PDF Combiner

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

Table of Contents

1 Symbol Reference

1.1 PdfCombiner Namespace

This is namespace PdfCombiner.

Namespaces

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

Classes

	Name	Description
	BaseException (see page 4)	This function is used to take exceptions which can be thrown in PDF Combiner App
	ExceptionExtensions (see page 5)	This is class PdfCombiner.ExceptionExtensions.
	FileInfo (see page 6)	This class is used to have File's Full Path and Name
	FrmMain (see page 7)	This is class PdfCombiner.FrmMain.
	Program (see page 33)	This is class PdfCombiner.Program.

1.1.1 PdfCombiner.Properties Namespace

This is namespace PdfCombiner.Properties.

Classes

	Name	Description
	Resources (see page 1)	A strongly-typed resource class, for looking up localized strings, etc.
	Settings (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 (see page 1)	A strongly-typed resource class, for looking up localized strings, etc.
	Settings (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.StronglyType
dResourceBuilder",
"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 (☐ see page 2)	This is Resources, a member of class Resources.

Resources Fields

	Name	Description
◈ₐ§	resourceCulture (☐ see page 3)	This is resourceCulture, a member of class Resources.
◈ₐ§	resourceMan (☐ see page 3)	This is resourceMan, a member of class Resources.

Resources Properties

	Name	Description
⊟¿§	AppTitle (☐ see page 3)	Looks up a localized string similar to PDF Birlestirici.
⊟¿§	Culture (☐ see page 3)	Overrides the current thread's CurrentUICulture property for all resource lookups using this strongly typed resource class.
⊟¿§	ResourceManager (☐ see page 3)	Returns the cached ResourceManager instance used by this class.

1.1.1.1.1.1 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: }
```

1.1.1.1.1.2 Resources Fields

Created with a commercial version of Doc-O-Matic. In order to make this message disappear you need to register this software. If you have problems registering this software please contact us at support@toolsfactory.com.

2

1.1.1.1.1.2.1 Resources.resourceCulture Field

C#

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

Description

This is resourceCulture, a member of class Resources.

1.1.1.1.1.2.2 Resources.resourceMan Field

C#

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

Description

This is resourceMan, a member of class Resources.

1.1.1.1.1.3 Resources Properties

1.1.1.1.1.3.1 Resources.AppTitle Property

Looks up a localized string similar to PDF Birlestirici.

C#

```
internal static string AppTitle;
```

1.1.1.1.1.3.2 Resources.Culture Property

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

C#

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

1.1.1.1.1.3.3 Resources.ResourceManager Property

Returns the cached ResourceManager instance used by this class.

C#

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

1.1.1.1.2 Settings Class

Class Hierarchy

```
global::System.Configuration.ApplicationSettingsBase  →  PdfCombiner.Properties.Settings
```
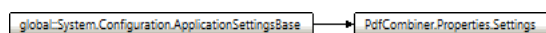
C#

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

File

Settings.Designer.cs (⬚ see page 55)

Description

This is class PdfCombiner.Properties.Settings.

Settings Fields

	Name	Description
🔹S	defaultInstance (▣ see page 4)	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
🔹A	BaseException (▣ see page 4)	This function is used to take exceptions which can be thrown in PDF Combiner App
🔹S	ExceptionExtensions (▣ see page 5)	This is class PdfCombiner.ExceptionExtensions.
🔹	FileInfo (▣ see page 6)	This class is used to have File's Full Path and Name
🔹	FrmMain (▣ see page 7)	This is class PdfCombiner.FrmMain.
🔹S	Program (▣ see page 33)	This is class PdfCombiner.Program.

1.1.2.1 BaseException Class

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

Class Hierarchy
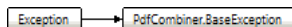
C#

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

File

BaseException.cs (▣ see page 34)

Methods

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

1.1.2.1.1 BaseException Constructor

1.1.2.1.1.1 BaseException.BaseException Constructor ()

C#

```
protected BaseException();
```

Description

This is BaseException, a member of class BaseException.

Body Source

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

1.1.2.1.1.2 BaseException.BaseException Constructor (string)

C#

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

Description

This is BaseException, a member of class BaseException.

Body Source

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

1.1.2.1.1.3 BaseException.BaseException Constructor (string, Exception)

C#

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

Description

This is BaseException, a member of class BaseException.

Body Source

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

1.1.2.2 ExceptionExtensions Class

Class Hierarchy

PdfCombiner.ExceptionExtensions

C#

