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
◆ S	Enums (↗ see page 4)	This class contains enums for detailed information In usage
◆ S	ExceptionExtensions (↗ see page 5)	This class is used to take generic exception messages With the inner exception, stack trace and source informations
◆	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.

1.1.1 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 55)

Description

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

Methods

	Name	Description
	Resources (see page 2)	This is Resources, a member of class Resources.

Resources Fields

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

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

1.1.1.1.1.2 Resources Fields

1.1.1.1.1.2.1 Resources.resourceCulture Field

C#

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

Description

This is resourceCulture, a member of class Resources.

1.1.1.1.1.2.2 Resources.resourceMan Field

C#

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

Description

This is resourceMan, a member of class Resources.

1.1.1.1.1.3 Resources Properties

1.1.1.1.1.3.1 Resources.AppTitle Property

Looks up a localized string similar to PDF Birlestirici.

C#

```
internal static string AppTitle;
```

1.1.1.1.1.3.2 Resources.Culture Property

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

C#

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

1.1.1.1.1.3.3 Resources.ResourceManager Property

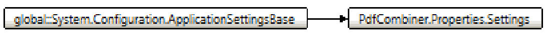
Returns the cached ResourceManager instance used by this class.

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 56)

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
S	Enums (see page 4)	This class contains enums for detailed information In usage
S	ExceptionExtensions (see page 5)	This class is used to take generic exception messages With the inner exception, stack trace and source informations
	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 Enums Class

This class contains enums for detailed information In usage

Class Hierarchy

PdfCombiner.Enums

C#

```
public static class Enums;
```

File

Enums.cs (see page 35)

Enums Enumerations

	Name	Description
	OrderType (see page 5)	This enum is used for Assigning the order type in listbox items Ascending or Descending

	SortType (see page 5)	This enum is used for Assigning the sort type in listbox items By Name or By Path

1.1.2.1.1 Enums Enumerations

1.1.2.1.1.1 PdfCombiner.Enums.OrderType Enumeration

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

C#

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

File

Enums.cs (see page 35)

1.1.2.1.1.2 PdfCombiner.Enums.SortType Enumeration

This enum is used for Assigning the sort type in listbox items By Name or By Path

C#

```
[Flags]
public enum SortType {
  ByName,
  ByPath
}
```

File

Enums.cs (see page 35)

1.1.2.2 ExceptionExtensions Class

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

Class Hierarchy

C#

```
public static class ExceptionExtensions;
```

File

ExceptionExtensions.cs (see page 36)

ExceptionExtensions Methods

	Name	Description
	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#

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

Parameters

Parameters	Description
this Exception exception	Exception

Returns

Detailed Error Message

Body Source

```
public static string GetAllMessages(this Exception exception)
{
    var sb = new StringBuilder();
    sb.AppendLine(exception.Message);
    if (exception.InnerException != null)
        sb.AppendLine($" {exception.InnerException.GetAllMessages()}");
    if (!string.IsNullOrEmpty(exception.StackTrace))
        sb.AppendLine($" {exception.StackTrace}");
    if (!string.IsNullOrEmpty(exception.Source))
        sb.AppendLine($" {exception.Source}");
    return sb.ToString();
}
```

1.1.2.3 FileInfo Class

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

Class Hierarchy

C#

```
public class FileInfo;
```

File

FileInfo.cs (see page 36)

FileInfo Properties

	Name	Description
	FileName (see page 6)	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 50)

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 27)	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 30)	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	SortItems (see page 32)	This method is used to order items by full path in listbox And return them with the given order as parametre
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 33)	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

```
public FrmMain()
{
    InitializeComponent();
}
```

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

```
private static void AddPdfContentToTargetPdfFileITextSharp(object t, PdfCopy pdfWriter,
ref int combinedFiles)
{
    var pdfReader =
        new iTextSharp.text.pdf.PdfReader(t.ToString());
    pdfReader.ConsolidateNamedDestinations();
    for (var j = 1; j <= pdfReader.NumberOfPages; j++)
    {
        var page = pdfWriter.GetImportedPage(pdfReader, j);
        pdfWriter.AddPage(page);
    }

    pdfReader.Close();
    combinedFiles++;
}
```

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

```
private static void AddPdfContentToTargetPdfFilePdfSharp(object t, PdfDocument
outputFile, ref int combinedFiles)
{
    var inputDocument = PdfReader.Open(t.ToString(),
        PdfDocumentOpenMode.Import);
    var count = inputDocument.PageCount;
    for (var index = 0; index < count; index++)
    {
        var page = inputDocument.Pages[index];
        outputFile.AddPage(page);
    }

    inputDocument.Close();
    combinedFiles++;
}
```

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

```
private void AddPdfFilesToList(ICollection<string> fileNames, ref int addedFileCount)
{
    foreach (var file in fileNames)
    {
        if (lbFiles.Items.Contains(file)) continue;
        lbFiles.Items.Add(file);
        addedFileCount++;

        pbFiles.Value = addedFileCount * 100 / fileNames.Count;
        if (pbFiles.Value > pbFiles.Maximum)
            pbFiles.Value = pbFiles.Maximum;
        if (ActiveForm != null) ActiveForm.Text = @"%" + pbFiles.Value;
    }
}
```

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

```
private void btnAddFile_Click(object sender, EventArgs e)
{
    using (var dialogAddFile = new OpenFileDialog())
    {
        InitializeFileDialog(dialogAddFile);

        var result = dialogAddFile.ShowDialog();
        var addedFileCount = 0;
        if (result != DialogResult.OK || dialogAddFile.FileNames == null) return;

        AddPdfFilesToList(dialogAddFile.FileNames, ref addedFileCount);
        GenerateAddFileMessage(addedFileCount);
    }
}
```

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

```
private void btnAddFolder_Click(object sender, EventArgs e)
{
    using (var dialogAddFolder = new FolderBrowserDialog())
    {
        var result = dialogAddFolder.ShowDialog();
        if (result != DialogResult.OK ||
string.IsNullOrEmpty(dialogAddFolder.SelectedPath)) return;
        var fileNames = Directory.GetFiles(dialogAddFolder.SelectedPath, "*.pdf",
            SearchOption.AllDirectories);
        var addedFileCount = 0;

        AddPdfFilesToList(fileNames, ref addedFileCount);
        GenerateAddFileMessage(addedFileCount);
    }
}
```

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

```
private void btnClearList_Click(object sender, EventArgs e)
{
    var fileCount = lbFiles.Items.Count;
    lbFiles.Items.Clear();
    GenerateDeleteItemMessage(fileCount);
}
```

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

```
private void btnCombineITextSharp_Click(object sender, EventArgs e)
{
    try
    {
        if (lbFiles.Items.Count < 2)
            GenerateCombineWarningMessage();
        else
        {
            using (var dialogExport = new FolderBrowserDialog())
            {
                var result = dialogExport.ShowDialog();
                if (result != DialogResult.OK || string.IsNullOrEmpty(dialogExport.SelectedPath)) return;

                SetInitialValuesForCombiningFiles(dialogExport, out var outputFileName, out var fileCount,
                    out var combinedFiles);

                var outputFile = new Document();
                using (var outputFileStream = new FileStream(outputFileName, FileMode.Create))
                {
                    using (var pdfWriter = new PdfCopy(outputFile, outputFileStream))
                    {
                        pdfWriter.SetFullCompression();
                        outputFile.Open();
                        foreach (var t in lbFiles.Items)
                        {
                            try
                            {
                                AddPdfContentToTargetPdfFileITextSharp(t, pdfWriter, ref combinedFiles);
                            }
                            catch (Exception)
                            {
                                fileCount--;
                            }

                            SetCombinationRatio(combinedFiles, fileCount);
                        }

                        pdfWriter.Close();
                    }

                    outputFile.Close();
                }

                GenerateCombineFileMessage(fileCount, outputFileName);
            }
        }
    }
    catch (Exception ex)
    {
        GenerateExceptionMessage(ex);
    }
}
```

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

```
private void btnCombinePdfSharp_Click(object sender, EventArgs e)
{
    try
    {
        if (lbFiles.Items.Count < 2)
            GenerateCombineWarningMessage();
        else
        {
            using (var dialogExport = new FolderBrowserDialog())
            {
                var result = dialogExport.ShowDialog();
                if (result != DialogResult.OK || string.IsNullOrEmpty(dialogExport.SelectedPath)) return;

                SetInitialValuesForCombiningFiles(dialogExport, out var outputFileName, out var fileCount,
                    out var combinedFiles);

                using (var outputFile = new PdfDocument())
                {
                    SetOutputFilePropertiesPdfSharp(outputFile);

                    foreach (var t in lbFiles.Items)
                    {
                        try
                        {
                            AddPdfContentToTargetPdfFilePdfSharp(t, outputFile, ref combinedFiles);
                        }
                        catch (Exception)
                        {
                            fileCount--;
                        }

                        SetCombinationRatio(combinedFiles, fileCount);
                    }

                    SaveOutputPdfFile(outputFile, outputFileName);
                }

                GenerateCombineFileMessage(fileCount, outputFileName);
            }
        }
    }
    catch (Exception ex)
    {
        GenerateExceptionMessage(ex);
    }
}
```

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

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

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

```
private void DeleteFilesFromListBox()
{
    var fileCount = lbFiles.SelectedItems.Count;
    var removedItems = (from object t in lbFiles.SelectedItems select
t.ToString()).ToList();

    var allItems = (from object item in lbFiles.Items where removedItems.All(j => j !=
item.ToString()) select item.ToString()).ToList();

    lbFiles.Items.Clear();
    foreach (var item in allItems)
    {
        lbFiles.Items.Add(item);
    }

    GenerateDeleteItemMessage(fileCount);
}
```

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

```
protected override void Dispose(bool disposing)
{
    if (disposing && (components != null))
    {
        components.Dispose();
    }
    base.Dispose(disposing);
}
```

