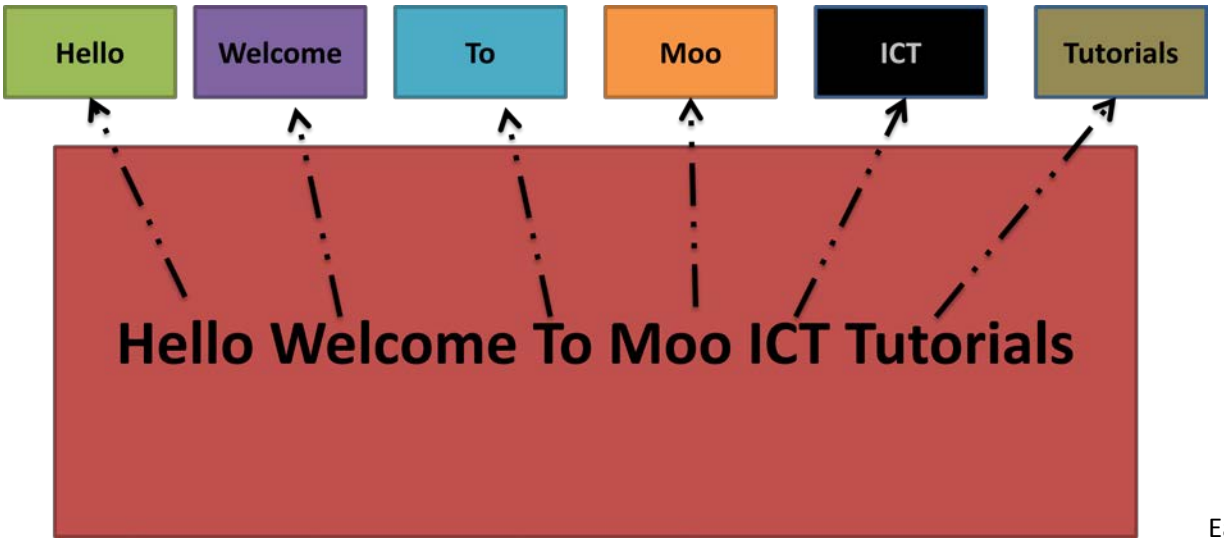
C# Tutorial - Create your own Speed Reading Software

Welcome to this exciting tutorial where we will create own Speed Reading Software. If you don't what it is, let's get to know it.

Speed reading is a number of techniques which allows us to read faster. Traditional ways of reading is good but we are always looking out for better ways to read. There are various software which helps with this process for example **www.speeder.com** or **www.spritz.com**. These websites take a whole article and separate them into words and each word comes up on the screen one at a time. You can control the speed of words. Few software are available which trains people to read faster which is awesome but for this tutorial we will focus on RSVP (**Rapid serial visual presentation**) functionalities. For more information you can read on RSVP here - **https://en.wikipedia.org/wiki/Rapid_serial_visual_presentation**

Below is a visual representation of the idea.

