo = CultureInfo.InstalledUICulture;
76:             firstLanguage = string.Empty;
77:             foreach (var item in LanguageList)
78:             {
79:                 if (cultureInfo.Name.Contains(item))
80:                     firstLanguage = item.ToLower(new CultureInfo("en-US"));
81:                 cmbLanguage.Items.Add(item);
82:             }
83:
84:             if (string.IsNullOrEmpty(firstLanguage))
85:                 firstLanguage = "tr";
86:         }
87:
88:         /// <summary>
89:         /// This function is used to set the _resource file
90:         /// And with this, you can see form elements, info messages etc. with the
selected
91:         /// _resource file contents. That is used to multi language support
92:         /// </summary>
93:         /// <param name="sender">The sender info (For example Main Form)</param>
94:         /// <param name="e">Event Arguments</param>
95:         private void cmbLanguage_SelectedIndexChanged(object sender, EventArgs e)
96:         {
97:             if (cmbLanguage.SelectedIndex == -1) return;
98:             SetProgramLanguage();
99:         }
100:
101:         /// <summary>
102:         /// This method is used to Set Program Language
103:         /// By the selection from Language Combobox
104:         /// Finally, with the selected language
105:         /// Form elements renamed with the selected language
106:         /// </summary>
107:         private void SetProgramLanguage()
108:         {
109:             try
110:             {
111:                 _resource = new ResourceManager(
112:                     "PdfCombiner.Resources.AppResources-" +
113:                     cmbLanguage.SelectedItem.ToString().ToLower(new

```

```

CultureInfo("en-US")),
114:         Assembly.GetExecutingAssembly());
115:     }
116:     catch (Exception)
117:     {
118:         _resource = new
ResourceManager("PdfCombiner.Resources.AppResources-en",
119:             Assembly.GetExecutingAssembly());
120:     }
121:     finally
122:     {
123:         FormMembersNameInitialization();
124:     }
125: }
126:
127: /// <summary>
128: /// This method is used to change text of form members by the
129: /// Resource file which is selected
130: /// </summary>
131: private void FormMembersNameInitialization()
132: {
133:     if (_resource == null) return;
134:     if (ActiveForm != null) ActiveForm.Text = _resource.GetString("AppTitle")
?? string.Empty;
135:     btnAddFile.Text = _resource.GetString("AddFile") ?? string.Empty;
136:     btnAddFolder.Text = _resource.GetString("AddFolder") ?? string.Empty;
137:     btnClearList.Text = _resource.GetString("ClearFileList") ?? string.Empty;
138:     btnCombineITextSharp.Text = _resource.GetString("CombineFilesITextSharp")
?? string.Empty;
139:     btnCombinePdfSharp.Text = _resource.GetString("CombineFilesPdfSharp") ??
string.Empty;
140:     menuItemDelete.Text = _resource.GetString("Delete") ?? string.Empty;
141:     menuItemOrderByPathAscending.Text =
_resource.GetString("OrderByPathAscending") ?? string.Empty;
142:     menuItemOrderByPathDescending.Text =
_resource.GetString("OrderByPathDescending") ?? string.Empty;
143:     menuItemOrderByNameAscending.Text =
_resource.GetString("OrderByNameAscending") ?? string.Empty;
144:     menuItemOrderByNameDescending.Text =
_resource.GetString("OrderByNameDescending") ?? string.Empty;
145:     cmbLanguage.Text = _resource.GetString("SelectLanguage") ?? string.Empty;
146: }
147:
148: #endregion
149:
150: #region Add Item Methods
151:
152: /// <summary>
153: /// It is used to add single or multiple PDF files to combine
154: /// When you choose file or files
155: /// The listbox in the form will be filled with the name of the files
156: /// Which you choose to combine
157: /// </summary>
158: /// <param name="sender">The sender info (For example Main Form)</param>
159: /// <param name="e">Event Arguments</param>
160: private void btnAddFile_Click(object sender, EventArgs e)
161: {
162:     using (var dialogAddFile = new OpenFileDialog())
163:     {
164:         InitializeFileDialog(dialogAddFile);
165:
166:         var result = dialogAddFile.ShowDialog();
167:         var addedFileCount = 0;
168:         if (result != DialogResult.OK || dialogAddFile.FileNames == null)
return;
169:
170:         AddPdfFilesToList(dialogAddFile.FileNames, ref addedFileCount);
171:         GenerateAddFileMessage(addedFileCount);
172:     }

```