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

```
private void FillLanguageComboboxAndSetFirstLanguage(out string firstLanguage)
{
    cmbLanguage.DropDownStyle = ComboBoxStyle.DropDownList;
    cmbLanguage.Items.Clear();
    var cultureInfo = CultureInfo.InstalledUICulture;
    firstLanguage = string.Empty;
    foreach (var item in LanguageList)
    {
        if (cultureInfo.Name.Contains(item))
            firstLanguage = item.ToLower(new CultureInfo("en-US"));
        cmbLanguage.Items.Add(item);
    }

    if (string.IsNullOrEmpty(firstLanguage))
        firstLanguage = "tr";
}
```

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

```
private void FormMembersNameInitialization()
{
    if (_resource == null) return;
    if (ActiveForm != null) ActiveForm.Text = _resource.GetString("AppTitle") ?? string.Empty;
    btnAddFile.Text = _resource.GetString("AddFile") ?? string.Empty;
    btnAddFolder.Text = _resource.GetString("AddFolder") ?? string.Empty;
    btnClearList.Text = _resource.GetString("ClearFileList") ?? string.Empty;
    btnCombineITextSharp.Text = _resource.GetString("CombineFilesITextSharp") ?? string.Empty;
    btnCombinePdfSharp.Text = _resource.GetString("CombineFilesPdfSharp") ?? string.Empty;
    menuItemDelete.Text = _resource.GetString("Delete") ?? string.Empty;
    menuItemOrderByPathAscending.Text = _resource.GetString("OrderByPathAscending") ?? string.Empty;
    menuItemOrderByPathDescending.Text = _resource.GetString("OrderByPathDescending") ?? string.Empty;
    menuItemOrderByNameAscending.Text = _resource.GetString("OrderByNameAscending") ?? string.Empty;
    menuItemOrderByNameDescending.Text = _resource.GetString("OrderByNameDescending") ?? string.Empty;
    cmbLanguage.Text = _resource.GetString("SelectLanguage") ?? string.Empty;
}
```

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

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

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

```
private void FrmMain_Shown(object sender, EventArgs e)
{
    lbFiles.ContextMenuStrip = menuStrip;

    FillLanguageComboboxAndSetFirstLanguage(out var firstLanguage);

    _resource = new ResourceManager("PdfCombiner.Resources.AppResources-" +
firstLanguage,
        Assembly.GetExecutingAssembly());
    cmbLanguage.SelectedIndex = cmbLanguage.FindStringExact(firstLanguage.ToUpper(new
CultureInfo("en-US")));

    FormMembersNameInitialization();
}
```

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

```
private void GenerateAddFileMessage(int addedFileCount)
{
    MessageBox.Show(addedFileCount + _resource.GetString("FileAddMessage"),
        _resource.GetString("AppTitle"), MessageBoxButtons.OK,
MessageBoxIcon.Information);

    pbFiles.Value = pbFiles.Minimum;
    if (ActiveForm != null) ActiveForm.Text = _resource.GetString("AppTitle");
}
```

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

```
private void GenerateCombineFileMessage(int fileCount, string outputFileName)
{
    MessageBox.Show(fileCount + _resource.GetString("CombineFileMessage") + outputFileName, _resource.GetString("AppTitle"), MessageBoxButtons.OK, MessageBoxIcon.Information);
    pbFiles.Value = pbFiles.Minimum;
    if (ActiveForm != null) ActiveForm.Text = _resource.GetString("AppTitle");
}
```

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

```
private static void GenerateCombineWarningMessage()
{
    MessageBox.Show(_resource.GetString("CombineWarning"), _resource.GetString("AppTitle"), MessageBoxButtons.OK, MessageBoxIcon.Error);
}
```

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

```
private static void GenerateDeleteItemMessage(int fileCount)
{
    MessageBox.Show(fileCount + _resource.GetString("DeleteFileMessage"), _resource.GetString("AppTitle"), MessageBoxButtons.OK, MessageBoxIcon.Information);
}
```

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

```
private void GenerateExceptionMessage(Exception ex)
{
    MessageBox.Show(_resource.GetString("CombineErrorMessage") + ex.GetAllMessages(),
_resource.GetString("AppTitle"), MessageBoxButtons.OK, MessageBoxIcon.Error);
    pbFiles.Value = pbFiles.Minimum;
    if (ActiveForm != null) ActiveForm.Text = _resource.GetString("AppTitle");
}
```

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

```
private static void GenerateNoFileInListBoxMessage()
{
    MessageBox.Show(_resource.GetString("NoFile"), _resource.GetString("AppTitle"),
MessageBoxButtons.OK, MessageBoxIcon.Error);
}
```

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

```
private static void GenerateNoSelectedFileInListBoxMessage()
{
    MessageBox.Show(_resource.GetString("NoSelectedFile"),
_resource.GetString("AppTitle"),
        MessageBoxButtons.OK, MessageBoxIcon.Error);
}
```