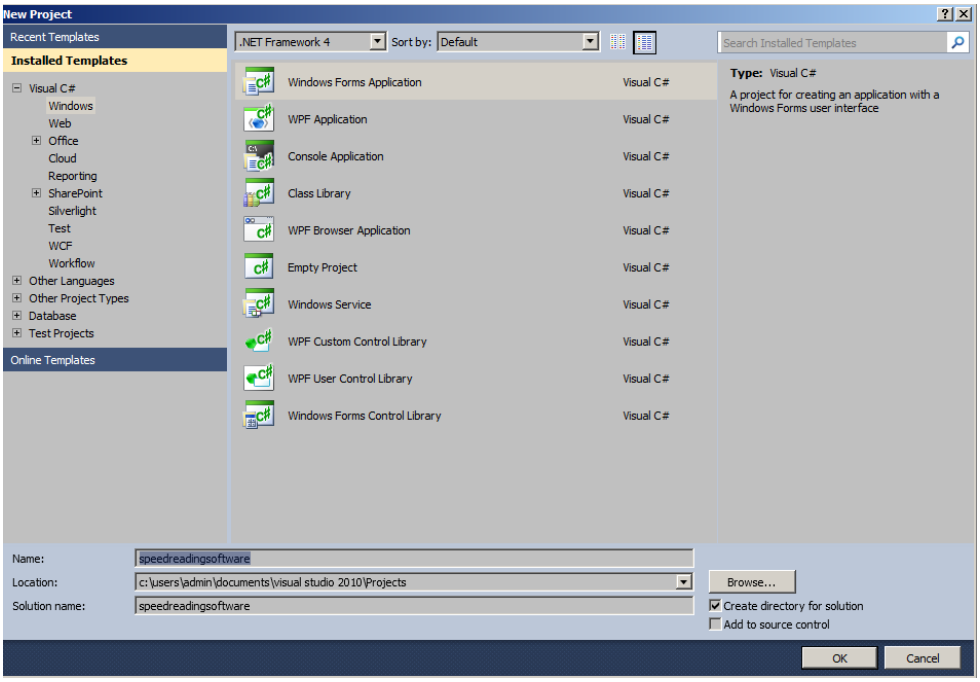


Each of the words will be separated and shown one at a time to screen which makes it easier for us to read. Now this example is showing only one line of text imagine you had a whole 200 lines to read. Would this software be helpful? I would think yes.

Let's get started on the tutorial then

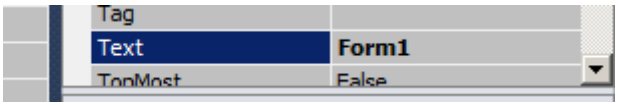
Start a new project in visual studio

Select Visual C# -> Windows Form Application

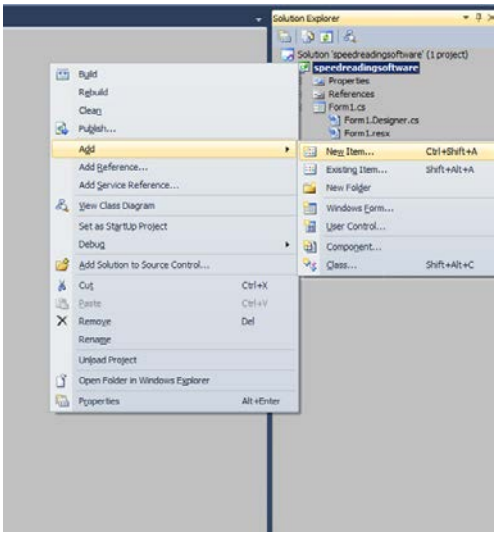


Call the project speed reading software no spaces.

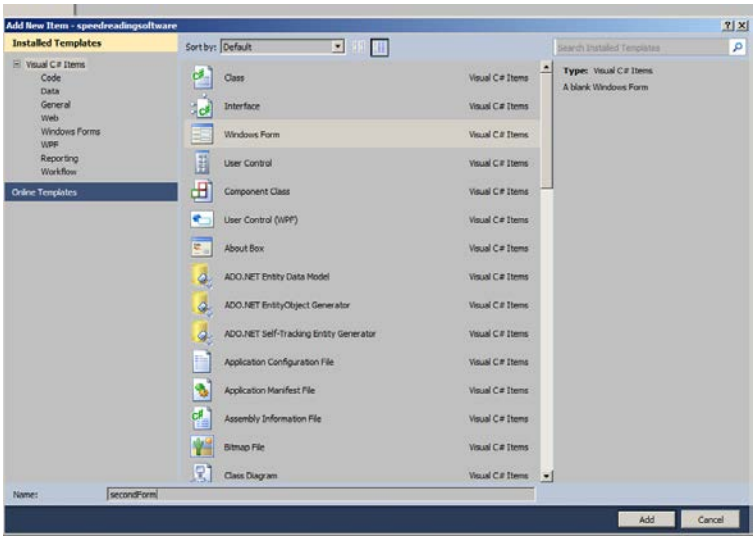
Change the title text of form1 to "Speed Reading Software"



We will add another form to the project



Right click on the project name in the solution explorer and hover on Add then click on Windows Form



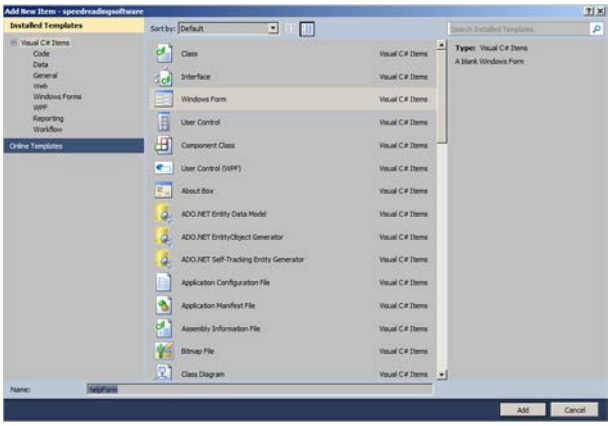
Call it **secondForm**.