```
public static class ExceptionExtensions;
```

File

ExceptionExtensions.cs (▣ see page 35)

Description

This is class PdfCombiner.ExceptionExtensions.

ExceptionExtensions Methods

	Name	Description
⇒◆⑤	GetAllMessages (⧉ see page 6)	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#

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

Parameters

Parameters	Description
this Exception exception	Exception

Returns

Detailed Error Message

Body Source

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

1.1.2.3 FileInfo Class

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

Class Hierarchy

PdfCombiner.FileInfo

C#

```csharp
public class FileInfo;
```

File

FileInfo.cs (⧉ see page 36)

FileInfo Properties

	Name	Description
🖼	FileName (⧉ see page 7)	File Name Property
🖼	FilePath (⧉ see page 7)	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
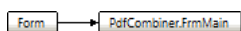
C#

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

File

FrmMain.Designer.cs (🗗 see page 48)

Description

This is class PdfCombiner.FrmMain.

Methods

	Name	Description
🔧	FrmMain (🗗 see page 9)	This is FrmMain, a member of class FrmMain.

FrmMain Fields

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

🔹	menuItemOrderByPathAscending (🖻 see page 12)	This is menuItemOrderByPathAscending, a member of class FrmMain.
🔹	menuItemOrderByPathDescending (🖻 see page 12)	This is menuItemOrderByPathDescending, a member of class FrmMain.
🔹	menuStrip (🖻 see page 12)	This is menuStrip, a member of class FrmMain.
🔹	pbFiles (🖻 see page 12)	This is pbFiles, a member of class FrmMain.

FrmMain Methods

	Name	Description
🔹S	AddPdfContentToTargetPdfFileITextSharp (🖻 see page 12)	This method is used to take source pdf file content And add to the target combined pdf file
🔹S	AddPdfContentToTargetPdfFilePdfSharp (🖻 see page 13)	This method is used to take source pdf file content And add to the target combined pdf file
🔹	AddPdfFilesToList (🖻 see page 13)	This method is used both adding files or folders to list
🔹	btnAddFile_Click (🖻 see page 14)	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 (🖻 see page 14)	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 (🖻 see page 15)	This function is used to clear the file list in listbox
🔹	btnCombineITextSharp_Click (🖻 see page 15)	This function is used to take list of PDF files in listbox, Combine them with a single file in location which you choose in folder dialog And you can see the progress in progress bar It uses iTextSharp Nuget Package
🔹	btnCombinePdfSharp_Click (🖻 see page 16)	This function is used to take list of PDF files in listbox, Combine them with a single file in location which you choose in folder dialog And you can see the progress in progress bar It uses PdfSharp Nuget Package
🔹	cmbLanguage_SelectedIndexChanged (🖻 see page 17)	This function is used to set the _resource (🖻 see page 10) file And with this, you can see form elements, info messages etc. with the selected _resource (🖻 see page 10) file contents. That is used to multi language support
🔹	DeleteFilesFromListBox (🖻 see page 18)	This method is used to Delete selected files from ListBox This is used different ways in this app So we created this as a method
🔹	Dispose (🖻 see page 18)	Clean up any resources being used.
🔹	FillLanguageComboboxAndSetFirstLanguage (🖻 see page 18)	This method is used to Fill the Language combobox And set initial value with The language computer used in
🔹	FormMembersNameInitialization (🖻 see page 19)	This method is used to change text of form members by the Resource file which is selected
🔹	FrmMain_FormClosed (🖻 see page 19)	This function is used to terminate application
🔹	FrmMain_Shown (🖻 see page 20)	This function is used to assign menu to listbox When form is shown
🔹	GenerateAddFileMessage (🖻 see page 20)	This method is used to show info message after adding files or folders
🔹	GenerateCombineFileMessage (🖻 see page 20)	This method is used to set successfull combining file message And set value of Form Caption etc.
🔹S	GenerateCombineWarningMessage (🖻 see page 21)	This method is used When user just wnat to combine 1 pdf file
🔹S	GenerateDeleteItemMessage (🖻 see page 21)	This method is used after deleting items from listbox
🔹	GenerateExceptionMessage (🖻 see page 21)	This method is used to Show error message when exception occurs and set form title initial value

⇒🔹S	GenerateNoFileInListBoxMessage (🔳 see page 22)	This method is used to generate message When there is no item in listbox
⇒🔹S	GenerateNoSelectedFileInListBoxMessage (🔳 see page 22)	This method is used in When there is no selected file in listbox
⇒🔹	InitializeComponent (🔳 see page 22)	Required method for Designer support - do not modify the contents of this method with the code editor.
⇒🔹S	InitializeFileDialog (🔳 see page 26)	This function is used to set options of file dialog like Filter, start path etc.
⇒🔹	lbFiles_DragDrop (🔳 see page 26)	This is used to drag and drop an item in listbox Which is used to order books and combine them with that order
⇒🔹	lbFiles_DragOver (🔳 see page 26)	This is used to give effect while dragging and dropping an item in listbox
⇒🔹	lbFiles_KeyDown (🔳 see page 27)	This function is used to operate some process in files in listbox which are chosen
⇒🔹	lbFiles_MouseDown (🔳 see page 27)	This is used to control mouse actions and if that is not right click Start to drag and drop item in listbox
⇒🔹	menuItemDelete_Click (🔳 see page 28)	This is used to delete selected items in listbox When menu item Delete is clicked and opened dialog
⇒🔹	menuItemOrderByNameAscending_Click (🔳 see page 28)	This method is used to order elements in listbox ascending by file name
⇒🔹	menuItemOrderByNameDescending_Click (🔳 see page 28)	This method is used to order elements in listbox descending by file name
⇒🔹	menuItemOrderByPathAscending_Click (🔳 see page 29)	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
⇒🔹S	SaveOutputPdfFile (🔳 see page 29)	This method is used to Save Combined PDF File
⇒🔹	SetCombinationRatio (🔳 see page 30)	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
⇒🔹S	SetOutputFilePropertiesPdfSharp (🔳 see page 31)	This method is used to set combined PDF file Properties (🔳 see page 1) like Compression etc.
⇒🔹	SetProgramLanguage (🔳 see page 31)	This method is used to Set Program (🔳 see page 33) Language By the selection from Language Combobox Finally, with the selected language Form elements renamed with the selected language
⇒🔹	ShowDeleteDialog (🔳 see page 31)	This method is used to Show delete file dialog
⇒🔹S	SortItemsByName (🔳 see page 32)	This method is used to order items by file name in listbox And return them with the given order as parametre
⇒🔹S	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,
ref int combinedFiles)
 2: {
 3:     var pdfReader =
 4:         new iTextSharp.text.pdf.PdfReader(t.ToString());
 5:     pdfReader.ConsolidateNamedDestinations();
 6:     for (var j = 1; j <= pdfReader.NumberOfPages; j++)
 7:     {
 8:         var page = pdfWriter.GetImportedPage(pdfReader, j);
 9:         pdfWriter.AddPage(page);
10:     }
11:
12:     pdfReader.Close();
13:     combinedFiles++;
14: }
```