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

```
private void InitializeComponent()
{
    this.components = new System.ComponentModel.Container();
    System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(FrmMain));
    this.btnCombinePdfSharp = new System.Windows.Forms.Button();
    this.btnAddFolder = new System.Windows.Forms.Button();
    this.btnAddFile = new System.Windows.Forms.Button();
    this.btnClearList = new System.Windows.Forms.Button();
    this.lbFiles = new System.Windows.Forms.ListBox();
    this.lblDetails = new System.Windows.Forms.Label();
    this.pbFiles = new System.Windows.Forms.ProgressBar();
    this.btnCombineITextSharp = new System.Windows.Forms.Button();
    this.menuStrip = new System.Windows.Forms.ContextMenuStrip(this.components);
    this.menuItemDelete = new System.Windows.Forms.ToolStripMenuItem();
    this.menuItemOrderByPathAscending = new System.Windows.Forms.ToolStripMenuItem();
    this.menuItemOrderByPathDescending = new System.Windows.Forms.ToolStripMenuItem();
    this.menuItemOrderByNameAscending = new System.Windows.Forms.ToolStripMenuItem();
    this.menuItemOrderByNameDescending = new System.Windows.Forms.ToolStripMenuItem();
    this.cmbLanguage = new System.Windows.Forms.ComboBox();
    this.menuStrip.SuspendLayout();
    this.SuspendLayout();
    //
    // btnCombinePdfSharp
    //
    this.btnCombinePdfSharp.Image = ((System.Drawing.Image)(resources.GetObject("btnCombinePdfSharp.Image")));
    this.btnCombinePdfSharp.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
    this.btnCombinePdfSharp.Location = new System.Drawing.Point(442, 5);
    this.btnCombinePdfSharp.Name = "btnCombinePdfSharp";
    this.btnCombinePdfSharp.Size = new System.Drawing.Size(214, 37);
    this.btnCombinePdfSharp.TabIndex = 0;
    this.btnCombinePdfSharp.Text = "Combine Files (PdfSharp)";
    this.btnCombinePdfSharp.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
    this.btnCombinePdfSharp.UseVisualStyleBackColor = true;
    this.btnCombinePdfSharp.Click += new System.EventHandler(this.btnCombinePdfSharp_Click);
    //
    // btnAddFolder
    //
    this.btnAddFolder.Image = ((System.Drawing.Image)(resources.GetObject("btnAddFolder.Image")));
    this.btnAddFolder.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
    this.btnAddFolder.Location = new System.Drawing.Point(197, 5);
    this.btnAddFolder.Name = "btnAddFolder";
    this.btnAddFolder.Size = new System.Drawing.Size(171, 37);
    this.btnAddFolder.TabIndex = 2;
    this.btnAddFolder.Text = "Add Folder";
    this.btnAddFolder.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
    this.btnAddFolder.UseVisualStyleBackColor = true;
    this.btnAddFolder.Click += new System.EventHandler(this.btnAddFolder_Click);
    //
    // btnAddFile
    //
    this.btnAddFile.Image = ((System.Drawing.Image)(resources.GetObject("btnAddFile.Image")));
    this.btnAddFile.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
    this.btnAddFile.Location = new System.Drawing.Point(12, 5);
    this.btnAddFile.Name = "btnAddFile";
    this.btnAddFile.Size = new System.Drawing.Size(179, 37);
    this.btnAddFile.TabIndex = 3;
    this.btnAddFile.Text = "Add Files";
    this.btnAddFile.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
    this.btnAddFile.UseVisualStyleBackColor = true;
    this.btnAddFile.Click += new System.EventHandler(this.btnAddFile_Click);
    //
    // btnClearList
    //
    this.btnClearList.Image = ((System.Drawing.Image)(resources.GetObject("btnClearList.Image")));
    this.btnClearList.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
    this.btnClearList.Location = new System.Drawing.Point(12, 45);
    this.btnClearList.Name = "btnClearList";
    this.btnClearList.Size = new System.Drawing.Size(179, 37);
    this.btnClearList.TabIndex = 4;
    this.btnClearList.Text = "Clear File List";
    this.btnClearList.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
    this.btnClearList.UseVisualStyleBackColor = true;
    this.btnClearList.Click += new System.EventHandler(this.btnClearList_Click);
    //
    // lbFiles
    //
    this.lbFiles.AllowDrop = true;
    this.lbFiles.FormattingEnabled = true;
    this.lbFiles.Location = new System.Drawing.Point(12, 88);
    this.lbFiles.Name = "lbFiles";
    this.lbFiles.SelectionMode = System.Windows.Forms.SelectionMode.MultiExtended;
```

```
this.lbFiles.Size = new System.Drawing.Size(644, 186);
this.lbFiles.TabIndex = 5;
this.lbFiles.DragDrop += new System.Windows.Forms.DragEventHandler(this.lbFiles_DragDrop);
this.lbFiles.DragOver += new System.Windows.Forms.DragEventHandler(this.lbFiles_DragOver);
this.lbFiles.KeyDown += new System.Windows.Forms.KeyEventHandler(this.lbFiles_KeyDown);
this.lbFiles.MouseDown += new System.Windows.Forms.MouseEventHandler(this.lbFiles_MouseDown);
//
// lblDetails
//
this.lblDetails.AutoSize = true;
this.lblDetails.Location = new System.Drawing.Point(12, 65);
this.lblDetails.Name = "lblDetails";
this.lblDetails.Size = new System.Drawing.Size(0, 13);
this.lblDetails.TabIndex = 6;
//
// pbFiles
//
this.pbFiles.Location = new System.Drawing.Point(197, 45);
this.pbFiles.Name = "pbFiles";
this.pbFiles.Size = new System.Drawing.Size(239, 37);
this.pbFiles.TabIndex = 7;
//
// btnCombineITextSharp
//
this.btnCombineITextSharp.Image = ((System.Drawing.Image)(resources.GetObject("btnCombineITextSharp.Image")));
this.btnCombineITextSharp.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
this.btnCombineITextSharp.Location = new System.Drawing.Point(442, 45);
this.btnCombineITextSharp.Name = "btnCombineITextSharp";
this.btnCombineITextSharp.Size = new System.Drawing.Size(214, 37);
this.btnCombineITextSharp.TabIndex = 9;
this.btnCombineITextSharp.Text = "Combine Files (iTextSharp)";
this.btnCombineITextSharp.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
this.btnCombineITextSharp.UseVisualStyleBackColor = true;
this.btnCombineITextSharp.Click += new System.EventHandler(this.btnCombineITextSharp_Click);
//
// menuStrip
//
this.menuStrip.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {this.menuItemDelete, this.menuItemOrderByPathAscending, this.menuItemOrderByPathDescending, this.menuItemOrderByNameAscending, this.menuItemOrderByNameDescending});
this.menuStrip.Name = "menuStrip";
this.menuStrip.Size = new System.Drawing.Size(229, 114);
//
// menuItemDelete
//
this.menuItemDelete.Image = ((System.Drawing.Image)(resources.GetObject("menuItemDelete.Image")));
this.menuItemDelete.Name = "menuItemDelete";
this.menuItemDelete.Size = new System.Drawing.Size(228, 22);
this.menuItemDelete.Text = "Delete";
this.menuItemDelete.Click += new System.EventHandler(this.menuItemDelete_Click);
//
// menuItemOrderByPathAscending
//
this.menuItemOrderByPathAscending.Image = ((System.Drawing.Image)(resources.GetObject("menuItemOrderByPathAscending.Image")));
this.menuItemOrderByPathAscending.Name = "menuItemOrderByPathAscending";
this.menuItemOrderByPathAscending.Size = new System.Drawing.Size(228, 22);
this.menuItemOrderByPathAscending.Text = "Order By Path (Ascending)";
this.menuItemOrderByPathAscending.Click += new System.EventHandler(this.menuItemOrderByPathAscending_Click);
//
```

```
      // menuItemOrderByPathDescending
      //
      this.menuItemOrderByPathDescending.Image = ((System.Drawing.Image)(resources.GetObject("menuItemOrderByPathDescending.Image")));
      this.menuItemOrderByPathDescending.Name = "menuItemOrderByPathDescending";
      this.menuItemOrderByPathDescending.Size = new System.Drawing.Size(228, 22);
      this.menuItemOrderByPathDescending.Text = "Order By Path (Descending)";
      this.menuItemOrderByPathDescending.Click += new System.EventHandler(this.menuItemOrderByPathDescending_Click);
      //
      // menuItemOrderByNameAscending
      //
      this.menuItemOrderByNameAscending.Image = ((System.Drawing.Image)(resources.GetObject("menuItemOrderByNameAscending.Image")));
      this.menuItemOrderByNameAscending.Name = "menuItemOrderByNameAscending";
      this.menuItemOrderByNameAscending.Size = new System.Drawing.Size(228, 22);
      this.menuItemOrderByNameAscending.Text = "Order By Name (Ascending)";
      this.menuItemOrderByNameAscending.Click += new System.EventHandler(this.menuItemOrderByNameAscending_Click);
      //
      // menuItemOrderByNameDescending
      //
      this.menuItemOrderByNameDescending.Image = ((System.Drawing.Image)(resources.GetObject("menuItemOrderByNameDescending.Image")));
      this.menuItemOrderByNameDescending.Name = "menuItemOrderByNameDescending";
      this.menuItemOrderByNameDescending.Size = new System.Drawing.Size(228, 22);
      this.menuItemOrderByNameDescending.Text = "Order By Name (Descending)";
      this.menuItemOrderByNameDescending.Click += new System.EventHandler(this.menuItemOrderByNameDescending_Click);
      //
      // cmbLanguage
      //
      this.cmbLanguage.Anchor = System.Windows.Forms.AnchorStyles.None;
      this.cmbLanguage.Font = new System.Drawing.Font("Microsoft Sans Serif", 15F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte) (162)));
      this.cmbLanguage.FormattingEnabled = true;
      this.cmbLanguage.Items.AddRange(new object[] {"TR", "EN", "DE", "FR", "RU", "ES"});
      this.cmbLanguage.Location = new System.Drawing.Point(374, 6);
      this.cmbLanguage.Name = "cmbLanguage";
      this.cmbLanguage.Size = new System.Drawing.Size(62, 33);
      this.cmbLanguage.TabIndex = 10;
      this.cmbLanguage.Text = "Select Language";
      this.cmbLanguage.SelectedIndexChanged += new System.EventHandler(this.cmbLanguage_SelectedIndexChanged);
      //
      // FrmMain
      //
      this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
      this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
      this.ClientSize = new System.Drawing.Size(668, 287);
      this.Controls.Add(this.cmbLanguage);
      this.Controls.Add(this.btnCombineITextSharp);
      this.Controls.Add(this.pbFiles);
      this.Controls.Add(this.lblDetails);
      this.Controls.Add(this.lbFiles);
      this.Controls.Add(this.btnClearList);
      this.Controls.Add(this.btnAddFile);
      this.Controls.Add(this.btnAddFolder);
      this.Controls.Add(this.btnCombinePdfSharp);
      this.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte) (162)));
      this.Name = "FrmMain";
      this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
      this.Text = "PDF Combiner";
      this.FormClosed += new System.Windows.Forms.FormClosedEventHandler(this.FrmMain_FormClosed);
      this.Shown += new System.EventHandler(this.FrmMain_Shown);
      this.menuStrip.ResumeLayout(false);
      this.ResumeLayout(false);
```

```
    this.PerformLayout();
}
```

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

```
private static void InitializeFileDialog(OpenFileDialog dialogAddFile)
{
    dialogAddFile.Multiselect = true;
    dialogAddFile.Filter = _resource.GetString("PdfFiles") + @" (*.pdf)|*.pdf";
    dialogAddFile.InitialDirectory = Application.StartupPath;
    dialogAddFile.Title = _resource.GetString("SelectPdfFile");
    dialogAddFile.DefaultExt = "PDF";
}
```

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

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

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

```
private void lbFiles_DragOver(object sender, DragEventArgs e)
{
    e.Effect = DragDropEffects.Move;
}
```

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

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

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

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

1.1.2.4.3.29 FrmMain.menuItemDelete_Click Method

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

C#

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

Parameters

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

Body Source

```
private void menuItemDelete_Click(object sender, EventArgs e)
{
    if (lbFiles.Items.Count > 0)
    {
        if (lbFiles.SelectedItems.Count > 0)
        {
            ShowDeleteDialog(out var deleteDialog);
            if (deleteDialog == DialogResult.Yes)
                DeleteFilesFromListBox();
        }
        else
            GenerateNoSelectedFileInListBoxMessage();
    }
    else
        GenerateNoFileInListBoxMessage();
}
```

1.1.2.4.3.30 FrmMain.menuItemOrderByNameAscending_Click Method

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

C#

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

Parameters

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

Body Source

```
private void menuItemOrderByNameAscending_Click(object sender, EventArgs e)
{
    if (lbFiles.Items.Count > 0)
        lbFiles = SortItems(lbFiles, SortType.ByName, OrderType.Ascending);
        // lbFiles = SortItemsByName(lbFiles, OrderType.Ascending);
    else
        GenerateNoFileInListBoxMessage();
}
```

1.1.2.4.3.31 FrmMain.menuItemOrderByNameDescending_Click Method

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

C#

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

Parameters

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

Body Source

```
private void menuItemOrderByNameDescending_Click(object sender, EventArgs e)
{
    if (lbFiles.Items.Count > 0)
        lbFiles = SortItems(lbFiles, SortType.ByName, OrderType.Descending);
        // lbFiles = SortItemsByName(lbFiles, OrderType.Descending);
    else
        GenerateNoFileInListBoxMessage();
}
```

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

```
private void menuItemOrderByPathAscending_Click(object sender, EventArgs e)
{
    if (lbFiles.Items.Count > 0)
        lbFiles = SortItems(lbFiles, SortType.ByPath, OrderType.Ascending);
        // lbFiles = SortItemsByPath(lbFiles, OrderType.Ascending);
    else
        GenerateNoFileInListBoxMessage();
}
```

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

```
private void menuItemOrderByPathDescending_Click(object sender, EventArgs e)
{
    if (lbFiles.Items.Count > 0)
        lbFiles = SortItems(lbFiles, SortType.ByPath, OrderType.Descending);
        // lbFiles = SortItemsByPath(lbFiles, OrderType.Descending);
    else
        GenerateNoFileInListBoxMessage();
}
```

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

```
private static void SaveOutputPdfFile(PdfDocument outputFile, string outputFileName)
{
    outputFile.Save(outputFileName);
    outputFile.Close();
}
```

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

```
private void SetCombinationRatio(int combinedFiles, int fileCount)
{
    pbFiles.Value = combinedFiles * 100 / fileCount;
    if (pbFiles.Value > pbFiles.Maximum)
        pbFiles.Value = pbFiles.Maximum;
    if (ActiveForm != null) ActiveForm.Text = @"%" + pbFiles.Value;
}
```

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

```
private void SetInitialValuesForCombiningFiles(FolderBrowserDialog dialogExport, out
string outputFileName,
    out int fileCount, out int combinedFiles)
{
    outputFileName = dialogExport.SelectedPath + "/" + Guid.NewGuid() + ".pdf";
    fileCount = lbFiles.Items.Count;
    combinedFiles = 0;
}
```

