Course: EGDF20
Module: EGE202 Application Programming

SDL2: Familiarizing with .NET Controls & Assemblies

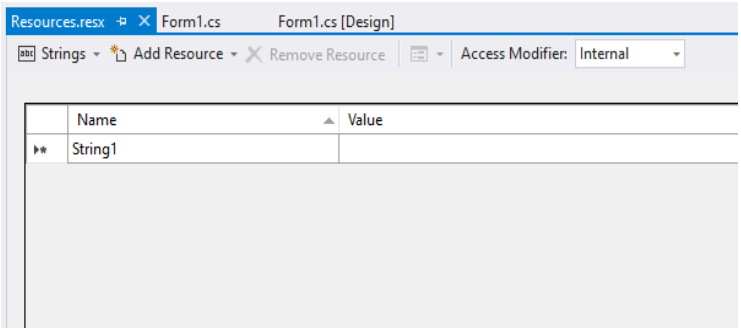
Objectives: At the end of this lab, the student should be familiarize with different type of Controls in .NET Framework. Student will also learn how add additional libraries in their solution in order to expand the functionality of the application program.

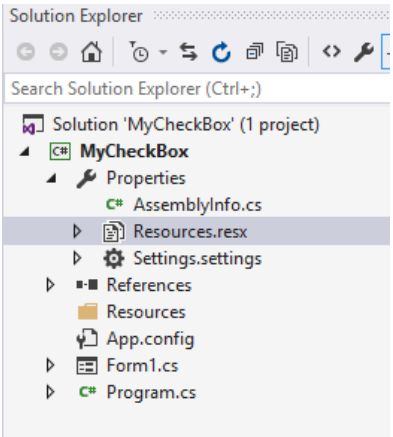
Exercise 1 – Experimenting with .NET Controls

Part 1: Working with CheckBox & PictureBox Control

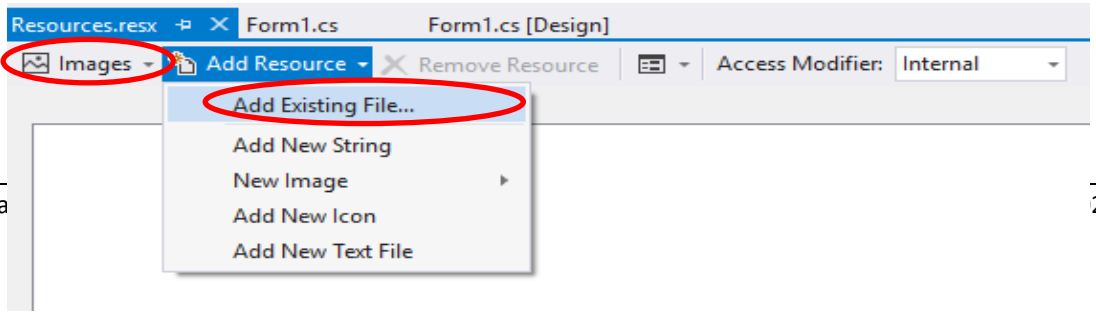
1. Under the ***File*** menu, click ***New Project*** or use the ***New Project*** button to create a new project. Alternatively, use the ***Create New Project*** link in the ***Get Started*** *popup* dialog.
2. From the pop-up dialog, select **"C#"** for the ***Language filter***, **"Windows"** for ***the Platform filter*** and **"Desktop"** for the ***Project type filter***.
3. Then choose ***Windows Forms App (.Net Framework)*** and click the ***Next*** button.
4. Type the name of your new project as ***MyCheckBox*** and keep the Solution name the same as Project name.
5. **Do not** tick on the check-box of [☐ **Place solution and project in the same directory**].
6. Click the **Create** button to start your project.

7. From the ***Solution Explorer*** window, expand the ***Properties*** node and double click on ***Resources.resx*** to open the ***Resources'*** tab.