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

1.1.2.4.3.3 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.FileNames == null) return;
10:
11:         AddPdfFilesToList(dialogAddFile.FileNames, 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 ||
string.IsNullOrEmpty(dialogExport.SelectedPath)) return;
13:
14:                    SetInitialValuesForCombiningFiles(dialogExport, out var outputFileName,
out var fileCount,
15:                        out var combinedFiles);
16:
17:                    var outputFile = new Document();
18:                    using (var outputFileStream = new FileStream(outputFileName,
FileMode.Create))
19:                    {
20:                        using (var pdfWriter = new PdfCopy(outputFile, outputFileStream))
21:                        {
22:                            pdfWriter.SetFullCompression();
23:                            outputFile.Open();
24:                            foreach (var t in lbFiles.Items)
25:                            {
26:                                try
27:                                {
28:                                    AddPdfContentToTargetPdfFileITextSharp(t, pdfWriter,
ref combinedFiles);
29:                                }
30:                                catch (Exception)
31:                                {
32:                                    fileCount--;
33:                                }
34:
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 ||
string.IsNullOrEmpty(dialogExport.SelectedPath)) return;
13:
14:                     SetInitialValuesForCombiningFiles(dialogExport, out var outputFileName,
out var fileCount,
15:                         out var combinedFiles);
16:
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 10) file And with this, you can see form elements, info messages etc. with the selected _resource (⬚ see page 10) file contents. That is used to multi language support

C#

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

Parameters

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

Body Source

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

1.1.2.4.3.10 **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
t.ToString()).ToList();
 5:
 6:     var allItems = (from object item in lbFiles.Items where removedItems.All(j => j !=
item.ToString()) select item.ToString()).ToList();
 7:
 8:     lbFiles.Items.Clear();
 9:     foreach (var item in allItems)
10:     {
11:         lbFiles.Items.Add(item);
12:     }
13:
14:     GenerateDeleteItemMessage(fileCount);
15: }
```

1.1.2.4.3.11 **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,
4:         MessageBoxIcon.Information);
5:     Application.Exit();
6: }
```

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

1.1.2.4.3.18 FrmMain.GenerateCombineWarningMessage Method

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

C#

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

Body Source

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

1.1.2.4.3.19 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"),
4:         MessageBoxButtons.OK, MessageBoxIcon.Information);
5: }
```

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

1.1.2.4.3.21 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,
4:         MessageBoxIcon.Error);
5: }
```