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

```
private static void SetOutputFilePropertiesPdfSharp(PdfDocument outputFile)
{
    outputFile.Options.CompressContentStreams = true;
    outputFile.Options.EnableCcittCompressionForBilevelImages = true;
    outputFile.Options.FlateEncodeMode = PdfFlateEncodeMode.BestCompression;
}
```

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

```
private void SetProgramLanguage()
{
    try
    {
        _resource = new ResourceManager(
            "PdfCombiner.Resources.AppResources-" +
            cmbLanguage.SelectedItem.ToString().ToLower(new CultureInfo("en-US")),
            Assembly.GetExecutingAssembly());
    }
    catch (Exception)
    {
        _resource = new ResourceManager("PdfCombiner.Resources.AppResources-en",
            Assembly.GetExecutingAssembly());
    }
    finally
    {
        FormMembersNameInitialization();
    }
}
```

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

```
private void ShowDeleteDialog(out DialogResult deleteDialog)
{
    deleteDialog = MessageBox.Show( _resource.GetString("DeleteWarning1") +
lbFiles.SelectedItems.Count + _resource.GetString("DeleteWarning2"),
_resource.GetString("AppTitle"), MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
}
```

1.1.2.4.3.40 FrmMain.SortItems 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 SortItems(ListBox listBox, SortType sortType, OrderType orderType);
```

Parameters

Parameters	Description
ListBox listBox	ListBox Info
SortType sortType	Sort Type (By Name or Path)
OrderType orderType	Order Type is Ascending Or Not

Body Source

```
private static ListBox SortItems(ListBox listBox, SortType sortType, OrderType orderType)
{
    var items = listBox.Items.OfType<object>().ToList();
    listBox.Items.Clear();

    // Sort Items By Name
    if (sortType == SortType.ByName)
    {
        listBox.Items.AddRange(orderType == OrderType.Ascending
            ? items.OrderBy(i => i).ToArray()
            : items.OrderByDescending(i => i).ToArray());
    }
    // Sort Items By Path
    else
    {
        const char c = '\u005c';
        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();

        fileList = orderType == OrderType.Ascending
            ? fileList.OrderBy(j => j.FileName).ThenBy(j => j.FilePath).ToList()
            : fileList.OrderByDescending(j => j.FileName).ThenByDescending(j => j.FilePath).ToList();

        foreach (var item in fileList)
            listBox.Items.Add(item.FilePath);
    }

    return listBox;
}
```

1.1.2.4.3.41 FrmMain.SortItemsByName Method

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

C#

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

Parameters

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

Body Source

```
private static ListBox SortItemsByName(ListBox listBox, OrderType orderType)
{
    var items = listBox.Items.OfType<object>().ToList();
    const char c = '\u005c';
    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();

    fileList = orderType == OrderType.Ascending
        ? fileList.OrderBy(j => j.FileName).ThenBy(j => j.FilePath).ToList()
        : fileList.OrderByDescending(j => j.FileName).ThenByDescending(j => j.FilePath).ToList();

    listBox.Items.Clear();
    foreach (var item in fileList)
        listBox.Items.Add(item.FilePath);

    return listBox;
}
```

1.1.2.4.3.42 FrmMain.SortItemsByPath Method

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

C#

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

Parameters

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

Body Source

```
private static ListBox SortItemsByPath(ListBox listBox, OrderType orderType)
{
    var items = listBox.Items.OfType<object>().ToList();
    listBox.Items.Clear();
    listBox.Items.AddRange(orderType == OrderType.Ascending
        ? items.OrderBy(i => i).ToArray()
        : items.OrderByDescending(i => i).ToArray());
    return listBox;
}
```

1.1.2.5 Program Class

Class Hierarchy

C#

```
internal static class Program;
```

File

Program.cs (see page 54)

Description

This is class PdfCombiner.Program.

Program Methods

	Name	Description
S	Main (see page 34)	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

```
[STAThread]
private static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new FrmMain());
}
```

1.2 Files

The following table lists files in this documentation.

Files

Name	Description
AssemblyInfo.cs (see page 34)	This is file AssemblyInfo.cs.
Enums.cs (see page 35)	This is file Enums.cs.
ExceptionExtensions.cs (see page 36)	This is file ExceptionExtensions.cs.
FileInfo.cs (see page 36)	This is file FileInfo.cs.
FrmMain.cs (see page 37)	This is file FrmMain.cs.
FrmMain.Designer.cs (see page 50)	This is file FrmMain.Designer.cs.
PdfCombiner.csproj (see page 54)	This is file PdfCombiner.csproj.
PdfCombiner.sln (see page 54)	This is file PdfCombiner.sln.
Program.cs (see page 54)	This is file Program.cs.
Resources.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.
Settings.Designer.cs (see page 56)	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

```
?using System.Reflection;
using System.Runtime.InteropServices;

```

```
// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
[assembly: AssemblyTitle("PdfCombiner")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("PdfCombiner")]
[assembly: AssemblyCopyright("Copyright ©  2022")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the types in this assembly not visible
// to COM components.  If you need to access a type in this assembly from
// COM, set the ComVisible attribute to true on that type.
[assembly: ComVisible(false)]

// The following GUID is for the ID of the typelib if this project is exposed to COM
[assembly: Guid("48e34ee8-dfaa-48db-acfd-70573cee9c00")]

// Version information for an assembly consists of the following four values:
//
//      Major Version
//      Minor Version
//      Build Number
//      Revision
//
// You can specify all the values or you can default the Build and Revision Numbers
// by using the '*' as shown below:
// [assembly: AssemblyVersion("1.0.*")]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
```

1.2.2 Enums.cs

This is file Enums.cs.

Body Source

```
?using System;

namespace PdfCombiner
{
    /// <summary>
    /// This class contains enums for detailed information
    /// In usage
    /// </summary>
    public static class Enums
    {
        /// <summary>
        /// This enum is used for
        /// Assigning the order type in listbox items
        /// Ascending or Descending
        /// </summary>
        [Flags]
        public enum OrderType
        {
            Ascending,
            Descending
        }

        /// <summary>
        /// This enum is used for
        /// Assigning the sort type in listbox items
        /// By Name or By Path
        /// </summary>
        [Flags]
        public enum SortType
        {
            ByName,
            ByPath
        }
    }
}
```

Namespaces

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

1.2.3 ExceptionExtensions.cs

This is file ExceptionExtensions.cs.

Body Source

```
?using System;
using System.Text;

namespace PdfCombiner
{
    /// <summary>
    /// This class is used to take generic exception messages
    /// With the inner exception, stack trace and source informations
    /// </summary>
    public static class ExceptionExtensions
    {
        /// <summary>
        /// This method is used to take all messages with stack trace info in exception
        /// </summary>
        /// <param name="exception">Exception</param>
        /// <returns>Detailed Error Message</returns>
        public static string GetAllMessages(this Exception exception)
        {
            var sb = new StringBuilder();
            sb.AppendLine(exception.Message);
            if (exception.InnerException != null)
                sb.AppendLine($" {exception.InnerException.GetAllMessages()}");
            if (!string.IsNullOrEmpty(exception.StackTrace))
                sb.AppendLine($" {exception.StackTrace}");
            if (!string.IsNullOrEmpty(exception.Source))
                sb.AppendLine($" {exception.Source}");
            return sb.ToString();
        }
    }
}
```

Namespaces

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

1.2.4 FileInfo.cs

This is file FileInfo.cs.

Body Source

```
?namespace PdfCombiner
{
    /// <summary>
    /// This class is used to have File's Full Path and Name
    /// </summary>
    public class FileInfo
    {
        /// <summary>
        /// File Path Property
        /// </summary>
        public string FilePath { get; set; }
        /// <summary>
        /// File Name Property
        /// </summary>
        public string FileName { get; set; }
    }
}
```

Namespaces

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

1.2.5 FrmMain.cs

This is file FrmMain.cs.

Body Source

```
?using iTextSharp.text;
using PdfSharp.Pdf;
using PdfSharp.Pdf.IO;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Resources;
using System.Windows.Forms;
using iTextSharp.text.pdf;
using PdfDocument = PdfSharp.Pdf.PdfDocument;
using PdfReader = PdfSharp.Pdf.IO.PdfReader;
using static PdfCombiner.Enums;

namespace PdfCombiner
{
    public partial class FrmMain : Form
    {
        private static ResourceManager _resource;

        // Culture lists whose resources have been added to project
        private static readonly List<string> LanguageList = new List<string>
            {"EN", "TR", "DE", "FR", "RU", "ES", "IT", "ZH", "AR", "NL", "PT", "IN",
"ID", "JP", "BG"};

        public FrmMain()
        {
            InitializeComponent();
        }

        #region Form Init, Language Settings and Finalize Methods
        /// <summary>
        /// This function is used to terminate application
        /// </summary>
        /// <param name="sender">The sender info (For example Main Form)</param>
        /// <param name="e">Event Arguments</param>
        private void FrmMain_FormClosed(object sender, FormClosedEventArgs e)
        {
            MessageBox.Show(_resource.GetString("ThanksMessage"), _resource.GetString("AppTitle"), MessageBoxButtons.OK, MessageBoxIcon.Information);
            Application.Exit();
        }

        /// <summary>
        /// This function is used to assign menu to listbox
        /// When form is shown
        /// </summary>
        /// <param name="sender">The sender info (For example Main Form)</param>
        /// <param name="e">Event Arguments</param>
        private void FrmMain_Shown(object sender, EventArgs e)
        {
            lbFiles.ContextMenuStrip = menuStrip;

            FillLanguageComboboxAndSetFirstLanguage(out var firstLanguage);

            _resource = new ResourceManager("PdfCombiner.Resources.AppResources-" + firstLanguage,
                Assembly.GetExecutingAssembly());
            cmbLanguage.SelectedIndex = cmbLanguage.FindStringExact(firstLanguage.ToUpper(new CultureInfo("en-US")));