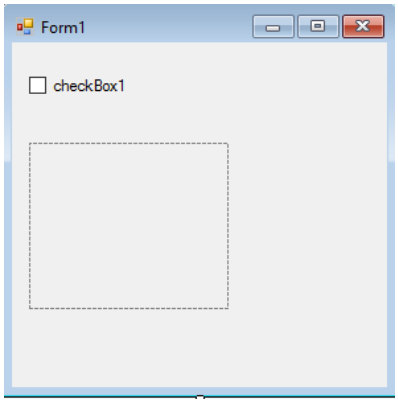




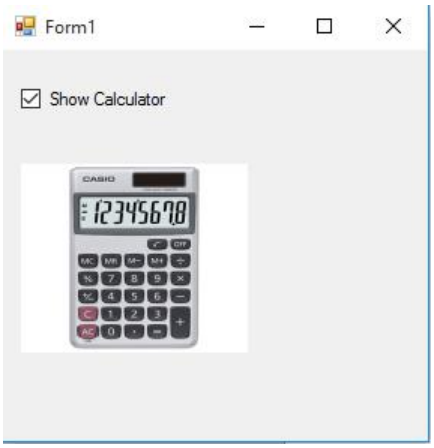
8. Locate both "abacus.jpg" and "calculator.jpg" files on your PC hard drive. From the ***Resources.resx*** tab choose ***Images*** and ***Add Resource->Add Existing File …*** to add both the files as image resources that you can use in the application.

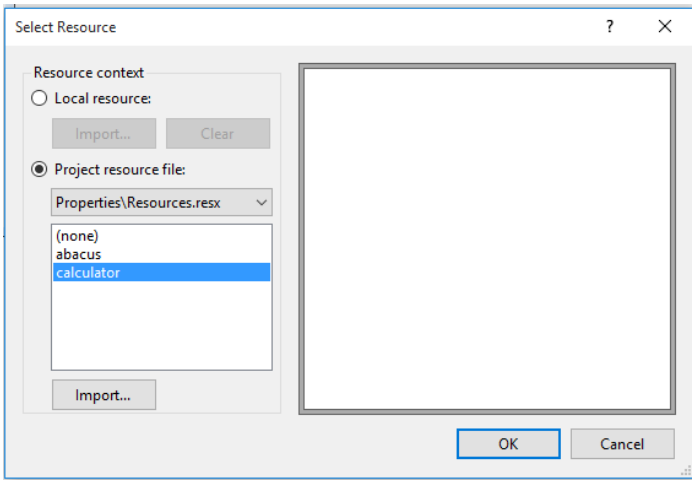


9. Double click on "Form1.cs" ***Solution Explorer*** window to launch the ***Form Designer*** tab.
10. From the ***Toolbar***, drag in 1 *CheckBox* and 1 *PictureBox* control into the *Form1* window. (Note: Resize *Form1* if necessary)
11. Modify the *(Name)* and *Text* properties based on the table below.



{Name} From	{Name} To	Property	Value
checkBox1	chkSel	{Text}	Show Calculator
		{Checked}	True
pcitureBox1	picImg	{Image}	Click ... to add calculator
		{SizeMode}	StretchImage





12. Build and run your application by hitting <F5> key or Start button. You should see an application with the UI shown in the diagram above.
13. Stop the application and proceed to double click on the "***CheckBox*** in the ***Form Designer***. That will automatically create *chkSel_CheckedChange(…)* function. Noticed that now it is no longer handling a *Clicked* event but rather a *CheckedChange* event which triggers each time the state of the *CheckBox* changed either from "unchecked" to "checked" or "checked" to "unchecked".
14. Modify *chkSel_CheckedChange(…)* to include the following codes:

```
private void chkSel_CheckedChanged(object sender, EventArgs e)
{
    if (chkSel.Checked == true)
    {
        picImg.Image = MyCheckBox.Properties.Resources.calculator;
    }
    else
```

```
        {
            picImg.Image = null;
        }
    }
```