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"),
_resource.GetString("AppTitle"),
4:         MessageBoxButtons.OK, MessageBoxIcon.Error);
5: }
```

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;
```

1

```
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#

```csharp
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

```csharp
 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#

```csharp
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

```csharp
1: private void menuItemOrderByNameAscending_Click(object sender, EventArgs e)
2: {
3:     if (lbFiles.Items.Count > 0)
4:         lbFiles = SortItemsByName(lbFiles, true);
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#

```csharp
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, false);
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, true);
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, false);
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 33) Language By the selection from Language Combobox Finally, with the selected language Form elements renamed with the selected language

C#

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

Body Source

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

1.1.2.4.3.39 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(
4:         _resource.GetString("DeleteWarning1") + lbFiles.SelectedItems.Count +
5:         _resource.GetString("DeleteWarning2"), _resource.GetString("AppTitle"),
6:         MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
7: }
```

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, bool ascending);
```

Parameters

Parameters	Description
ListBox listBox	ListBox Info
bool ascending	Order Type is Ascending Or Not

Body Source

```
 1: private static ListBox SortItemsByName(ListBox listBox, bool ascending)
 2: {
 3:     var items = listBox.Items.OfType<object>().ToList();
 4:     const char c = '\u005c';
 5:     var fileList = (from item in items select item.ToString() into fullPath let list =
fullPath.Split(c).ToList() where list.Count > 0 select new FileInfo {FilePath = fullPath,
FileName = list[list.Count - 1]}).ToList();
 6:
 7:     fileList = 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, bool ascending);
```

Parameters

Parameters	Description
ListBox listBox	ListBox Info
bool ascending	Order Type is Ascending Or Not

Body Source

```
1: private static ListBox SortItemsByPath(ListBox listBox, bool ascending)
2: {
3:     var items = listBox.Items.OfType<object>().ToList();
4:     listBox.Items.Clear();
5:     listBox.Items.AddRange(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
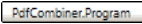
C#

```
internal static class Program;
```

File

Program.cs (see page 53)

Description

This is class PdfCombiner.Program.

Program Methods

	Name	Description
	Main (see page 33)	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 (see page 34)	This is file AssemblyInfo.cs.
BaseException.cs (see page 34)	This is file BaseException.cs.
ExceptionExtensions.cs (see page 35)	This is file ExceptionExtensions.cs.
FileInfo.cs (see page 36)	This is file FileInfo.cs.
FrmMain.cs (see page 36)	This is file FrmMain.cs.

FrmMain.Designer.cs (⊡ see page 48)	This is file FrmMain.Designer.cs.
PdfCombiner.csproj (⊡ see page 53)	This is file PdfCombiner.csproj.
PdfCombiner.sln (⊡ see page 53)	This is file PdfCombiner.sln.
Program.cs (⊡ see page 53)	This is file Program.cs.
Resources.Designer.cs (⊡ see page 53)	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 (⊡ see page 55)	This code was generated by a tool. Runtime Version:4.0.30319.42000 Changes to this file may cause incorrect behavior and will be lost if the code is regenerated.

1.2.1 AssemblyInfo.cs

This is file AssemblyInfo.cs.

Body Source

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

1.2.2 BaseException.cs

This is file BaseException.cs.

Body Source