            FormMembersNameInitialization();
        }

        /// <summary>
        /// This method is used to Fill the Language combobox
        /// And set initial value with
        /// The language computer used in
        /// </summary>
        /// <param name="firstLanguage">First Language Info</param>
        private void FillLanguageComboboxAndSetFirstLanguage(out string firstLanguage)
        {
            cmbLanguage.DropDownStyle = ComboBoxStyle.DropDownList;
            cmbLanguage.Items.Clear();
            var cultureInfo = CultureInfo.InstalledUICulture;
            firstLanguage = string.Empty;
            foreach (var item in LanguageList)
            {
                if (cultureInfo.Name.Contains(item))
                    firstLanguage = item.ToLower(new CultureInfo("en-US"));
                cmbLanguage.Items.Add(item);
            }

            if (string.IsNullOrEmpty(firstLanguage))
                firstLanguage = "tr";
        }

        /// <summary>
        /// This function is used to set the _resource file
        /// And with this, you can see form elements, info messages etc. with the selected
        /// _resource file contents. That is used to multi language support
        /// </summary>
        /// <param name="sender">The sender info (For example Main Form)</param>
        /// <param name="e">Event Arguments</param>
        private void cmbLanguage_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (cmbLanguage.SelectedIndex == -1) return;
            SetProgramLanguage();
        }

        /// <summary>
        /// This method is used to Set Program Language
        /// By the selection from Language Combobox
        /// Finally, with the selected language
        /// Form elements renamed with the selected language
        /// </summary>
        private void SetProgramLanguage()
        {
            try
            {
                _resource = new ResourceManager(
                    "PdfCombiner.Resources.AppResources-" +
                    cmbLanguage.SelectedItem.ToString().ToLower(new CultureInfo("en-US")),
                    Assembly.GetExecutingAssembly());
            }
            catch (Exception)
            {
                _resource = new ResourceManager("PdfCombiner.Resources.AppResources-en",
                    Assembly.GetExecutingAssembly());
            }
            finally
            {
                FormMembersNameInitialization();
            }
        }

        /// <summary>
        /// This method is used to change text of form members by the
        /// Resource file which is selected
        /// </summary>
        private void FormMembersNameInitialization()
        {
            if (_resource == null) return;
            if (ActiveForm != null) ActiveForm.Text = _resource.GetString("AppTitle") ?? string.Empty;
            btnAddFile.Text = _resource.GetString("AddFile") ?? string.Empty;
            btnAddFolder.Text = _resource.GetString("AddFolder") ?? string.Empty;
            btnClearList.Text = _resource.GetString("ClearFileList") ?? string.Empty;
            btnCombineITextSharp.Text = _resource.GetString("CombineFilesITextSharp") ?? string.Empty;
            btnCombinePdfSharp.Text = _resource.GetString("CombineFilesPdfSharp") ?? string.Empty;
            menuItemDelete.Text = _resource.GetString("Delete") ?? string.Empty;
            menuItemOrderByPathAscending.Text = _resource.GetString("OrderByPathAscending") ?? string.Empty;
            menuItemOrderByPathDescending.Text = _resource.GetString("OrderByPathDescending") ?? string.Empty;
            menuItemOrderByNameAscending.Text = _resource.GetString("OrderByNameAscending") ?? string.Empty;
            menuItemOrderByNameDescending.Text = _resource.GetString("OrderByNameDescending") ?? string.Empty;
            cmbLanguage.Text = _resource.GetString("SelectLanguage") ?? string.Empty;
        }

        #endregion

        #region Add Item Methods

        /// <summary>
        /// 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
        /// </summary>
        /// <param name="sender">The sender info (For example Main Form)</param>
        /// <param name="e">Event Arguments</param>
        private void btnAddFile_Click(object sender, EventArgs e)
        {
            using (var dialogAddFile = new OpenFileDialog())
            {
                InitializeFileDialog(dialogAddFile);

                var result = dialogAddFile.ShowDialog();
                var addedFileCount = 0;
                if (result != DialogResult.OK || dialogAddFile.FileNames == null) return;

                AddPdfFilesToList(dialogAddFile.FileNames, ref addedFileCount);
                GenerateAddFileMessage(addedFileCount);
            }
        }

        /// <summary>
        /// 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
        /// </summary>
        /// <param name="sender">The sender info (For example Main Form)</param>
        /// <param name="e">Event Arguments</param>
        private void btnAddFolder_Click(object sender, EventArgs e)
        {
            using (var dialogAddFolder = new FolderBrowserDialog())
            {
                var result = dialogAddFolder.ShowDialog();
                if (result != DialogResult.OK || string.IsNullOrEmpty(dialogAddFolder.SelectedPath)) return;
                var fileNames = Directory.GetFiles(dialogAddFolder.SelectedPath, "*.pdf",
                    SearchOption.AllDirectories);
                var addedFileCount = 0;

                AddPdfFilesToList(fileNames, ref addedFileCount);
                GenerateAddFileMessage(addedFileCount);
            }
        }

        /// <summary>
        /// This function is used to set options of file dialog like
        /// Filter, start path etc.
        /// </summary>
        /// <param name="dialogAddFile">File Dialog</param>
        private static void InitializeFileDialog(OpenFileDialog dialogAddFile)
        {
            dialogAddFile.Multiselect = true;
            dialogAddFile.Filter = _resource.GetString("PdfFiles") + @" (*.pdf)|*.pdf";
            dialogAddFile.InitialDirectory = Application.StartupPath;
            dialogAddFile.Title = _resource.GetString("SelectPdfFile");
            dialogAddFile.DefaultExt = "PDF";
        }

        /// <summary>
        /// This method is used both adding files or folders to list
        /// </summary>
        /// <param name="fileNames">List Of File Names</param>
        /// <param name="addedFileCount">Added File Count For Progress Bar</param>
        private void AddPdfFilesToList(ICollection<string> fileNames, ref int addedFileCount)
        {
            foreach (var file in fileNames)
            {
                if (lbFiles.Items.Contains(file)) continue;
                lbFiles.Items.Add(file);
                addedFileCount++;

```

1

```
                pbFiles.Value = addedFileCount * 100 / fileNames.Count;
                if (pbFiles.Value > pbFiles.Maximum)
                    pbFiles.Value = pbFiles.Maximum;
                if (ActiveForm != null) ActiveForm.Text = @"%" + pbFiles.Value;
            }
        }

        /// <summary>
        /// This method is used to show info message after adding files or folders
        /// </summary>
        /// <param name="addedFileCount">Added File Count</param>
        private void GenerateAddFileMessage(int addedFileCount)
        {
            MessageBox.Show(addedFileCount + _resource.GetString("FileAddMessage"),
                _resource.GetString("AppTitle"), MessageBoxButtons.OK, MessageBoxIcon.Information);

            pbFiles.Value = pbFiles.Minimum;
            if (ActiveForm != null) ActiveForm.Text = _resource.GetString("AppTitle");
        }

        #endregion

        #region Combine PDF Methods

        /// <summary>
        /// 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
        /// </summary>
        /// <param name="sender">The sender info (For example Main Form)</param>
        /// <param name="e">Event Arguments</param>
        private void btnCombinePdfSharp_Click(object sender, EventArgs e)
        {
            try
            {
                if (lbFiles.Items.Count < 2)
                    GenerateCombineWarningMessage();
                else
                {
                    using (var dialogExport = new FolderBrowserDialog())
                    {
                        var result = dialogExport.ShowDialog();
                        if (result != DialogResult.OK || string.IsNullOrEmpty(dialogExport.SelectedPath)) return;

                        SetInitialValuesForCombiningFiles(dialogExport, out var outputFileName, out var fileCount,
                            out var combinedFiles);

                        using (var outputFile = new PdfDocument())
                        {
                            SetOutputFilePropertiesPdfSharp(outputFile);

                            foreach (var t in lbFiles.Items)
                            {
                                try
                                {
                                    AddPdfContentToTargetPdfFilePdfSharp(t, outputFile, ref combinedFiles);
                                }
                                catch (Exception)
                                {
                                    fileCount--;
                                }

                                SetCombinationRatio(combinedFiles, fileCount);
                            }

                            SaveOutputPdfFile(outputFile, outputFileName);
                        }

                        GenerateCombineFileMessage(fileCount, outputFileName);
                    }
                }
            }
            catch (Exception ex)
            {
                GenerateExceptionMessage(ex);
            }
        }

        /// <summary>
        /// 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
        /// </summary>
        /// <param name="sender">The sender info (For example Main Form)</param>
        /// <param name="e">Event Arguments</param>
        private void btnCombineITextSharp_Click(object sender, EventArgs e)
        {
            try
            {
                if (lbFiles.Items.Count < 2)
                    GenerateCombineWarningMessage();
                else
                {
                    using (var dialogExport = new FolderBrowserDialog())
                    {
                        var result = dialogExport.ShowDialog();
                        if (result != DialogResult.OK || string.IsNullOrEmpty(dialogExport.SelectedPath)) return;

                        SetInitialValuesForCombiningFiles(dialogExport, out var outputFileName, out var fileCount,
                            out var combinedFiles);

                        var outputFile = new Document();
                        using (var outputFileStream = new FileStream(outputFileName, FileMode.Create))
                        {
                            using (var pdfWriter = new PdfCopy(outputFile, outputFileStream))
                            {
                                pdfWriter.SetFullCompression();
                                outputFile.Open();
                                foreach (var t in lbFiles.Items)
                                {
                                    try
                                    {
                                        AddPdfContentToTargetPdfFileITextSharp(t, pdfWriter, ref combinedFiles);
                                    }
                                    catch (Exception)
                                    {
                                        fileCount--;
                                    }

                                    SetCombinationRatio(combinedFiles, fileCount);
                                }

                                pdfWriter.Close();
                            }

                            outputFile.Close();
                        }

                        GenerateCombineFileMessage(fileCount, outputFileName);
                    }
                }
            }
            catch (Exception ex)
            {
                GenerateExceptionMessage(ex);
            }
        }

        /// <summary>
        /// This method is used to
        /// Save Combined PDF File
        /// </summary>
        /// <param name="outputFile">Combined PDF File</param>
        /// <param name="outputFileName">Combined PDF File Path</param>
        private static void SaveOutputPdfFile(PdfDocument outputFile, string
outputFileName)
        {
            outputFile.Save(outputFileName);
            outputFile.Close();
        }