15. Analyze and explain the codes added in step 12.

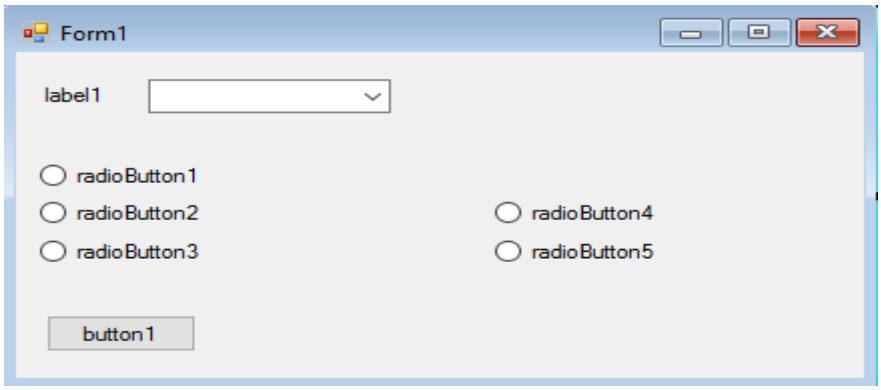
No	Actions	Observation / Action
1	What is the observation when you uncheck and check the *CheckBox*?	

2	`if (chkSel.Checked == true)`
	`{`
	`. . .`
	`}`	

	`picImg.Image = MyCheckBox.Properties.Resources.calculator;`	
	`picImg.Image = null;`	
3	**<u>Coding Task:</u>** Modify the codes such that when the *CheckBox* is unchecked the abacus image is displayed instead	

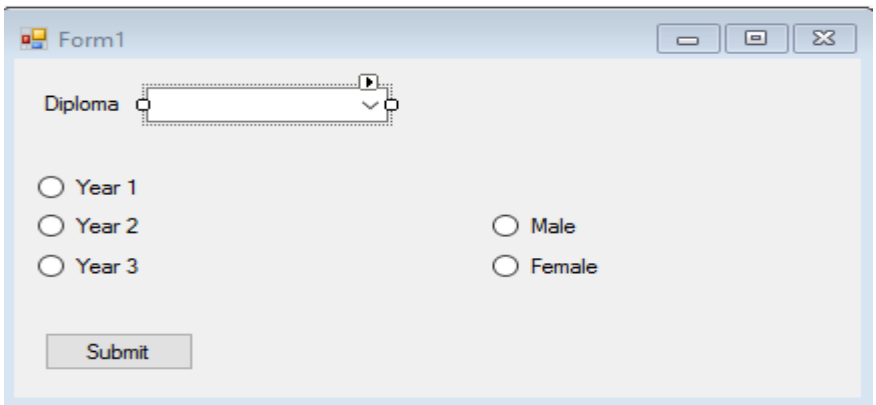
Part 2: Working with ComboBox Control and RadioButton Control

1. Download the pre-created **Visual Studio** solution ***MyRadioButton(Student).zip*** and unzipped it to your own folder.
2. Under the ***File*** menu, click ***Open -> Project/Solution*** to open the project.
3. Your Form design should look like the figure below