```csharp
 1: ?using System;
 2:
 3: namespace PdfCombiner
 4: {
 5:     /// <summary>
 6:     /// This function is used to take exceptions which can be thrown in
 7:     /// PDF Combiner App
 8:     /// </summary>
 9:     [Serializable]
10:     public abstract class BaseException : Exception
11:     {
12:         protected BaseException()
13:         {
14:         }
15:
16:         protected BaseException(string message) : base(message)
17:         {
18:         }
19:
20:         protected BaseException(string message, Exception innerException) :
base(message, innerException)
21:         {
22:         }
23:     }
24: }
```

Namespaces

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

1.2.3 **ExceptionExtensions.cs**

This is file ExceptionExtensions.cs.

Body Source

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

Namespaces

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

1.2.4 **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:
17: namespace PdfCombiner
18: {
19:     public partial class FrmMain : Form
20:     {
```

```csharp
 21:          private static ResourceManager _resource;
 22:
 23:          // Culture lists whose resources have been added to project
 24:          private static readonly List<string> LanguageList = new List<string>
 25:              {"EN", "TR", "DE", "FR", "RU", "ES", "IT", "ZH", "AR", "NL", "PT", "IN",
"ID", "JP", "BG"};
 26:
 27:          public FrmMain()
 28:          {
 29:              InitializeComponent();
 30:          }
 31:
 32:          #region Form Init, Language Settings and Finalize Methods
 33:
 34:          /// <summary>
 35:          /// This function is used to terminate application
 36:          /// </summary>
 37:          /// <param name="sender">The sender info (For example Main Form)</param>
 38:          /// <param name="e">Event Arguments</param>
 39:          private void FrmMain_FormClosed(object sender, FormClosedEventArgs e)
 40:          {
 41:              MessageBox.Show(_resource.GetString("ThanksMessage"),
_resource.GetString("AppTitle"), MessageBoxButtons.OK,
 42:                  MessageBoxIcon.Information);
 43:              Application.Exit();
 44:          }
 45:
 46:          /// <summary>
 47:          /// 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:
```

```csharp
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(),
386:                     PdfDocumentOpenMode.Import);
387:                 var count = inputDocument.PageCount;
388:                 for (var idx = 0; idx < count; idx++)
389:                 {
390:                     var page = inputDocument.Pages[idx];
391:                     outputFile.AddPage(page);
392:                 }
393:
394:                 inputDocument.Close();
395:                 combinedFiles++;
396:             }
397:
398:             /// <summary>
399:             /// This method is used to set combined PDF file
400:             /// Properties like Compression etc.
401:             /// </summary>
402:             /// <param name="outputFile">Combined PDF File</param>
403:             private static void SetOutputFilePropertiesPdfSharp(PdfDocument outputFile)
```

```
404:            {
405:                    outputFile.Options.CompressContentStreams = true;
406:                    outputFile.Options.EnableCcittCompressionForBilevelImages = true;
407:                    outputFile.Options.FlateEncodeMode = PdfFlateEncodeMode.BestCompression;
408:            }
409:
410:        /// <summary>
411:        /// This method is used to take source pdf file content
412:        /// And add to the target combined pdf file
413:        /// </summary>
414:        /// <param name="t">Source PDF File Path</param>
415:        /// <param name="pdfWriter">PDF Writer</param>
416:        /// <param name="combinedFiles">Combined File Count</param>
417:        private static void AddPdfContentToTargetPdfFileITextSharp(object t, PdfCopy
pdfWriter, ref int combinedFiles)
418:            {
419:                var pdfReader =
420:                    new iTextSharp.text.pdf.PdfReader(t.ToString());
421:                pdfReader.ConsolidateNamedDestinations();
422:                for (var j = 1; j <= pdfReader.NumberOfPages; j++)
423:                {
424:                    var page = pdfWriter.GetImportedPage(pdfReader, j);
425:                    pdfWriter.AddPage(page);
426:                }
427:
428:                pdfReader.Close();
429:                combinedFiles++;
430:            }
431:
432:        /// <summary>
433:        /// This method is used
434:        /// When user just wnat to combine 1 pdf file
435:        /// </summary>
436:        private static void GenerateCombineWarningMessage()
437:            {
438:                MessageBox.Show(_resource.GetString("CombineWarning"),
_resource.GetString("AppTitle"),
439:                    MessageBoxButtons.OK, MessageBoxIcon.Error);
440:            }
441:
442:        /// <summary>
443:        /// This method is used to
444:        /// Show error message when exception occurs and set form title initial value
445:        /// </summary>
446:        /// <param name="ex">Exception information</param>
447:        private void GenerateExceptionMessage(Exception ex)
448:            {
449:                MessageBox.Show(_resource.GetString("CombineErrorMessage") +
ex.GetAllMessages(),
450:                    _resource.GetString("AppTitle"), MessageBoxButtons.OK,
MessageBoxIcon.Error);
451:                pbFiles.Value = pbFiles.Minimum;
452:                if (ActiveForm != null) ActiveForm.Text = _resource.GetString("AppTitle");
453:            }
454:
455:        /// <summary>
456:        /// This method is used to set initial values of some properties
457:        /// Which are used in combining PDF files
458:        /// In both PdfSharp and ITextSharp
459:        /// </summary>
460:        /// <param name="dialogExport">Export Dialog Info</param>
461:        /// <param name="outputFileName">Output File Name</param>
462:        /// <param name="fileCount">Total File Count</param>
463:        /// <param name="combinedFiles">Combined Files Count</param>
464:        private void SetInitialValuesForCombiningFiles(FolderBrowserDialog
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:
472:            /// <summary>
473:            /// This method is used to set combination degree while combining PDF files
474:            /// And show this ratio on Form Caption
475:            /// </summary>
476:            /// <param name="combinedFiles">Combined File Count</param>
477:            /// <param name="fileCount">Total File Count</param>
478:            private void SetCombinationRatio(int combinedFiles, int fileCount)
479:            {
480:                pbFiles.Value = combinedFiles * 100 / fileCount;
481:                if (pbFiles.Value > pbFiles.Maximum)
482:                    pbFiles.Value = pbFiles.Maximum;
483:                if (ActiveForm != null) ActiveForm.Text = @"%" + pbFiles.Value;
484:            }
485:
486:            /// <summary>
487:            /// This method is used to set successfull combining file message
488:            /// And set value of Form Caption etc.
489:            /// </summary>
490:            /// <param name="fileCount">Total File Count</param>
491:            /// <param name="outputFileName">Output File Name</param>
492:            private void GenerateCombineFileMessage(int fileCount, string outputFileName)
493:            {
494:                MessageBox.Show(fileCount + _resource.GetString("CombineFileMessage") +
     outputFileName,
495:                    _resource.GetString("AppTitle"), MessageBoxButtons.OK,
     MessageBoxIcon.Information);
496:                pbFiles.Value = pbFiles.Minimum;
497:                if (ActiveForm != null) ActiveForm.Text = _resource.GetString("AppTitle");
498:            }