```

173:     }
174:
175:     /// <summary>
176:     /// This function is used to add PDF files in a folder which you choose in
177:     /// Folder Dialog recursively
178:     /// When you use it the added files will be seen in listbox
179:     /// </summary>
180:     /// <param name="sender">The sender info (For example Main Form)</param>
181:     /// <param name="e">Event Arguments</param>
182:     private void btnAddFolder_Click(object sender, EventArgs e)
183:     {
184:         using (var dialogAddFolder = new FolderBrowserDialog())
185:         {
186:             var result = dialogAddFolder.ShowDialog();
187:             if (result != DialogResult.OK ||
string.IsNullOrEmpty(dialogAddFolder.SelectedPath)) return;
188:             var fileNames = Directory.GetFiles(dialogAddFolder.SelectedPath,
"*.pdf",
189:             SearchOption.AllDirectories);
190:             var addedFileCount = 0;
191:
192:             AddPdfFilesToList(fileNames, ref addedFileCount);
193:             GenerateAddFileMessage(addedFileCount);
194:         }
195:     }
196:
197:     /// <summary>
198:     /// This function is used to set options of file dialog like
199:     /// Filter, start path etc.
200:     /// </summary>
201:     /// <param name="dialogAddFile">File Dialog</param>
202:     private static void InitializeFileDialog(OpenFileDialog dialogAddFile)
203:     {
204:         dialogAddFile.Multiselect = true;
205:         dialogAddFile.Filter = _resource.GetString("PdfFiles") + @" (*.pdf)|*.pdf";
206:         dialogAddFile.InitialDirectory = Application.StartupPath;
207:         dialogAddFile.Title = _resource.GetString("SelectPdfFile");
208:         dialogAddFile.DefaultExt = "PDF";
209:     }
210:
211:     /// <summary>
212:     /// This method is used both adding files or folders to list
213:     /// </summary>
214:     /// <param name="fileNames">List Of File Names</param>
215:     /// <param name="addedFileCount">Added File Count For Progress Bar</param>
216:     private void AddPdfFilesToList(ICollection<string> fileNames, ref int
addedFileCount)
217:     {
218:         foreach (var file in fileNames)
219:         {
220:             if (lbFiles.Items.Contains(file)) continue;
221:             lbFiles.Items.Add(file);
222:             addedFileCount++;
223:
224:             pbFiles.Value = addedFileCount * 100 / fileNames.Count;
225:             if (pbFiles.Value > pbFiles.Maximum)
226:                 pbFiles.Value = pbFiles.Maximum;
227:             if (ActiveForm != null) ActiveForm.Text = @"%" + pbFiles.Value;
228:         }
229:     }
230:
231:     /// <summary>
232:     /// This method is used to show info message after adding files or folders
233:     /// </summary>
234:     /// <param name="addedFileCount">Added File Count</param>
235:     private void GenerateAddFileMessage(int addedFileCount)
236:     {
237:         MessageBox.Show(addedFileCount + _resource.GetString("FileAddMessage"),
238:             _resource.GetString("AppTitle"), MessageBoxButtons.OK,

```

```

MessageBoxIcon.Information);
239:
240:         pbFiles.Value = pbFiles.Minimum;
241:         if (ActiveForm != null) ActiveForm.Text = _resource.GetString("AppTitle");
242:     }
243:
244:     #endregion
245:
246:     #region Combine PDF Methods
247:
248:     /// <summary>
249:     /// This function is used to take list of PDF files in listbox,
250:     /// Combine them with a single file in location which you choose in folder
dialog
251:     /// And you can see the progress in progress bar
252:     /// It uses PdfSharp Nuget Package
253:     /// </summary>
254:     /// <param name="sender">The sender info (For example Main Form)</param>
255:     /// <param name="e">Event Arguments</param>
256:     private void btnCombinePdfSharp_Click(object sender, EventArgs e)
257:     {
258:         try
259:         {
260:             if (lbFiles.Items.Count < 2)
261:                 GenerateCombineWarningMessage();
262:             else
263:             {
264:                 using (var dialogExport = new FolderBrowserDialog())
265:                 {
266:                     var result = dialogExport.ShowDialog();
267:                     if (result != DialogResult.OK ||
string.IsNullOrEmpty(dialogExport.SelectedPath)) return;
268:
269:                     SetInitialValuesForCombiningFiles(dialogExport, out var
outputFileName, out var fileCount,
270:                                                         out var combinedFiles);
271:
272:                     using (var outputFile = new PdfDocument())
273:                     {
274:                         SetOutputFilePropertiesPdfSharp(outputFile);
275:
276:                         foreach (var t in lbFiles.Items)
277:                         {
278:                             try
279:                             {
280:                                 AddPdfContentToTargetPdfFilePdfSharp(t,
outputFile, ref combinedFiles);
281:                             }
282:                             catch (Exception)
283:                             {
284:                                 fileCount--;
285:                             }
286:
287:                             SetCombinationRatio(combinedFiles, fileCount);
288:                         }
289:
290:                         SaveOutputPdfFile(outputFile, outputFileName);
291:                     }
292:
293:                     GenerateCombineFileMessage(fileCount, outputFileName);
294:                 }
295:             }
296:         }
297:         catch (Exception ex)
298:         {
299:             GenerateExceptionMessage(ex);
300:         }
301:     }
302:

```