Click Add

Now lets add the third form to the project. This one will be our help screen.

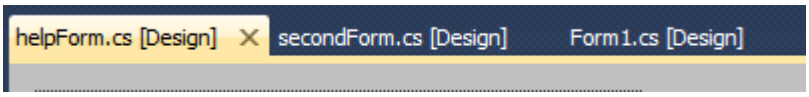
Once again right click on the solution -> add -> windows form

Name it helpForm



Click add.

Now you have three different forms to work with.



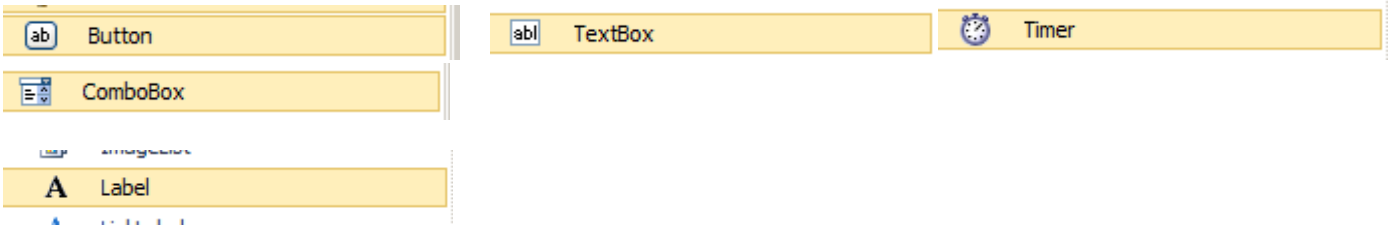
The first form where we will do the set up and the second form will display the text for reading.

We will work on Form1 for now.

Lets add the following components to Form1

Component	Name	Purpose
Button	startButton	Start the process of speed reading
Button	helpButton	To show a help screen, which will help the user to use this system more reliably.

Text Box	mainText	This is where we will paste our articles
Combo Box	speedCombo	This will contain a number from 1 - 9 which will then increase of decrease the speed. Lower number means faster and higher means slower.
Timer	readingTimer	This will animate the label on second form to bring one word at a time.
Label	Label1(leave it defult)	This will indicate the user where to change the speed



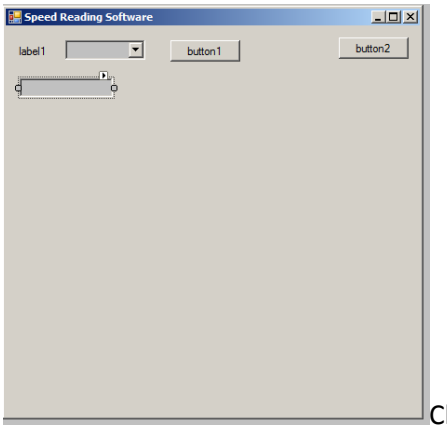
We will get to the second form when we have done with form 1.

Before we begin we need to change the size of the form .

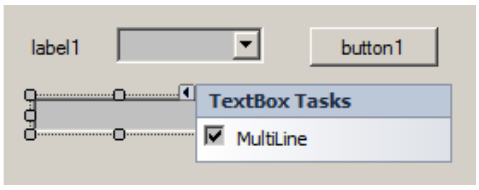
Click on the form and go into properties window you will find the size of the form change the size to the following

453, 446

Here is the outline of our project



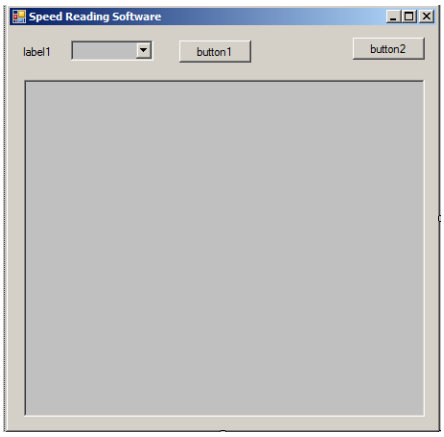
Click on the text box, see that little white triangle on the top right corner click on it for more options.



Click on Multi line this will make it easier for us text large amount of text in to it.