499:
500:            #endregion
501:
502:            #region Delete Item Methods
503:
504:            /// <summary>
505:            /// This function is used to operate some process in files in listbox which
     are chosen
506:            /// </summary>
507:            /// <param name="sender">The sender info (For example Main Form)</param>
508:            /// <param name="e">Event Arguments</param>
509:            private void lbFiles_KeyDown(object sender, KeyEventArgs e)
510:            {
511:                // this is used to delete selected files in listbox when you press Delete
     key
512:                if (e.KeyValue == Keys.Delete.GetHashCode())
513:                    DeleteFilesFromListBox();
514:                // This is used to select all items in listbox when you press Ctrl + A key
     combination
515:                if (!e.Control || e.KeyCode != Keys.A) return;
516:                for (var i = 0; i < lbFiles.Items.Count; i++)
517:                {
518:                    lbFiles.SetSelected(i, true);
519:                }
520:            }
521:
522:            /// <summary>
523:            /// This is used to delete selected items in listbox
524:            /// When menu item Delete is clicked and opened dialog
525:            /// </summary>
526:            /// <param name="sender">The sender info (For example Main Form)</param>
527:            /// <param name="e">Event Arguments</param>
528:            private void menuItemDelete_Click(object sender, EventArgs e)
529:            {
530:                if (lbFiles.Items.Count > 0)
531:                {
```

```
532:                    if (lbFiles.SelectedItems.Count > 0)
533:                    {
534:                        ShowDeleteDialog(out var deleteDialog);
535:                        if (deleteDialog == DialogResult.Yes)
536:                            DeleteFilesFromListBox();
537:                    }
538:                    else
539:                        GenerateNoSelectedFileInListBoxMessage();
540:                }
541:                else
542:                    GenerateNoFileInListBoxMessage();
543:            }
544:
545:            /// <summary>
546:            /// This method is used in
547:            /// When there is no selected file in listbox
548:            /// </summary>
549:            private static void GenerateNoSelectedFileInListBoxMessage()
550:            {
551:                MessageBox.Show(_resource.GetString("NoSelectedFile"),
_resource.GetString("AppTitle"),
552:                    MessageBoxButtons.OK, MessageBoxIcon.Error);
553:            }
554:
555:            /// <summary>
556:            /// This method is used to
557:            /// Show delete file dialog
558:            /// </summary>
559:            /// <param name="deleteDialog">Delete File Dialog</param>
560:            private void ShowDeleteDialog(out DialogResult deleteDialog)
561:            {
562:                deleteDialog = MessageBox.Show(
563:                    _resource.GetString("DeleteWarning1") + lbFiles.SelectedItems.Count +
564:                    _resource.GetString("DeleteWarning2"), _resource.GetString("AppTitle"),
565:                    MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
566:            }
567:
568:            /// <summary>
569:            /// This method is used to Delete selected files from ListBox
570:            /// This is used different ways in this app
571:            /// So we created this as a method
572:            /// </summary>
573:            private void DeleteFilesFromListBox()
574:            {
575:                var fileCount = lbFiles.SelectedItems.Count;
576:                var removedItems = (from object t in lbFiles.SelectedItems select
t.ToString()).ToList();
577:
578:                var allItems = (from object item in lbFiles.Items where removedItems.All(j
=> j != item.ToString()) select item.ToString()).ToList();
579:
580:                lbFiles.Items.Clear();
581:                foreach (var item in allItems)
582:                {
583:                    lbFiles.Items.Add(item);
584:                }
585:
586:                GenerateDeleteItemMessage(fileCount);
587:            }
588:
589:            /// <summary>
590:            /// This function is used to clear the file list in listbox
591:            /// </summary>
592:            /// <param name="sender">The sender info (For example Main Form)</param>
593:            /// <param name="e">Event Arguments</param>
594:            private void btnClearList_Click(object sender, EventArgs e)
595:            {
596:                var fileCount = lbFiles.Items.Count;
597:                lbFiles.Items.Clear();
```

```
598:               GenerateDeleteItemMessage(fileCount);
599:           }
600:
601:           /// <summary>
602:           /// This method is used after deleting items from listbox
603:           /// </summary>
604:           /// <param name="fileCount">Deleted File Count Information</param>
605:           private static void GenerateDeleteItemMessage(int fileCount)
606:           {
607:               MessageBox.Show(fileCount + _resource.GetString("DeleteFileMessage"),
_resource.GetString("AppTitle"),
608:                   MessageBoxButtons.OK, MessageBoxIcon.Information);
609:           }
610:
611:           #endregion
612:
613:           #region Order Item Methods
614:
615:           /// <summary>
616:           /// This method is used to order elements in listbox ascending by full path
617:           /// </summary>
618:           /// <param name="sender">The sender info (For example Main Form)</param>
619:           /// <param name="e">Event Arguments</param>
620:           private void menuItemOrderByPathAscending_Click(object sender, EventArgs e)
621:           {
622:               if (lbFiles.Items.Count > 0)
623:                   lbFiles = SortItemsByPath(lbFiles, true);
624:               else
625:                   GenerateNoFileInListBoxMessage();
626:           }
627:
628:           /// <summary>
629:           /// This method is used to order elements in listbox descending by full path
630:           /// </summary>
631:           /// <param name="sender">The sender info (For example Main Form)</param>
632:           /// <param name="e">Event Arguments</param>
633:           private void menuItemOrderByPathDescending_Click(object sender, EventArgs e)
634:           {
635:               if (lbFiles.Items.Count > 0)
636:                   lbFiles = SortItemsByPath(lbFiles, false);
637:               else
638:                   GenerateNoFileInListBoxMessage();
639:           }
640:
641:           /// <summary>
642:           /// This method is used to order elements in listbox ascending by file name
643:           /// </summary>
644:           /// <param name="sender">The sender info (For example Main Form)</param>
645:           /// <param name="e">Event Arguments</param>
646:           private void menuItemOrderByNameAscending_Click(object sender, EventArgs e)
647:           {
648:               if (lbFiles.Items.Count > 0)
649:                   lbFiles = SortItemsByName(lbFiles, true);
650:               else
651:                   GenerateNoFileInListBoxMessage();
652:           }
653:
654:           /// <summary>
655:           /// This method is used to order elements in listbox descending by file name
656:           /// </summary>
657:           /// <param name="sender">The sender info (For example Main Form)</param>
658:           /// <param name="e">Event Arguments</param>
659:           private void menuItemOrderByNameDescending_Click(object sender, EventArgs e)
660:           {
661:               if (lbFiles.Items.Count > 0)
662:                   lbFiles = SortItemsByName(lbFiles, false);
663:               else
664:                   GenerateNoFileInListBoxMessage();
665:           }
```