```

303:         /// <summary>
304:         /// This function is used to take list of PDF files in listbox,
305:         /// Combine them with a single file in location which you choose in folder
dialog
306:         /// And you can see the progress in progress bar
307:         /// It uses iTextSharp Nuget Package
308:         /// </summary>
309:         /// <param name="sender">The sender info (For example Main Form)</param>
310:         /// <param name="e">Event Arguments</param>
311:         private void btnCombineITextSharp_Click(object sender, EventArgs e)
312:         {
313:             try
314:             {
315:                 if (lbFiles.Items.Count < 2)
316:                     GenerateCombineWarningMessage();
317:                 else
318:                 {
319:                     using (var dialogExport = new FolderBrowserDialog())
320:                     {
321:                         var result = dialogExport.ShowDialog();
322:                         if (result != DialogResult.OK ||
string.IsNullOrEmpty(dialogExport.SelectedPath)) return;
323:
324:                         SetInitialValuesForCombiningFiles(dialogExport, out var
outputFileName, out var fileCount,
325:                             out var combinedFiles);
326:
327:                         var outputFile = new Document();
328:                         using (var outputFileStream = new FileStream(outputFileName,
FileMode.Create))
329:                         {
330:                             using (var pdfWriter = new PdfCopy(outputFile,
outputFileStream))
331:                             {
332:                                 pdfWriter.SetFullCompression();
333:                                 outputFile.Open();
334:                                 foreach (var t in lbFiles.Items)
335:                                 {
336:                                     try
337:                                     {
338:                                         AddPdfContentToTargetPdfFileITextSharp(t,
pdfWriter, ref combinedFiles);
339:                                     }
340:                                     catch (Exception)
341:                                     {
342:                                         fileCount--;
343:                                     }
344:
345:                                     SetCombinationRatio(combinedFiles, fileCount);
346:                                 }
347:
348:                                 pdfWriter.Close();
349:                             }
350:
351:                             outputFile.Close();
352:                         }
353:
354:                         GenerateCombineFileMessage(fileCount, outputFileName);
355:                     }
356:                 }
357:             }
358:             catch (Exception ex)
359:             {
360:                 GenerateExceptionMessage(ex);
361:             }
362:         }
363:
364:         /// <summary>
365:         /// This method is used to

```

```

366:         /// Save Combined PDF File
367:         /// </summary>
368:         /// <param name="outputFile">Combined PDF File</param>
369:         /// <param name="outputFileName">Combined PDF File Path</param>
370:         private static void SaveOutputPdfFile(PdfDocument outputFile, string
outputFileName)
371:         {
372:             outputFile.Save(outputFileName);
373:             outputFile.Close();
374:         }
375:
376:         /// <summary>
377:         /// This method is used to take source pdf file content
378:         /// And add to the target combined pdf file
379:         /// </summary>
380:         /// <param name="t">Source PDF File Path</param>
381:         /// <param name="outputFile">Combined PDF File</param>
382:         /// <param name="combinedFiles">Combined File Count</param>
383:         private static void AddPdfContentToTargetPdfFilePdfSharp(object t, PdfDocument
outputFile, ref int combinedFiles)
384:         {
385:             var inputDocument = PdfReader.Open(t.ToString(),
PdfDocumentOpenMode.Import);
386:             var count = inputDocument.PageCount;
387:             for (var index = 0; index < count; index++)
388:             {
389:                 var page = inputDocument.Pages[index];
390:                 outputFile.AddPage(page);
391:             }
392:
393:             inputDocument.Close();
394:             combinedFiles++;
395:         }
396:
397:         /// <summary>
398:         /// This method is used to set combined PDF file
399:         /// Properties like Compression etc.
400:         /// </summary>
401:         /// <param name="outputFile">Combined PDF File</param>
402:         private static void SetOutputFilePropertiesPdfSharp(PdfDocument outputFile)
403:         {
404:             outputFile.Options.CompressContentStreams = true;
405:             outputFile.Options.EnableCcittCompressionForBilevelImages = true;
406:             outputFile.Options.FlateEncodeMode = PdfFlateEncodeMode.BestCompression;
407:         }
408:
409:         /// <summary>
410:         /// This method is used to take source pdf file content
411:         /// And add to the target combined pdf file
412:         /// </summary>
413:         /// <param name="t">Source PDF File Path</param>
414:         /// <param name="pdfWriter">PDF Writer</param>
415:         /// <param name="combinedFiles">Combined File Count</param>
416:         private static void AddPdfContentToTargetPdfFileITextSharp(object t, PdfCopy
pdfWriter, ref int combinedFiles)
417:         {
418:             var pdfReader =
419:                 new iTextSharp.text.pdf.PdfReader(t.ToString());
420:             pdfReader.ConsolidateNamedDestinations();
421:             for (var j = 1; j <= pdfReader.NumberOfPages; j++)
422:             {
423:                 var page = pdfWriter.GetImportedPage(pdfReader, j);
424:                 pdfWriter.AddPage(page);
425:             }
426:
427:             pdfReader.Close();
428:             combinedFiles++;
429:         }
430:
431:

```