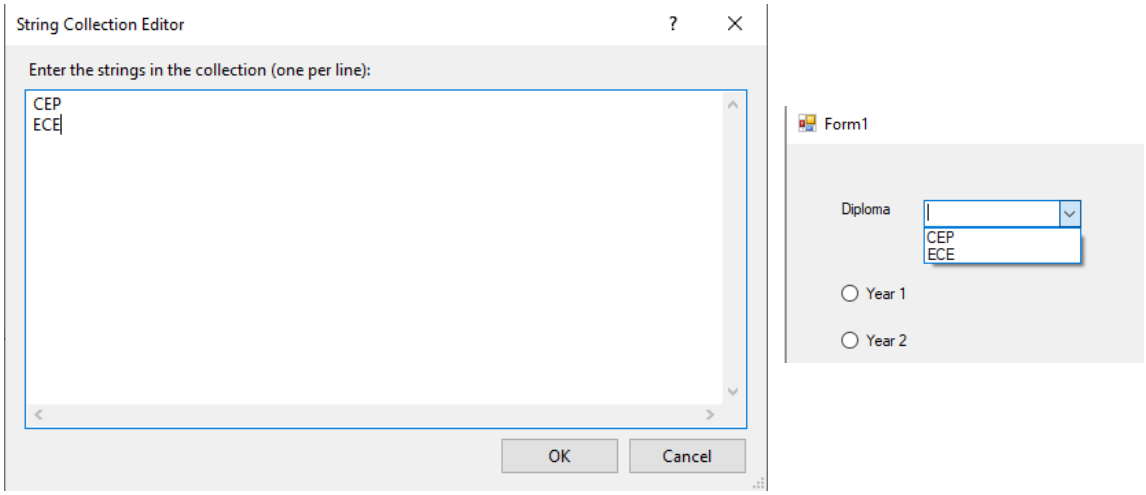


4. The properties based on the table below.

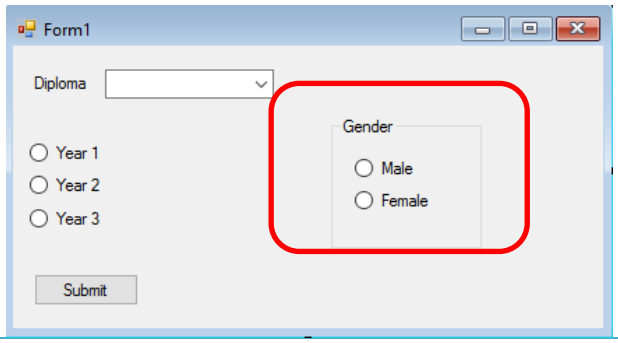
{Name} From	{Name} To	Property	Value
radiobutton1	rbYear1	{Text}	Year 1
radiobutton2	rbYear2	{Text}	Year 2
radiobutton3	rbYear3	{Text}	Year 3
radiobutton4	rbMale	{Text}	Male
radiobutton5	rbFemale	{Text}	Female
label1	lblDip	{Text}	Diploma
button1	btnSubmit	{Text}	Submit
comboBox1	cbCourse	{Items}	Click ... to add



5. Select **cbCourse** *ComboBox* in the Form Designer. Click on the "triangle" icon on the ComboBox or from the ***Properties*** window click on the ... and choose ***Edit Items*** button to launch ***the String Collection Editor***. Enter "ECC" and "ASM" in the editor and click the **OK** button.



6. Build and test the application. Click on the ***ComboBox*** and you should see that the two strings that you have entered earlier appears in the *ComboBox*.
7. Complete the following task

No	Actions	Observation / Action
1	**<u>Coding Task:</u>** Add the **Submit** button event handler to show the selected *ComboBox* value `MessageBox.Show(cbCourse.SelectedItem.ToString()+ " Selected");`	
2	Experiment with the RadioButton controls. Observe the problem associated with these buttons	
3	Drag in the **GroupBox** control into the **Form Designer**.  	Note: 1. If you cannot find ***GroupBox***, ensure under ***ToolBox***, ***All Windows Forms*** is selected 2. You need to place both Male and Female ***RadioButton***s into the ***GroupBox***
4	Experiment with the RadioButton controls. Observe the problem associated with these radiobuttons again	

8. Double click ***Male*** and ***Female*** *RadioButton* to create a 'checked change' event and add following codes:

```
string gender = null;

private void rbMale_CheckedChanged(object sender, EventArgs e)
{
    gender = "Male";
}

private void rbFemlale_CheckedChanged (object sender, EventArgs e)
{
    gender = "Female";
}
```

9. Modify the ***Submit*** button click event handler to the following such that the gender and diploma of the student is shown in the *MessageBox.*

```
private void btnSubmit_Click(object sender, EventArgs e)
{
    if (gender !=null)
        MessageBox.Show(gender + " student, " +
            cbCourse.SelectedItem.ToString());

}
```