1

```
666:
667:            /// <summary>
668:            /// This method is used to generate message
669:            /// When there is no item in listbox
670:            /// </summary>
671:            private static void GenerateNoFileInListBoxMessage()
672:            {
673:                MessageBox.Show(_resource.GetString("NoFile"),
_resource.GetString("AppTitle"), MessageBoxButtons.OK,
674:                    MessageBoxIcon.Error);
675:            }
676:
677:            /// <summary>
678:            /// This method is used to order items by full path in listbox
679:            /// And return them with the given order as parametre
680:            /// </summary>
681:            /// <param name="listBox">ListBox Info</param>
682:            /// <param name="ascending">Order Type is Ascending Or Not</param>
683:            private static ListBox SortItemsByPath(ListBox listBox, bool ascending)
684:            {
685:                var items = listBox.Items.OfType<object>().ToList();
686:                listBox.Items.Clear();
687:                listBox.Items.AddRange(ascending
688:                    ? items.OrderBy(i => i).ToArray()
689:                    : items.OrderByDescending(i => i).ToArray());
690:                return listBox;
691:            }
692:
693:            /// <summary>
694:            /// This method is used to order items by file name in listbox
695:            /// And return them with the given order as parametre
696:            /// </summary>
697:            /// <param name="listBox">ListBox Info</param>
698:            /// <param name="ascending">Order Type is Ascending Or Not</param>
699:            private static ListBox SortItemsByName(ListBox listBox, bool ascending)
700:            {
701:                var items = listBox.Items.OfType<object>().ToList();
702:                const char c = '\u005c';
703:                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();
704:
705:                fileList = ascending
706:                    ? fileList.OrderBy(j => j.FileName).ThenBy(j => j.FilePath).ToList()
707:                    : fileList.OrderByDescending(j => j.FileName).ThenByDescending(j =>
j.FilePath).ToList();
708:
709:                listBox.Items.Clear();
710:                foreach (var item in fileList)
711:                    listBox.Items.Add(item.FilePath);
712:
713:                return listBox;
714:            }
715:
716:            #endregion
717:
718:            #region Item Drag and Drop Methods
719:
720:            /// <summary>
721:            /// This is used to control mouse actions and if that is not right click
722:            /// Start to drag and drop item in listbox
723:            /// </summary>
724:            /// <param name="sender">The sender info (For example Main Form)</param>
725:            /// <param name="e">Event Arguments</param>
726:            private void lbFiles_MouseDown(object sender, MouseEventArgs e)
727:            {
728:                if (e.Button.Equals(MouseButtons.Right)) return;
729:                if (lbFiles.SelectedItem == null) return;
730:                lbFiles.DoDragDrop(lbFiles.SelectedItem, DragDropEffects.Move);
```

```
731:            }
732:
733:            /// <summary>
734:            /// This is used to give effect while dragging and dropping an item in listbox
735:            /// </summary>
736:            /// <param name="sender">The sender info (For example Main Form)</param>
737:            /// <param name="e">Event Arguments</param>
738:            private void lbFiles_DragOver(object sender, DragEventArgs e)
739:            {
740:                e.Effect = DragDropEffects.Move;
741:            }
742:
743:            /// <summary>
744:            /// This is used to drag and drop an item in listbox
745:            /// Which is used to order books and combine them with that order
746:            /// </summary>
747:            /// <param name="sender">The sender info (For example Main Form)</param>
748:            /// <param name="e">Event Arguments</param>m
749:            private void lbFiles_DragDrop(object sender, DragEventArgs e)
750:            {
751:                var point = lbFiles.PointToClient(new Point(e.X, e.Y));
752:                var index = lbFiles.IndexFromPoint(point);
753:                if (index < 0) index = lbFiles.Items.Count - 1;
754:                // type of string because file names stored in string in listbox
755:                var data = e.Data.GetData(typeof(string));
756:                lbFiles.Items.Remove(data);
757:                lbFiles.Items.Insert(index, data);
758:            }
759:
760:            #endregion
761:        }
762: }
```