```

432:         /// <summary>
433:         /// This method is used
434:         /// When user just want to combine 1 pdf file
435:         /// </summary>
436:         private static void GenerateCombineWarningMessage()
437:         {
438:             MessageBox.Show(_resource.GetString("CombineWarning"),
439:                 _resource.GetString("AppTitle"), MessageBoxButtons.OK, MessageBoxIcon.Error);
440:         }
441:         /// <summary>
442:         /// This method is used to
443:         /// Show error message when exception occurs and set form title initial value
444:         /// </summary>
445:         /// <param name="ex">Exception information</param>
446:         private void GenerateExceptionMessage(Exception ex)
447:         {
448:             MessageBox.Show(_resource.GetString("CombineErrorMessage") +
449:                 ex.GetAllMessages(), _resource.GetString("AppTitle"), MessageBoxButtons.OK,
450:                 MessageBoxIcon.Error);
451:             pbFiles.Value = pbFiles.Minimum;
452:             if (ActiveForm != null) ActiveForm.Text = _resource.GetString("AppTitle");
453:         }
454:         /// <summary>
455:         /// This method is used to set initial values of some properties
456:         /// Which are used in combining PDF files
457:         /// In both PdfSharp and ITextSharp
458:         /// </summary>
459:         /// <param name="dialogExport">Export Dialog Info</param>
460:         /// <param name="outputFileName">Output File Name</param>
461:         /// <param name="fileCount">Total File Count</param>
462:         /// <param name="combinedFiles">Combined Files Count</param>
463:         private void SetInitialValuesForCombiningFiles(FolderBrowserDialog
464:             dialogExport, out string outputFileName,
465:             out int fileCount, out int combinedFiles)
466:         {
467:             outputFileName = dialogExport.SelectedPath + "/" + Guid.NewGuid() + ".pdf";
468:             fileCount = lbFiles.Items.Count;
469:             combinedFiles = 0;
470:         }
471:         /// <summary>
472:         /// This method is used to set combination degree while combining PDF files
473:         /// And show this ratio on Form Caption
474:         /// </summary>
475:         /// <param name="combinedFiles">Combined File Count</param>
476:         /// <param name="fileCount">Total File Count</param>
477:         private void SetCombinationRatio(int combinedFiles, int fileCount)
478:         {
479:             pbFiles.Value = combinedFiles * 100 / fileCount;
480:             if (pbFiles.Value > pbFiles.Maximum)
481:                 pbFiles.Value = pbFiles.Maximum;
482:             if (ActiveForm != null) ActiveForm.Text = @"%" + pbFiles.Value;
483:         }
484:         /// <summary>
485:         /// This method is used to set successfull combining file message
486:         /// And set value of Form Caption etc.
487:         /// </summary>
488:         /// <param name="fileCount">Total File Count</param>
489:         /// <param name="outputFileName">Output File Name</param>
490:         private void GenerateCombineFileMessage(int fileCount, string outputFileName)
491:         {
492:             MessageBox.Show(fileCount + _resource.GetString("CombineFileMessage") +
493:                 outputFileName, _resource.GetString("AppTitle"), MessageBoxButtons.OK,
494:                 MessageBoxIcon.Information);
495:             pbFiles.Value = pbFiles.Minimum;
496:             if (ActiveForm != null) ActiveForm.Text = _resource.GetString("AppTitle");

```

```

495:         }
496:
497:         #endregion
498:
499:         #region Delete Item Methods
500:
501:         /// <summary>
502:         /// This function is used to operate some process in files in listbox which
are chosen
503:         /// </summary>
504:         /// <param name="sender">The sender info (For example Main Form)</param>
505:         /// <param name="e">Event Arguments</param>
506:         private void lbFiles_KeyDown(object sender, KeyEventArgs e)
507:         {
508:             // this is used to delete selected files in listbox when you press Delete
key
509:             if (e.KeyValue == Keys.Delete.GetHashCode())
510:                 DeleteFilesFromListBox();
511:             // This is used to select all items in listbox when you press Ctrl + A key
combination
512:             if (!e.Control || e.KeyCode != Keys.A) return;
513:             for (var i = 0; i < lbFiles.Items.Count; i++)
514:             {
515:                 lbFiles.SetSelected(i, true);
516:             }
517:         }
518:
519:         /// <summary>
520:         /// This is used to delete selected items in listbox
521:         /// When menu item Delete is clicked and opened dialog
522:         /// </summary>
523:         /// <param name="sender">The sender info (For example Main Form)</param>
524:         /// <param name="e">Event Arguments</param>
525:         private void menuItemDelete_Click(object sender, EventArgs e)
526:         {
527:             if (lbFiles.Items.Count > 0)
528:             {
529:                 if (lbFiles.SelectedItems.Count > 0)
530:                 {
531:                     ShowDeleteDialog(out var deleteDialog);
532:                     if (deleteDialog == DialogResult.Yes)
533:                         DeleteFilesFromListBox();
534:                 }
535:                 else
536:                     GenerateNoSelectedFileInListBoxMessage();
537:             }
538:             else
539:                 GenerateNoFileInListBoxMessage();
540:         }
541:
542:         /// <summary>
543:         /// This method is used in
544:         /// When there is no selected file in listbox
545:         /// </summary>
546:         private static void GenerateNoSelectedFileInListBoxMessage()
547:         {
548:             MessageBox.Show(_resource.GetString("NoSelectedFile"),
_resource.GetString("AppTitle"),
549:                 MessageBoxButtons.OK, MessageBoxIcon.Error);
550:         }
551:
552:         /// <summary>
553:         /// This method is used to
554:         /// Show delete file dialog
555:         /// </summary>
556:         /// <param name="deleteDialog">Delete File Dialog</param>
557:         private void ShowDeleteDialog(out DialogResult deleteDialog)
558:         {
559:             deleteDialog = MessageBox.Show( _resource.GetString("DeleteWarning1") +

```