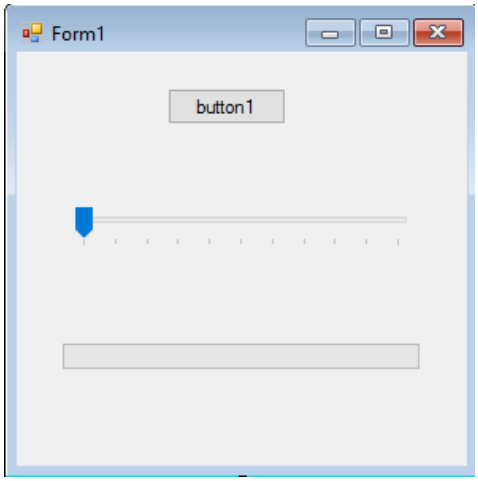
10. Build and test the application
11. Analyze and explain the codes added in step 8 and 9.

No	Actions	Observation / Action
1	What is the observation when the gender option is not selected?	
2	What is ***null*** used for? `string gender = null;`	
	Why do we compare gender to a ***null***? `if (gender !=null)`	

Part 3: Experimenting with Timer and Numeric Value Control

1. As with the previous exercise, create a new project and named it as ***MyTimerAndProgressBar***.
2. Double click on "Form1.cs" ***Solution Explorer*** window to launch the ***Form Designer*** tab.

3. From the ***Toolbar***, drag in 1 *Timer*, 1 *Button*, 1 *ProgressBar* and 1 *TrackBar* control into the *Form1* window.



{Name} From	Property	Value
progressBar1	{ Maximum}	1000
trackBar1	{Maximum}	1000
	{LargeChange}	50
	{SmallChange}	10
	{TickFrquency}	50
button1	{Text}	Start Timer
timer1		

(Note: For the *TrackBar*, *LargeChange* can be controlled via keyboard *PgUp/PgDn* keys while *SmallChange* via keyboard *Arrow* keys)

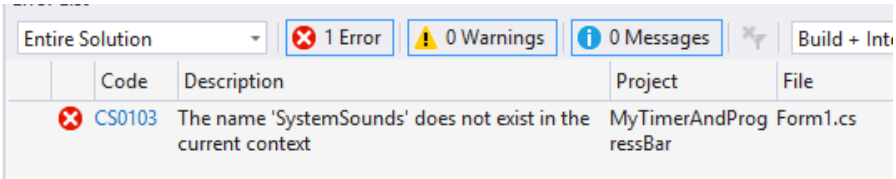
4. Double click on the Button Control and key in the following code:

```
private void button1_Click(object sender, EventArgs e)
{
    timer1.Interval = 100;
    timer1.Enabled = true;

    progressBar1.Value = 0;
    button1.Text = "Timer Running";
    button1.Enabled = false;
    trackBar1.Enabled = false;
}
```

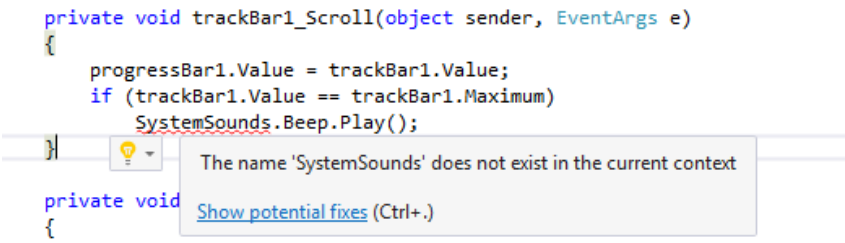
5. Double click on the TrackBar control and type in the following codes:

```
private void trackBar1_Scroll(object sender, EventArgs e)
{
    progressBar1.Value = trackBar1.Value;
    if (trackBar1.Value == trackBar1.Maximum)
    {
        SystemSounds.Beep.Play();
    }
}
```