        /// <summary>
        /// This method is used to take source pdf file content
        /// And add to the target combined pdf file
        /// </summary>
        /// <param name="t">Source PDF File Path</param>
        /// <param name="outputFile">Combined PDF File</param>
        /// <param name="combinedFiles">Combined File Count</param>
        private static void AddPdfContentToTargetPdfFilePdfSharp(object t, PdfDocument
outputFile, ref int combinedFiles)
        {
            var inputDocument = PdfReader.Open(t.ToString(),
                PdfDocumentOpenMode.Import);
            var count = inputDocument.PageCount;
            for (var index = 0; index < count; index++)
            {
                var page = inputDocument.Pages[index];
                outputFile.AddPage(page);
            }

            inputDocument.Close();
            combinedFiles++;
        }

        /// <summary>
        /// This method is used to set combined PDF file
        /// Properties like Compression etc.
        /// </summary>
        /// <param name="outputFile">Combined PDF File</param>
        private static void SetOutputFilePropertiesPdfSharp(PdfDocument outputFile)
        {
            outputFile.Options.CompressContentStreams = true;
            outputFile.Options.EnableCcittCompressionForBilevelImages = true;
            outputFile.Options.FlateEncodeMode = PdfFlateEncodeMode.BestCompression;
        }

        /// <summary>
        /// This method is used to take source pdf file content
        /// And add to the target combined pdf file
        /// </summary>
        /// <param name="t">Source PDF File Path</param>
        /// <param name="pdfWriter">PDF Writer</param>
        /// <param name="combinedFiles">Combined File Count</param>
        private static void AddPdfContentToTargetPdfFileITextSharp(object t, PdfCopy
pdfWriter, ref int combinedFiles)
        {
            var pdfReader =
                new iTextSharp.text.pdf.PdfReader(t.ToString());
            pdfReader.ConsolidateNamedDestinations();
            for (var j = 1; j <= pdfReader.NumberOfPages; j++)
            {
                var page = pdfWriter.GetImportedPage(pdfReader, j);
                pdfWriter.AddPage(page);
            }

            pdfReader.Close();
            combinedFiles++;
        }

        /// <summary>
        /// This method is used
        /// When user just wnat to combine 1 pdf file
        /// </summary>
        private static void GenerateCombineWarningMessage()
        {
            MessageBox.Show(_resource.GetString("CombineWarning"),
_resource.GetString("AppTitle"), MessageBoxButtons.OK, MessageBoxIcon.Error);
        }

        /// <summary>
        /// This method is used to
        /// Show error message when exception occurs and set form title initial value
        /// </summary>
        /// <param name="ex">Exception information</param>
        private void GenerateExceptionMessage(Exception ex)
        {
            MessageBox.Show(_resource.GetString("CombineErrorMessage") +
ex.GetAllMessages(), _resource.GetString("AppTitle"), MessageBoxButtons.OK,
MessageBoxIcon.Error);
            pbFiles.Value = pbFiles.Minimum;
            if (ActiveForm != null) ActiveForm.Text = _resource.GetString("AppTitle");
        }

        /// <summary>
        /// This method is used to set initial values of some properties
        /// Which are used in combining PDF files
        /// In both PdfSharp and ITextSharp
        /// </summary>
        /// <param name="dialogExport">Export Dialog Info</param>
        /// <param name="outputFileName">Output File Name</param>
        /// <param name="fileCount">Total File Count</param>
        /// <param name="combinedFiles">Combined Files Count</param>
        private void SetInitialValuesForCombiningFiles(FolderBrowserDialog
dialogExport, out string outputFileName,
            out int fileCount, out int combinedFiles)
        {
            outputFileName = dialogExport.SelectedPath + "/" + Guid.NewGuid() + ".pdf";
            fileCount = lbFiles.Items.Count;
            combinedFiles = 0;
        }

        /// <summary>
        /// This method is used to set combination degree while combining PDF files
        /// And show this ratio on Form Caption
        /// </summary>
        /// <param name="combinedFiles">Combined File Count</param>
        /// <param name="fileCount">Total File Count</param>
        private void SetCombinationRatio(int combinedFiles, int fileCount)
        {
            pbFiles.Value = combinedFiles * 100 / fileCount;
            if (pbFiles.Value > pbFiles.Maximum)
                pbFiles.Value = pbFiles.Maximum;
            if (ActiveForm != null) ActiveForm.Text = @"%" + pbFiles.Value;
        }

        /// <summary>
        /// This method is used to set successfull combining file message
        /// And set value of Form Caption etc.
        /// </summary>
        /// <param name="fileCount">Total File Count</param>
        /// <param name="outputFileName">Output File Name</param>
        private void GenerateCombineFileMessage(int fileCount, string outputFileName)
        {
            MessageBox.Show(fileCount + _resource.GetString("CombineFileMessage") + outputFileName, _resource.GetString("AppTitle"), MessageBoxButtons.OK, MessageBoxIcon.Information);
            pbFiles.Value = pbFiles.Minimum;
            if (ActiveForm != null) ActiveForm.Text = _resource.GetString("AppTitle");
        }

        #endregion

        #region Delete Item Methods

        /// <summary>
        /// This function is used to operate some process in files in listbox which are chosen
        /// </summary>
        /// <param name="sender">The sender info (For example Main Form)</param>
        /// <param name="e">Event Arguments</param>
        private void lbFiles_KeyDown(object sender, KeyEventArgs e)
        {
            // this is used to delete selected files in listbox when you press Delete key
            if (e.KeyValue == Keys.Delete.GetHashCode())
                DeleteFilesFromListBox();
            // This is used to select all items in listbox when you press Ctrl + A key combination
            if (!e.Control || e.KeyCode != Keys.A) return;
            for (var i = 0; i < lbFiles.Items.Count; i++)
            {
                lbFiles.SetSelected(i, true);
            }
        }

        /// <summary>
        /// This is used to delete selected items in listbox
        /// When menu item Delete is clicked and opened dialog
        /// </summary>
        /// <param name="sender">The sender info (For example Main Form)</param>
        /// <param name="e">Event Arguments</param>
        private void menuItemDelete_Click(object sender, EventArgs e)
        {
            if (lbFiles.Items.Count > 0)
            {
                if (lbFiles.SelectedItems.Count > 0)
                {
                    ShowDeleteDialog(out var deleteDialog);
                    if (deleteDialog == DialogResult.Yes)
                        DeleteFilesFromListBox();
                }
                else
                    GenerateNoSelectedFileInListBoxMessage();
            }
            else
                GenerateNoFileInListBoxMessage();
        }

        /// <summary>
        /// This method is used in
        /// When there is no selected file in listbox
        /// </summary>
        private static void GenerateNoSelectedFileInListBoxMessage()
        {
            MessageBox.Show(_resource.GetString("NoSelectedFile"), _resource.GetString("AppTitle"),
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }

        /// <summary>
        /// This method is used to
        /// Show delete file dialog
        /// </summary>
        /// <param name="deleteDialog">Delete File Dialog</param>
        private void ShowDeleteDialog(out DialogResult deleteDialog)
        {
            deleteDialog = MessageBox.Show( _resource.GetString("DeleteWarning1") + lbFiles.SelectedItems.Count + _resource.GetString("DeleteWarning2"), _resource.GetString("AppTitle"), MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
        }

        /// <summary>
        /// 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
        /// </summary>
        private void DeleteFilesFromListBox()
        {
            var fileCount = lbFiles.SelectedItems.Count;
            var removedItems = (from object t in lbFiles.SelectedItems select t.ToString()).ToList();

            var allItems = (from object item in lbFiles.Items where removedItems.All(j => j != item.ToString()) select item.ToString()).ToList();

            lbFiles.Items.Clear();
            foreach (var item in allItems)
            {
                lbFiles.Items.Add(item);
            }

            GenerateDeleteItemMessage(fileCount);
        }

        /// <summary>
        /// This function is used to clear the file list in listbox
        /// </summary>
        /// <param name="sender">The sender info (For example Main Form)</param>
        /// <param name="e">Event Arguments</param>
        private void btnClearList_Click(object sender, EventArgs e)
        {
            var fileCount = lbFiles.Items.Count;
            lbFiles.Items.Clear();
            GenerateDeleteItemMessage(fileCount);
        }

        /// <summary>
        /// This method is used after deleting items from listbox
        /// </summary>
        /// <param name="fileCount">Deleted File Count Information</param>
        private static void GenerateDeleteItemMessage(int fileCount)
        {
            MessageBox.Show(fileCount + _resource.GetString("DeleteFileMessage"), _resource.GetString("AppTitle"), MessageBoxButtons.OK, MessageBoxIcon.Information);
        }

        #endregion

        #region Order Item Methods

        /// <summary>
        /// This method is used to order elements in listbox ascending by full path
        /// </summary>
        /// <param name="sender">The sender info (For example Main Form)</param>
        /// <param name="e">Event Arguments</param>
        private void menuItemOrderByPathAscending_Click(object sender, EventArgs e)
        {
            if (lbFiles.Items.Count > 0)
                lbFiles = SortItems(lbFiles, SortType.ByPath, OrderType.Ascending);
                // lbFiles = SortItemsByPath(lbFiles, OrderType.Ascending);
            else
                GenerateNoFileInListBoxMessage();
        }

        /// <summary>
        /// This method is used to order elements in listbox descending by full path
        /// </summary>
        /// <param name="sender">The sender info (For example Main Form)</param>
        /// <param name="e">Event Arguments</param>
        private void menuItemOrderByPathDescending_Click(object sender, EventArgs e)
        {
            if (lbFiles.Items.Count > 0)
                lbFiles = SortItems(lbFiles, SortType.ByPath, OrderType.Descending);
                // lbFiles = SortItemsByPath(lbFiles, OrderType.Descending);
            else
                GenerateNoFileInListBoxMessage();
        }