```

lbFiles.SelectedItems.Count + _resource.GetString("DeleteWarning2"),
_resource.GetString("AppTitle"), MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
560:     }
561:
562:     /// <summary>
563:     /// This method is used to Delete selected files from ListBox
564:     /// This is used different ways in this app
565:     /// So we created this as a method
566:     /// </summary>
567:     private void DeleteFilesFromListBox()
568:     {
569:         var fileCount = lbFiles.SelectedItems.Count;
570:         var removedItems = (from object t in lbFiles.SelectedItems select
t.ToString()).ToList();
571:
572:         var allItems = (from object item in lbFiles.Items where removedItems.All(j
=> j != item.ToString()) select item.ToString()).ToList();
573:
574:         lbFiles.Items.Clear();
575:         foreach (var item in allItems)
576:         {
577:             lbFiles.Items.Add(item);
578:         }
579:
580:         GenerateDeleteItemMessage(fileCount);
581:     }
582:
583:     /// <summary>
584:     /// This function is used to clear the file list in listbox
585:     /// </summary>
586:     /// <param name="sender">The sender info (For example Main Form)</param>
587:     /// <param name="e">Event Arguments</param>
588:     private void btnClearList_Click(object sender, EventArgs e)
589:     {
590:         var fileCount = lbFiles.Items.Count;
591:         lbFiles.Items.Clear();
592:         GenerateDeleteItemMessage(fileCount);
593:     }
594:
595:     /// <summary>
596:     /// This method is used after deleting items from listbox
597:     /// </summary>
598:     /// <param name="fileCount">Deleted File Count Information</param>
599:     private static void GenerateDeleteItemMessage(int fileCount)
600:     {
601:         MessageBox.Show(fileCount + _resource.GetString("DeleteFileMessage"),
_resource.GetString("AppTitle"), MessageBoxButtons.OK, MessageBoxIcon.Information);
602:     }
603:
604:     #endregion
605:
606:     #region Order Item Methods
607:
608:     /// <summary>
609:     /// This method is used to order elements in listbox ascending by full path
610:     /// </summary>
611:     /// <param name="sender">The sender info (For example Main Form)</param>
612:     /// <param name="e">Event Arguments</param>
613:     private void menuItemOrderByPathAscending_Click(object sender, EventArgs e)
614:     {
615:         if (lbFiles.Items.Count > 0)
616:             lbFiles = SortItemsByPath(lbFiles, OrderType.Ascending);
617:         else
618:             GenerateNoFileInListBoxMessage();
619:     }
620:
621:     /// <summary>
622:     /// This method is used to order elements in listbox descending by full path
623:     /// </summary>

```