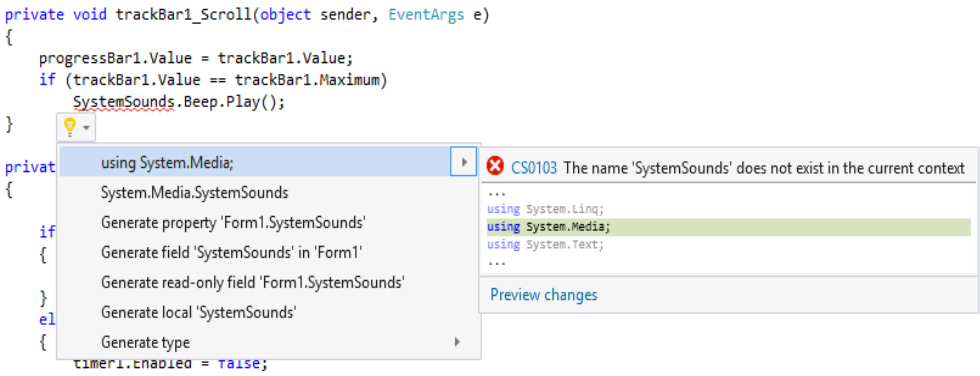
6. Build the project. Take note that ***SystemSounds*** is highlighted with the following error:



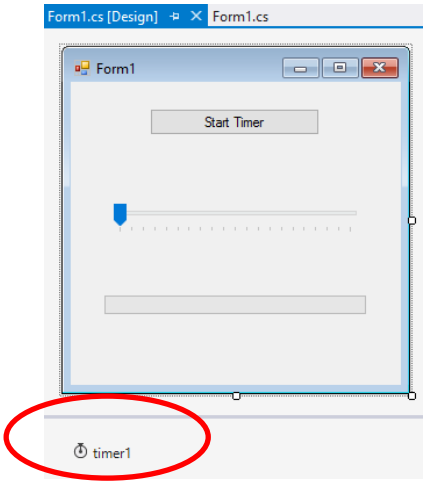
7. The error occurs due to missing of ***using*** statements in the project. Hover the mouse somewhere at the error statement, Visual Studio Intellisense can automatically suggest the right ***Namespac***e to be included using the ***using*** statement.



8. Click on the "Show potential fixes", and choose adding "***using System.Media".***



9. Next double click on the Timer Control and key in the following codes:



```
private void timer1_Tick(object sender, EventArgs e)
{
    if (progressBar1.Value < 1000)
    {
        progressBar1.Value += 50;
    }
    else
    {
        timer1.Enabled = false;
        button1.Text = "Start Timer";
        button1.Enabled = true;
        trackBar1.Enabled = true;
    }
}
```

10. Build and run the application and next analyze the codes that we have added

No	Actions	Observation / Action
1	What is the role of the ***using*** keyword and explain the use of ***Namespaces***?	

2	Let's examine how timer works?
	`private void button1_Click(object sender, EventArgs e)`
	`{`
	`timer1.Interval = 100;`
	`timer1.Enabled = true;`
	`. . .`
	`}`
	`private void timer1_Tick(object sender, EventArgs e)`
	`{`
	`. . .`
	`timer1.Enabled = false;`
	`}`
	What is the role of timer1_Tick?
	
	How frequent does the timer object triggers? How to change it to every 1 second?	

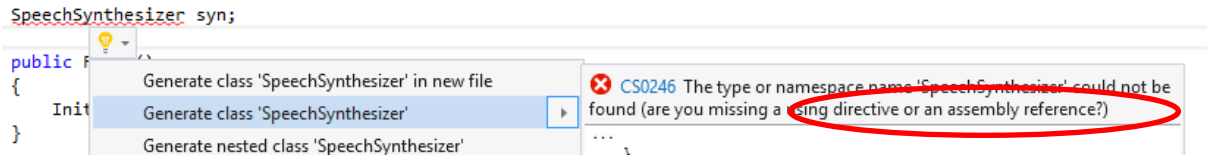
3	`private void trackBar1_Scroll(object sender, EventArgs e)`
	`{`
	`progressBar1.Value = trackBar1.Value;`
	`if (trackBar1.Value == trackBar1.Maximum)`
	`SystemSounds.Beep.Play();`
	`}`
	What the code does?	