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.AutoScaleMode.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: [global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.StronglyType
dResourceBuilder",
"15.0.0.0")]
 23:     [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
 24:     [global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
 25:     internal class Resources {
 26:
 27:         private static global::System.Resources.ResourceManager resourceMan;
 28:
 29:         private static global::System.Globalization.CultureInfo resourceCulture;
 30:
 31:
[global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1811:AvoidUncalledPrivateCode")]
 32:         internal Resources() {
 33:         }
 34:
 35:         /// <summary>
 36:         ///   Returns the cached ResourceManager instance used by this class.
 37:         /// </summary>
 38:
[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.Editor
BrowsableState.Advanced)]
 39:         internal static global::System.Resources.ResourceManager ResourceManager {
 40:             get {
 41:                 if (object.ReferenceEquals(resourceMan, null)) {
 42:                     global::System.Resources.ResourceManager temp = new
global::System.Resources.ResourceManager("PdfCombiner.Properties.Resources",
typeof(Resources).Assembly);
 43:                     resourceMan = temp;
 44:                 }
 45:                 return resourceMan;
 46:             }
 47:         }
 48:
 49:         /// <summary>
 50:         ///   Overrides the current thread's CurrentUICulture property for all
 51:         ///   resource lookups using this strongly typed resource class.
 52:         /// </summary>
 53:
[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.Editor
BrowsableState.Advanced)]
 54:         internal static global::System.Globalization.CultureInfo Culture {
 55:             get {
```

```
56:                     return resourceCulture;
57:                 }
58:             set {
59:                     resourceCulture = value;
60:                 }
61:         }
62:
63:         /// <summary>
64:         ///   Looks up a localized string similar to PDF Birlestirici.
65:         /// </summary>
66:         internal static string AppTitle {
67:             get {
68:                     return ResourceManager.GetString("AppTitle", resourceCulture);
69:                 }
70:         }
71:     }
72: }
```

Namespaces

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

1.2.11 **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.Sett
ingsDesigner.SettingsSingleFileGenerator",
"11.0.0.0")]
16:     internal sealed partial class Settings :
global::System.Configuration.ApplicationSettingsBase
17:     {
18:
19:         private static Settings defaultInstance =
((Settings)(global::System.Configuration.ApplicationSettingsBase.Synchronized(new
Settings())));
20:     }
21: }
```

Namespaces

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

Index

A

B

E

F