```

624:     /// <param name="sender">The sender info (For example Main Form)</param>
625:     /// <param name="e">Event Arguments</param>
626:     private void menuItemOrderByPathDescending_Click(object sender, EventArgs e)
627:     {
628:         if (lbFiles.Items.Count > 0)
629:             lbFiles = SortItemsByPath(lbFiles, OrderType.Descending);
630:         else
631:             GenerateNoFileInListBoxMessage();
632:     }
633:
634:     /// <summary>
635:     /// This method is used to order elements in listbox ascending by file name
636:     /// </summary>
637:     /// <param name="sender">The sender info (For example Main Form)</param>
638:     /// <param name="e">Event Arguments</param>
639:     private void menuItemOrderByNameAscending_Click(object sender, EventArgs e)
640:     {
641:         if (lbFiles.Items.Count > 0)
642:             lbFiles = SortItemsByName(lbFiles, OrderType.Ascending);
643:         else
644:             GenerateNoFileInListBoxMessage();
645:     }
646:
647:     /// <summary>
648:     /// This method is used to order elements in listbox descending by file name
649:     /// </summary>
650:     /// <param name="sender">The sender info (For example Main Form)</param>
651:     /// <param name="e">Event Arguments</param>
652:     private void menuItemOrderByNameDescending_Click(object sender, EventArgs e)
653:     {
654:         if (lbFiles.Items.Count > 0)
655:             lbFiles = SortItemsByName(lbFiles, OrderType.Descending);
656:         else
657:             GenerateNoFileInListBoxMessage();
658:     }
659:
660:     /// <summary>
661:     /// This method is used to generate message
662:     /// When there is no item in listbox
663:     /// </summary>
664:     private static void GenerateNoFileInListBoxMessage()
665:     {
666:         MessageBox.Show(_resource.GetString("NoFile"),
667:             _resource.GetString("AppTitle"), MessageBoxButtons.OK, MessageBoxIcon.Error);
668:     }
669:
670:     /// <summary>
671:     /// This method is used to order items by full path in listbox
672:     /// And return them with the given order as parametre
673:     /// </summary>
674:     /// <param name="listBox">ListBox Info</param>
675:     /// <param name="orderType">Order Type is Ascending Or Not</param>
676:     private static ListBox SortItemsByPath(ListBox listBox, OrderType orderType)
677:     {
678:         var items = listBox.Items.OfType<object>().ToList();
679:         listBox.Items.Clear();
680:         listBox.Items.AddRange(orderType == OrderType.Ascending
681:             ? items.OrderBy(i => i).ToArray()
682:             : items.OrderByDescending(i => i).ToArray());
683:         return listBox;
684:     }
685:
686:     /// <summary>
687:     /// This method is used to order items by file name in listbox
688:     /// And return them with the given order as parametre
689:     /// </summary>
690:     /// <param name="listBox">ListBox Info</param>
691:     /// <param name="orderType">Order Type is Ascending Or Not</param>
692:     private static ListBox SortItemsByName(ListBox listBox, OrderType orderType)

```



```

692:         {
693:             var items = listBox.Items.OfType<object>().ToList();
694:             const char c = '\u005c';
695:             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();
696:
697:             fileList = orderType == OrderType.Ascending
698:                 ? fileList.OrderBy(j => j.FileName).ThenBy(j => j.FilePath).ToList()
699:                 : fileList.OrderByDescending(j => j.FileName).ThenByDescending(j =>
j.FilePath).ToList();
700:
701:             listBox.Items.Clear();
702:             foreach (var item in fileList)
703:                 listBox.Items.Add(item.FilePath);
704:
705:             return listBox;
706:         }
707:
708:     #endregion
709:
710:     #region Item Drag and Drop Methods
711:
712:     /// <summary>
713:     /// This is used to control mouse actions and if that is not right click
714:     /// Start to drag and drop item in listbox
715:     /// </summary>
716:     /// <param name="sender">The sender info (For example Main Form)</param>
717:     /// <param name="e">Event Arguments</param>
718:     private void lbFiles_MouseDown(object sender, MouseEventArgs e)
719:     {
720:         if (e.Button.Equals(MouseButtons.Right)) return;
721:         if (lbFiles.SelectedItem == null) return;
722:         lbFiles.DoDragDrop(lbFiles.SelectedItem, DragDropEffects.Move);
723:     }
724:
725:     /// <summary>
726:     /// This is used to give effect while dragging and dropping an item in listbox
727:     /// </summary>
728:     /// <param name="sender">The sender info (For example Main Form)</param>
729:     /// <param name="e">Event Arguments</param>
730:     private void lbFiles_DragOver(object sender, DragEventArgs e)
731:     {
732:         e.Effect = DragDropEffects.Move;
733:     }
734:
735:     /// <summary>
736:     /// This is used to drag and drop an item in listbox
737:     /// Which is used to order books and combine them with that order
738:     /// </summary>
739:     /// <param name="sender">The sender info (For example Main Form)</param>
740:     /// <param name="e">Event Arguments</param>
741:     private void lbFiles_DragDrop(object sender, DragEventArgs e)
742:     {
743:         var point = lbFiles.PointToClient(new Point(e.X, e.Y));
744:         var index = lbFiles.IndexFromPoint(point);
745:         if (index < 0) index = lbFiles.Items.Count - 1;
746:         // type of string because file names stored in string in listbox
747:         var data = e.Data.GetData(typeof(string));
748:         lbFiles.Items.Remove(data);
749:         lbFiles.Items.Insert(index, data);
750:     }
751:
752:     #endregion
753: }
754: }

```

## Namespaces

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

## 1.2.6 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 (see page 1)	This is namespace PdfCombiner.

## 1.2.7 PdfCombiner.csproj

This is file PdfCombiner.csproj.

## 1.2.8 PdfCombiner.sln

This is file PdfCombiner.sln.

## 1.2.9 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.

1.2.10 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:     }
```

```
32:         internal Resources() {
33:         }
34:
35:         /// <summary>
36:         ///     Returns the cached ResourceManager instance used by this class.
37:         /// </summary>
38:
39: [global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]
40:         internal static global::System.Resources.ResourceManager ResourceManager {
41:             get {
42:                 if (object.ReferenceEquals(resourceMan, null)) {
43:                     global::System.Resources.ResourceManager temp = new
44: global::System.Resources.ResourceManager("PdfCombiner.Properties.Resources",
45: typeof(Resources).Assembly);
46:                     resourceMan = temp;
47:                 }
48:                 return resourceMan;
49:             }
50:         }
51:         /// <summary>
52:         ///     Overrides the current thread's CurrentUICulture property for all
53:         ///     resource lookups using this strongly typed resource class.
54:         /// </summary>
55:
56: [global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]
57:         internal static global::System.Globalization.CultureInfo Culture {
58:             get {
59:                 return resourceCulture;
60:             }
61:             set {
62:                 resourceCulture = value;
63:             }
64:         }
65:         /// <summary>
66:         ///     Looks up a localized string similar to PDF Birlestirici.
67:         /// </summary>
68:         internal static string AppTitle {
69:             get {
70:                 return ResourceManager.GetString("AppTitle", resourceCulture);
71:             }
72:         }
```