Adding Speech Capability using System.Speech Library (Assembly)

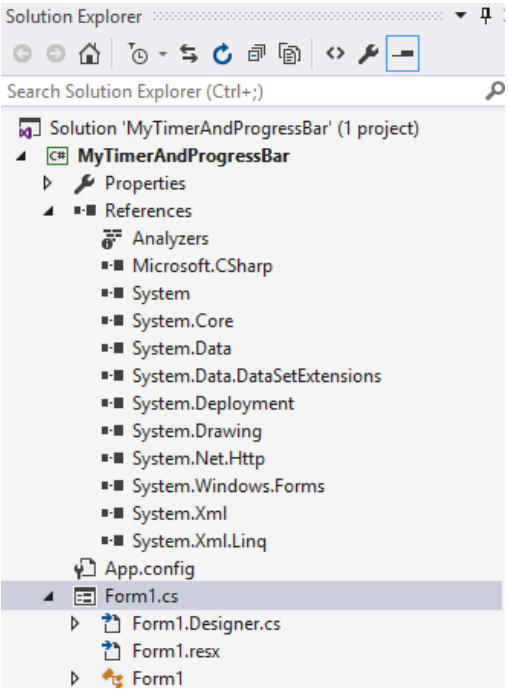
11. This part continues from the previous project that was built in Part 3. If the project was closed, under the ***File*** menu, click ***Open->Project/Solution*** and navigate the folders to select ***MyTimerAndProgressBar***.
12. In order to use speech capability, we need to make use of the ***SpeechSynthesizer*** class in the System.Speech library (assembly).
13. First let's declare and instantiate an object of type ***SpeechSynthsizer***. Observe that there Visual Studio will highlight as an error. Use Intellisense to try to resolve the error.

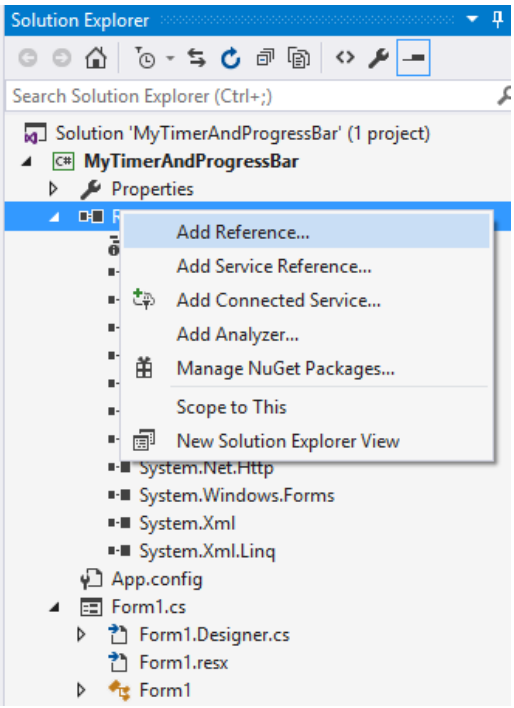
```
public partial class Form1 : Form
{
    SpeechSynthesizer syn = new SpeechSynthesizer();

    public Form1()
    {
        InitializeComponent();
    }
```

14. Noticed that Intellisense does not offer any Namespace to be added. However in the hint "are you missing a using directive or assembly reference?" suggested that the speech library is not added to the project.