        /// <summary>
        /// This method is used to order elements in listbox ascending by file name
        /// </summary>
        /// <param name="sender">The sender info (For example Main Form)</param>
        /// <param name="e">Event Arguments</param>
        private void menuItemOrderByNameAscending_Click(object sender, EventArgs e)
        {
            if (lbFiles.Items.Count > 0)
                lbFiles = SortItems(lbFiles, SortType.ByName, OrderType.Ascending);
                // lbFiles = SortItemsByName(lbFiles, OrderType.Ascending);
            else
                GenerateNoFileInListBoxMessage();
        }

        /// <summary>
        /// This method is used to order elements in listbox descending by file name
        /// </summary>
        /// <param name="sender">The sender info (For example Main Form)</param>
        /// <param name="e">Event Arguments</param>
        private void menuItemOrderByNameDescending_Click(object sender, EventArgs e)
        {
            if (lbFiles.Items.Count > 0)
                lbFiles = SortItems(lbFiles, SortType.ByName, OrderType.Descending);
                // lbFiles = SortItemsByName(lbFiles, OrderType.Descending);
            else
                GenerateNoFileInListBoxMessage();
        }

        /// <summary>
        /// This method is used to generate message
        /// When there is no item in listbox
        /// </summary>
        private static void GenerateNoFileInListBoxMessage()
        {
            MessageBox.Show(_resource.GetString("NoFile"),
_resource.GetString("AppTitle"), MessageBoxButtons.OK, MessageBoxIcon.Error);
        }

        /// <summary>
        /// This method is used to order items by full path in listbox
        /// And return them with the given order as parametre
        /// </summary>
        /// <param name="listBox">ListBox Info</param>
        /// <param name="sortType">Sort Type (By Name or Path)</param>
        /// <param name="orderType">Order Type is Ascending Or Not</param>
        private static ListBox SortItems(ListBox listBox, SortType sortType, OrderType orderType)
        {
            var items = listBox.Items.OfType<object>().ToList();
            listBox.Items.Clear();

            // Sort Items By Name
            if (sortType == SortType.ByName)
            {
                listBox.Items.AddRange(orderType == OrderType.Ascending
                    ? items.OrderBy(i => i).ToArray()
                    : items.OrderByDescending(i => i).ToArray());
            }
            // Sort Items By Path
            else
            {
                const char c = '\u005c';
                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();

                fileList = orderType == OrderType.Ascending
                    ? fileList.OrderBy(j => j.FileName).ThenBy(j => j.FilePath).ToList()
                    : fileList.OrderByDescending(j => j.FileName).ThenByDescending(j => j.FilePath).ToList();

                foreach (var item in fileList)
                    listBox.Items.Add(item.FilePath);
            }

            return listBox;
        }

        /// <summary>
        /// This method is used to order items by full path in listbox
        /// And return them with the given order as parametre
        /// </summary>
        /// <param name="listBox">ListBox Info</param>
        /// <param name="orderType">Order Type is Ascending Or Not</param>
        private static ListBox SortItemsByPath(ListBox listBox, OrderType orderType)
        {
            var items = listBox.Items.OfType<object>().ToList();
            listBox.Items.Clear();
            listBox.Items.AddRange(orderType == OrderType.Ascending
                ? items.OrderBy(i => i).ToArray()
                : items.OrderByDescending(i => i).ToArray());
            return listBox;
        }

        /// <summary>
        /// This method is used to order items by file name in listbox
        /// And return them with the given order as parametre
        /// </summary>
        /// <param name="listBox">ListBox Info</param>
        /// <param name="orderType">Order Type is Ascending Or Not</param>
        private static ListBox SortItemsByName(ListBox listBox, OrderType orderType)
        {
            var items = listBox.Items.OfType<object>().ToList();
            const char c = '\u005c';
            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();

            fileList = orderType == OrderType.Ascending
                ? fileList.OrderBy(j => j.FileName).ThenBy(j => j.FilePath).ToList()
                : fileList.OrderByDescending(j => j.FileName).ThenByDescending(j => j.FilePath).ToList();

            listBox.Items.Clear();
            foreach (var item in fileList)
                listBox.Items.Add(item.FilePath);

            return listBox;
        }

        #endregion

        #region Item Drag and Drop Methods

        /// <summary>
        /// This is used to control mouse actions and if that is not right click
        /// Start to drag and drop item in listbox
        /// </summary>
        /// <param name="sender">The sender info (For example Main Form)</param>
        /// <param name="e">Event Arguments</param>
        private void lbFiles_MouseDown(object sender, MouseEventArgs e)
        {
            if (e.Button.Equals(MouseButtons.Right)) return;
            if (lbFiles.SelectedItem == null) return;
            lbFiles.DoDragDrop(lbFiles.SelectedItem, DragDropEffects.Move);
        }

        /// <summary>
        /// This is used to give effect while dragging and dropping an item in listbox
        /// </summary>
        /// <param name="sender">The sender info (For example Main Form)</param>
        /// <param name="e">Event Arguments</param>
        private void lbFiles_DragOver(object sender, DragEventArgs e)
        {
            e.Effect = DragDropEffects.Move;
        }

        /// <summary>
        /// This is used to drag and drop an item in listbox
        /// Which is used to order books and combine them with that order
        /// </summary>
        /// <param name="sender">The sender info (For example Main Form)</param>
        /// <param name="e">Event Arguments</param>m>
        private void lbFiles_DragDrop(object sender, DragEventArgs e)
        {
            var point = lbFiles.PointToClient(new Point(e.X, e.Y));
            var index = lbFiles.IndexFromPoint(point);
            if (index < 0) index = lbFiles.Items.Count - 1;
            // type of string because file names stored in string in listbox
            var data = e.Data.GetData(typeof(string));
            lbFiles.Items.Remove(data);
            lbFiles.Items.Insert(index, data);
        }