Namespaces

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

1.2.11 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:     [global::System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.VisualStudio.Editors.SettingsDesigner.SettingsSingleFileGenerator",
16: "11.0.0.0")]
17:     internal sealed partial class Settings :
18:         global::System.Configuration.ApplicationSettingsBase
19:     {
20:         private static Settings defaultInstance =
21:             ((Settings)(global::System.Configuration.ApplicationSettingsBase.Synchronized(new
22: Settings())));
23:     }
```

Namespaces

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

# Index

## A

Ascending enumeration member 5  
AssemblyInfo.cs 33

## D

Descending enumeration member 5

## E

Enums class 4  
    about Enums class 4  
Enums.cs 34  
ExceptionExtensions class 5  
    about ExceptionExtensions class 5  
    GetAllMessages 5  
ExceptionExtensions.cs 35

## F

FileInfo class 6  
    about FileInfo class 6  
    FileName 6  
    FilePath 6  
FileInfo.cs 35  
Files 33  
FrmMain class 6  
    \_resource 9  
    about FrmMain class 6  
    AddPdfContentToTargetPdfFileITextSharp 12  
    AddPdfContentToTargetPdfFilePdfSharp 12  
    AddPdfFilesToList 13  
    btnAddFile 9  
    btnAddFile\_Click 13  
    btnAddFolder 10  
    btnAddFolder\_Click 14  
    btnClearList 10  
    btnClearList\_Click 14  
    btnCombineITextSharp 10  
    btnCombineITextSharp\_Click 15

btnCombinePdfSharp 10  
btnCombinePdfSharp\_Click 16  
cmbLanguage 10  
cmbLanguage\_SelectedIndexChanged 17  
components 10  
DeleteFilesFromListBox 17  
Dispose 18  
FillLanguageComboboxAndSetFirstLanguage 18  
FormMembersNameInitialization 18  
FrmMain 9  
FrmMain\_FormClosed 19  
FrmMain\_Shown 19  
GenerateAddFileMessage 20  
GenerateCombineFileMessage 20  
GenerateCombineWarningMessage 20  
GenerateDeleteItemMessage 21  
GenerateExceptionMessage 21  
GenerateNoFileInListBoxMessage 21  
GenerateNoSelectedFileInListBoxMessage 22  
InitializeComponent 22  
InitializeFileDialog 25  
LanguageList 10  
lbFiles 11  
lbFiles\_DragDrop 26  
lbFiles\_DragOver 26  
lbFiles\_KeyDown 26  
lbFiles\_MouseDown 27  
lbIDetails 11  
menuItemDelete 11  
menuItemDelete\_Click 27  
menuItemOrderByNameAscending 11  
menuItemOrderByNameAscending\_Click 28  
menuItemOrderByNameDescending 11  
menuItemOrderByNameDescending\_Click 28  
menuItemOrderByPathAscending 11  
menuItemOrderByPathAscending\_Click 28  
menuItemOrderByPathDescending 11  
menuItemOrderByPathDescending\_Click 29  
menuStrip 12  
pbFiles 12  
SaveOutputPdfFile 29

SetCombinationRatio 29	PdfCombiner.FrmMain.cmbLanguage 10
SetInitialValuesForCombiningFiles 30	PdfCombiner.FrmMain.cmbLanguage_SelectedIndexChanged 17
SetOutputFilePropertiesPdfSharp 30	PdfCombiner.FrmMain.components 10
SetProgramLanguage 30	PdfCombiner.FrmMain.DeleteFilesFromListBox 17
ShowDeleteDialog 31	PdfCombiner.FrmMain.Dispose 18
SortItemsByName 31	PdfCombiner.FrmMain.FillLanguageComboboxAndSetFirstLanguage 18
SortItemsByPath 32	PdfCombiner.FrmMain.FormMembersNameInitialization 18
FrmMain.cs 36	PdfCombiner.FrmMain.FrmMain 9
FrmMain.Designer.cs 48	PdfCombiner.FrmMain.FrmMain_FormClosed 19
	PdfCombiner.FrmMain.FrmMain_Shown 19