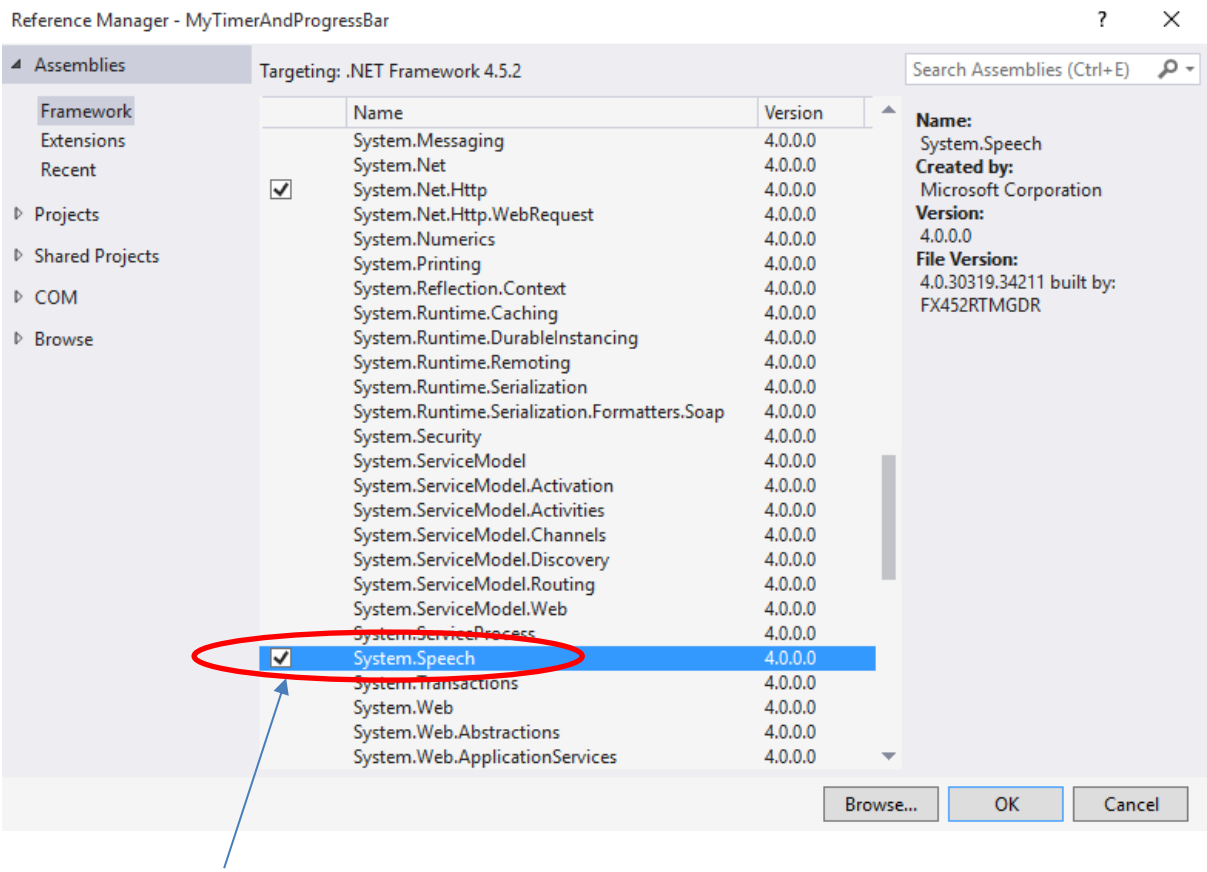


15. The libraries (assemblies) used in the project can be seen under ***References*** in the ***Solution Explorer***.





16. From the ***Solution Explorer*** right click ***References*** node and select ***Add Reference*** option. That will launch the ***Reference Manager***. Under ***Assemblies->Framework***, scroll to checked ***System.Speech*** and click **OK** button to add the speech library.



Important to click on the **Checkbox** to check the System.Speech

17. Use Intellisense to try to resolve the error now. Noticed that now the Intellisense will propose adding of a **Namespace** ***(System.Speech.Synthesis)*** to resolve the error. Select that option and proceed.
18. Now modify the *Trackbar* scroll event handler function:

```
private void trackBar1_Scroll(object sender, EventArgs e)
{
    progressBar1.Value = trackBar1.Value;
    if (trackBar1.Value == trackBar1.Maximum)
    {
        SystemSounds.Beep.Play();

        syn.Speak("The track bar has reached the maximum");

    }
}
```

19. Build and run the application. Congratulations, you have just added speech capability to your application.