        #endregion
    }
}
```

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

```
?namespace PdfCombiner
{
    partial class FrmMain
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            System.ComponentModel.ComponentResourceManager resources = new System.ComponentModel.ComponentResourceManager(typeof(FrmMain));
            this.btnCombinePdfSharp = new System.Windows.Forms.Button();
            this.btnAddFolder = new System.Windows.Forms.Button();
            this.btnAddFile = new System.Windows.Forms.Button();
            this.btnClearList = new System.Windows.Forms.Button();
            this.lbFiles = new System.Windows.Forms.ListBox();
            this.lblDetails = new System.Windows.Forms.Label();
            this.pbFiles = new System.Windows.Forms.ProgressBar();
            this.btnCombineITextSharp = new System.Windows.Forms.Button();
            this.menuStrip = new System.Windows.Forms.ContextMenuStrip(this.components);
            this.menuItemDelete = new System.Windows.Forms.ToolStripMenuItem();
            this.menuItemOrderByPathAscending = new System.Windows.Forms.ToolStripMenuItem();
            this.menuItemOrderByPathDescending = new System.Windows.Forms.ToolStripMenuItem();
            this.menuItemOrderByNameAscending = new System.Windows.Forms.ToolStripMenuItem();
            this.menuItemOrderByNameDescending = new System.Windows.Forms.ToolStripMenuItem();
            this.cmbLanguage = new System.Windows.Forms.ComboBox();
            this.menuStrip.SuspendLayout();
            this.SuspendLayout();
            //
            // btnCombinePdfSharp
            //
            this.btnCombinePdfSharp.Image = ((System.Drawing.Image)(resources.GetObject("btnCombinePdfSharp.Image")));
            this.btnCombinePdfSharp.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
            this.btnCombinePdfSharp.Location = new System.Drawing.Point(442, 5);
            this.btnCombinePdfSharp.Name = "btnCombinePdfSharp";
            this.btnCombinePdfSharp.Size = new System.Drawing.Size(214, 37);
            this.btnCombinePdfSharp.TabIndex = 0;
            this.btnCombinePdfSharp.Text = "Combine Files (PdfSharp)";
            this.btnCombinePdfSharp.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
            this.btnCombinePdfSharp.UseVisualStyleBackColor = true;
            this.btnCombinePdfSharp.Click += new System.EventHandler(this.btnCombinePdfSharp_Click);
            //
            // btnAddFolder
            //
            this.btnAddFolder.Image = ((System.Drawing.Image)(resources.GetObject("btnAddFolder.Image")));
            this.btnAddFolder.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
            this.btnAddFolder.Location = new System.Drawing.Point(197, 5);
            this.btnAddFolder.Name = "btnAddFolder";
            this.btnAddFolder.Size = new System.Drawing.Size(171, 37);
            this.btnAddFolder.TabIndex = 2;
            this.btnAddFolder.Text = "Add Folder";
            this.btnAddFolder.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
            this.btnAddFolder.UseVisualStyleBackColor = true;
            this.btnAddFolder.Click += new System.EventHandler(this.btnAddFolder_Click);
            //
            // btnAddFile
            //
            this.btnAddFile.Image = ((System.Drawing.Image)(resources.GetObject("btnAddFile.Image")));
            this.btnAddFile.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
            this.btnAddFile.Location = new System.Drawing.Point(12, 5);
            this.btnAddFile.Name = "btnAddFile";
            this.btnAddFile.Size = new System.Drawing.Size(179, 37);
            this.btnAddFile.TabIndex = 3;
            this.btnAddFile.Text = "Add Files";
            this.btnAddFile.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
            this.btnAddFile.UseVisualStyleBackColor = true;
            this.btnAddFile.Click += new System.EventHandler(this.btnAddFile_Click);
            //
            // btnClearList
            //
            this.btnClearList.Image = ((System.Drawing.Image)(resources.GetObject("btnClearList.Image")));
            this.btnClearList.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
            this.btnClearList.Location = new System.Drawing.Point(12, 45);
            this.btnClearList.Name = "btnClearList";
            this.btnClearList.Size = new System.Drawing.Size(179, 37);
            this.btnClearList.TabIndex = 4;
            this.btnClearList.Text = "Clear File List";
            this.btnClearList.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
            this.btnClearList.UseVisualStyleBackColor = true;
            this.btnClearList.Click += new System.EventHandler(this.btnClearList_Click);
            //
            // lbFiles
            //
            this.lbFiles.AllowDrop = true;
            this.lbFiles.FormattingEnabled = true;
            this.lbFiles.Location = new System.Drawing.Point(12, 88);
            this.lbFiles.Name = "lbFiles";
            this.lbFiles.SelectionMode = System.Windows.Forms.SelectionMode.MultiExtended;
            this.lbFiles.Size = new System.Drawing.Size(644, 186);
            this.lbFiles.TabIndex = 5;
            this.lbFiles.DragDrop += new System.Windows.Forms.DragEventHandler(this.lbFiles_DragDrop);
            this.lbFiles.DragOver += new System.Windows.Forms.DragEventHandler(this.lbFiles_DragOver);
            this.lbFiles.KeyDown += new System.Windows.Forms.KeyEventHandler(this.lbFiles_KeyDown);
            this.lbFiles.MouseDown += new System.Windows.Forms.MouseEventHandler(this.lbFiles_MouseDown);
            // 
            // lblDetails
            // 
            this.lblDetails.AutoSize = true;
            this.lblDetails.Location = new System.Drawing.Point(12, 65);
            this.lblDetails.Name = "lblDetails";
            this.lblDetails.Size = new System.Drawing.Size(0, 13);
            this.lblDetails.TabIndex = 6;
            // 
            // pbFiles
            // 
            this.pbFiles.Location = new System.Drawing.Point(197, 45);
            this.pbFiles.Name = "pbFiles";
            this.pbFiles.Size = new System.Drawing.Size(239, 37);
            this.pbFiles.TabIndex = 7;
            // 
            // btnCombineITextSharp
            // 
            this.btnCombineITextSharp.Image = ((System.Drawing.Image)(resources.GetObject("btnCombineITextSharp.Image")));
            this.btnCombineITextSharp.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
            this.btnCombineITextSharp.Location = new System.Drawing.Point(442, 45);
            this.btnCombineITextSharp.Name = "btnCombineITextSharp";
            this.btnCombineITextSharp.Size = new System.Drawing.Size(214, 37);
            this.btnCombineITextSharp.TabIndex = 9;
            this.btnCombineITextSharp.Text = "Combine Files (iTextSharp)";
            this.btnCombineITextSharp.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
            this.btnCombineITextSharp.UseVisualStyleBackColor = true;
            this.btnCombineITextSharp.Click += new System.EventHandler(this.btnCombineITextSharp_Click);
            // 
            // menuStrip
            // 
            this.menuStrip.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {this.menuItemDelete, this.menuItemOrderByPathAscending, this.menuItemOrderByPathDescending, this.menuItemOrderByNameAscending, this.menuItemOrderByNameDescending});
            this.menuStrip.Name = "menuStrip";
            this.menuStrip.Size = new System.Drawing.Size(229, 114);
            // 
            // menuItemDelete
            // 
            this.menuItemDelete.Image = ((System.Drawing.Image)(resources.GetObject("menuItemDelete.Image")));
            this.menuItemDelete.Name = "menuItemDelete";
            this.menuItemDelete.Size = new System.Drawing.Size(228, 22);
            this.menuItemDelete.Text = "Delete";
            this.menuItemDelete.Click += new System.EventHandler(this.menuItemDelete_Click);
            // 
            // menuItemOrderByPathAscending
            // 
            this.menuItemOrderByPathAscending.Image = ((System.Drawing.Image)(resources.GetObject("menuItemOrderByPathAscending.Image")));
            this.menuItemOrderByPathAscending.Name = "menuItemOrderByPathAscending";
            this.menuItemOrderByPathAscending.Size = new System.Drawing.Size(228, 22);
            this.menuItemOrderByPathAscending.Text = "Order By Path (Ascending)";
            this.menuItemOrderByPathAscending.Click += new System.EventHandler(this.menuItemOrderByPathAscending_Click);
            //
            // menuItemOrderByPathDescending
            //
            this.menuItemOrderByPathDescending.Image = ((System.Drawing.Image)(resources.GetObject("menuItemOrderByPathDescending.Image")));
            this.menuItemOrderByPathDescending.Name = "menuItemOrderByPathDescending";
            this.menuItemOrderByPathDescending.Size = new System.Drawing.Size(228, 22);
            this.menuItemOrderByPathDescending.Text = "Order By Path (Descending)";
            this.menuItemOrderByPathDescending.Click += new System.EventHandler(this.menuItemOrderByPathDescending_Click);
            //
            // menuItemOrderByNameAscending
            //
            this.menuItemOrderByNameAscending.Image = ((System.Drawing.Image)(resources.GetObject("menuItemOrderByNameAscending.Image")));
            this.menuItemOrderByNameAscending.Name = "menuItemOrderByNameAscending";
            this.menuItemOrderByNameAscending.Size = new System.Drawing.Size(228, 22);
            this.menuItemOrderByNameAscending.Text = "Order By Name (Ascending)";
            this.menuItemOrderByNameAscending.Click += new System.EventHandler(this.menuItemOrderByNameAscending_Click);
            //
            // menuItemOrderByNameDescending
            //
            this.menuItemOrderByNameDescending.Image = ((System.Drawing.Image)(resources.GetObject("menuItemOrderByNameDescending.Image")));
            this.menuItemOrderByNameDescending.Name = "menuItemOrderByNameDescending";
            this.menuItemOrderByNameDescending.Size = new System.Drawing.Size(228, 22);
            this.menuItemOrderByNameDescending.Text = "Order By Name (Descending)";
            this.menuItemOrderByNameDescending.Click += new System.EventHandler(this.menuItemOrderByNameDescending_Click);
            //
            // cmbLanguage
            //
            this.cmbLanguage.Anchor = System.Windows.Forms.AnchorStyles.None;
            this.cmbLanguage.Font = new System.Drawing.Font("Microsoft Sans Serif", 15F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte) (162)));
            this.cmbLanguage.FormattingEnabled = true;
            this.cmbLanguage.Items.AddRange(new object[] {"TR", "EN", "DE", "FR", "RU", "ES"});
            this.cmbLanguage.Location = new System.Drawing.Point(374, 6);
            this.cmbLanguage.Name = "cmbLanguage";
            this.cmbLanguage.Size = new System.Drawing.Size(62, 33);
            this.cmbLanguage.TabIndex = 10;
            this.cmbLanguage.Text = "Select Language";
            this.cmbLanguage.SelectedIndexChanged += new System.EventHandler(this.cmbLanguage_SelectedIndexChanged);
            //
            // FrmMain
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(668, 287);
            this.Controls.Add(this.cmbLanguage);
            this.Controls.Add(this.btnCombineITextSharp);
            this.Controls.Add(this.pbFiles);
            this.Controls.Add(this.lblDetails);
            this.Controls.Add(this.lbFiles);
            this.Controls.Add(this.btnClearList);
            this.Controls.Add(this.btnAddFile);
            this.Controls.Add(this.btnAddFolder);
            this.Controls.Add(this.btnCombinePdfSharp);
            this.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte) (162)));
            this.Name = "FrmMain";
            this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
            this.Text = "PDF Combiner";
            this.FormClosed += new System.Windows.Forms.FormClosedEventHandler(this.FrmMain_FormClosed);
            this.Shown += new System.EventHandler(this.FrmMain_Shown);
            this.menuStrip.ResumeLayout(false);
            this.ResumeLayout(false);
            this.PerformLayout();
        }

        public System.Windows.Forms.ComboBox cmbLanguage;

        #endregion

        private System.Windows.Forms.Button btnCombinePdfSharp;
        private System.Windows.Forms.Button btnAddFolder;
        private System.Windows.Forms.Button btnAddFile;
        private System.Windows.Forms.Button btnClearList;
        private System.Windows.Forms.ListBox lbFiles;
        private System.Windows.Forms.Label lblDetails;
        private System.Windows.Forms.ProgressBar pbFiles;
        private System.Windows.Forms.Button btnCombineITextSharp;
        private System.Windows.Forms.ContextMenuStrip menuStrip;
        private System.Windows.Forms.ToolStripMenuItem menuItemDelete;
        private System.Windows.Forms.ToolStripMenuItem menuItemOrderByPathAscending;
        private System.Windows.Forms.ToolStripMenuItem menuItemOrderByPathDescending;
        private System.Windows.Forms.ToolStripMenuItem menuItemOrderByNameAscending;
        private System.Windows.Forms.ToolStripMenuItem menuItemOrderByNameDescending;
    }
}

```

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

```
?using System;
using System.Windows.Forms;

namespace PdfCombiner
{
    internal static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        private static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new FrmMain());
        }
    }
}
```

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

```
?//------------------------------------------------------------------------------
// <auto-generated>
//     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.
// </auto-generated>
//------------------------------------------------------------------------------

namespace PdfCombiner.Properties {
    using System;


    /// <summary>
    ///   A strongly-typed resource class, for looking up localized strings, etc.
    /// </summary>
    // 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.
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.StronglyTypedResourceBuilder",
"15.0.0.0")]
    [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
    [global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
    internal class Resources {

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

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

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

        /// <summary>
        ///   Returns the cached ResourceManager instance used by this class.
        /// </summary>
[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]
        internal static global::System.Resources.ResourceManager ResourceManager {
            get {
                if (object.ReferenceEquals(resourceMan, null)) {
                    global::System.Resources.ResourceManager temp = new global::System.Resources.ResourceManager("PdfCombiner.Properties.Resources", typeof(Resources).Assembly);
                    resourceMan = temp;
                }
                return resourceMan;
            }
        }
        
        /// <summary>
        ///   Overrides the current thread's CurrentUICulture property for all
        ///   resource lookups using this strongly typed resource class.
        /// </summary>
[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]
        internal static global::System.Globalization.CultureInfo Culture {
            get {
                return resourceCulture;
            }
            set {
                resourceCulture = value;
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to PDF Birlestirici.
        /// </summary>
        internal static string AppTitle {
            get {
                return ResourceManager.GetString("AppTitle", resourceCulture);
            }
        }
    }
}
```

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

```
?//------------------------------------------------------------------------------
// <auto-generated>
//     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.
// </auto-generated>
//------------------------------------------------------------------------------

```

```
namespace PdfCombiner.Properties
{

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

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

Namespaces

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

Index