	PdfCombiner.FrmMain.GenerateAddFileMessage 20
	PdfCombiner.FrmMain.GenerateCombineFileMessage 20
	PdfCombiner.FrmMain.GenerateCombineWarningMessage 20
	PdfCombiner.FrmMain.GenerateDeleteItemMessage 21
	PdfCombiner.FrmMain.GenerateExceptionMessage 21
	PdfCombiner.FrmMain.GenerateNoFileInListBoxMessage 21
	PdfCombiner.FrmMain.GenerateNoSelectedFileInListBoxMessage 22
	PdfCombiner.FrmMain.InitializeComponent 22
	PdfCombiner.FrmMain.InitializeFileDialog 25
	PdfCombiner.FrmMain.LanguageList 10
	PdfCombiner.FrmMain.lbFiles 11
	PdfCombiner.FrmMain.lbFiles_DragDrop 26
	PdfCombiner.FrmMain.lbFiles_DragOver 26
	PdfCombiner.FrmMain.lbFiles_KeyDown 26
	PdfCombiner.FrmMain.lbFiles_MouseDown 27
	PdfCombiner.FrmMain.lblDetails 11
	PdfCombiner.FrmMain.menuItemDelete 11
	PdfCombiner.FrmMain.menuItemDelete_Click 27
	PdfCombiner.FrmMain.menuItemOrderByNameAscending 11
	PdfCombiner.FrmMain.menuItemOrderByNameAscending_Click 28
	PdfCombiner.FrmMain.menuItemOrderByNameDescending 11
	PdfCombiner.FrmMain.menuItemOrderByNameDescending_Click 28
	PdfCombiner.FrmMain.menuItemOrderByPathAscending 11
	PdfCombiner.FrmMain.menuItemOrderByPathAscending_Click 16

## P

PdfCombiner 1	
PdfCombiner namespace 1	
Classes 4	
PdfCombiner.csproj 52	
PdfCombiner.Enums 4	
PdfCombiner.Enums.OrderType 5	
PdfCombiner.Enums.OrderType enumeration 5	
PdfCombiner.ExceptionExtensions 5	
PdfCombiner.ExceptionExtensions.GetAllMessages 5	
PdfCombiner.FileInfo 6	
PdfCombiner.FileInfo.FileName 6	
PdfCombiner.FileInfo.FilePath 6	
PdfCombiner.FrmMain 6	
PdfCombiner.FrmMain._resource 9	
PdfCombiner.FrmMain.AddPdfContentToTargetPdfFileTextSharp 12	
PdfCombiner.FrmMain.AddPdfContentToTargetPdfFilePdfSharp 12	
PdfCombiner.FrmMain.AddPdfFilesToList 13	
PdfCombiner.FrmMain.btnAddFile 9	
PdfCombiner.FrmMain.btnAddFile_Click 13	
PdfCombiner.FrmMain.btnAddFolder 10	
PdfCombiner.FrmMain.btnAddFolder_Click 14	
PdfCombiner.FrmMain.btnClearList 10	
PdfCombiner.FrmMain.btnClearList_Click 14	
PdfCombiner.FrmMain.btnCombineTextSharp 10	
PdfCombiner.FrmMain.btnCombineTextSharp_Click 15	
PdfCombiner.FrmMain.btnCombinePdfSharp 10	
PdfCombiner.FrmMain.btnCombinePdfSharp_Click 16	

k  
 28  
 PdfCombiner.FrmMain.menuItemOrderByPathDescending 11  
 PdfCombiner.FrmMain.menuItemOrderByPathDescending\_Click  
 29  
 PdfCombiner.FrmMain.menuStrip 12  
 PdfCombiner.FrmMain.pbFiles 12  
 PdfCombiner.FrmMain.SaveOutputPdfFile 29  
 PdfCombiner.FrmMain.SetCombinationRatio 29  
 PdfCombiner.FrmMain.SetInitialValuesForCombiningFiles 30  
 PdfCombiner.FrmMain.SetOutputFilePropertiesPdfSharp 30  
 PdfCombiner.FrmMain.SetProgramLanguage 30  
 PdfCombiner.FrmMain.ShowDeleteDialog 31  
 PdfCombiner.FrmMain.SortItemsByName 31  
 PdfCombiner.FrmMain.SortItemsByPath 32  
 PdfCombiner.Program 32  
 PdfCombiner.Program.Main 32  
 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 52  
 Program class 32  
     about Program class 32  
     Main 32  
 Program.cs 52  
     resourceMan 3  
     ResourceManager 3  
     Resources 2  
     Resources.Designer.cs 53

## S

Settings class 3  
     about Settings class 3  
     defaultInstance 4  
 Settings.Designer.cs 54

## R

Resources class 1  
     about Resources class 1  
     AppTitle 3  
     Culture 3  
     resourceCulture 3