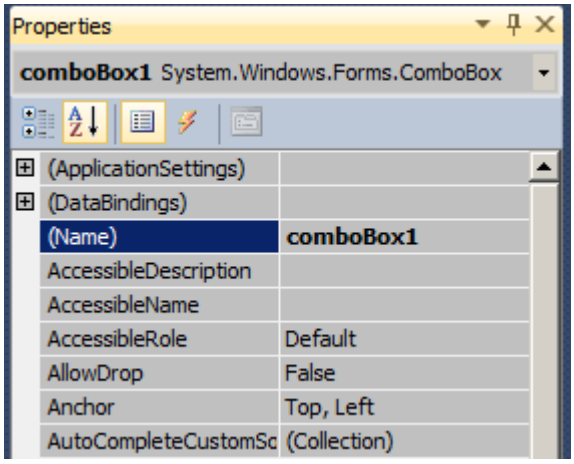
Now lets add the timer to the form. Drag and drop the timer to the form.

Click on the timer and change the name to readingTimer.



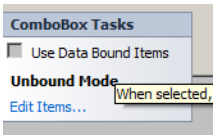
Now let's make some changes to the components.

Start with the combo box right click on it and click on properties.

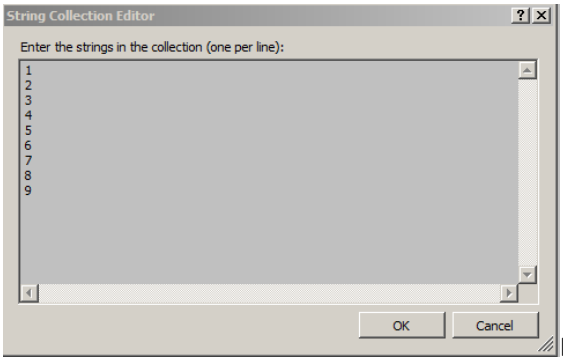


We need to change the name to **speedCombo** and press enter.

Click on the little triangle on top of the combo box.



Click on **Edit Items**



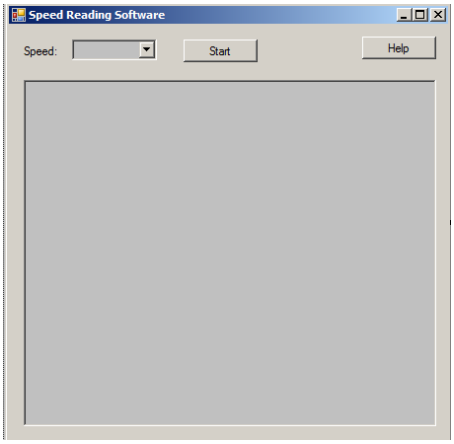
Enter the number from 1 - 9 in the box. Enter each as a new paragraph.

For button1, change name to startButton. After the change the text of the button to Start.

For button2 change name to helpButton. After this change the button text to Help.

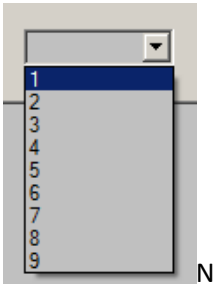
text box change name to mainText

Change the label1 text to Speed:



This is the final view of the form1.

If you run this form now and click on the combo box it will show the numbers like this.



Nice right.

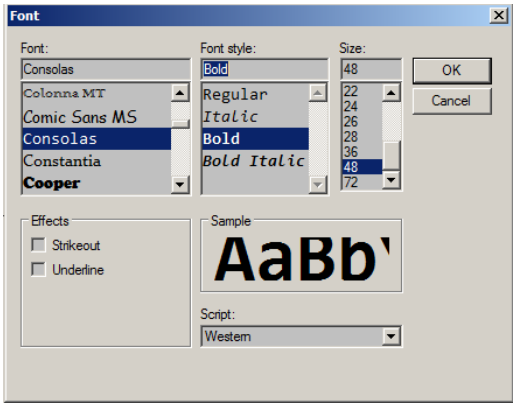
Now to the second form.

This form will be used to read the changing text on the screen. There is only one component on the screen which is the label.

Lets add a label to the screen.

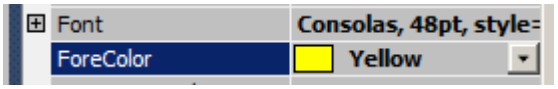
Now click on the form and change the size to **1200**, **500**.

Click on the LABEL1 in the properties window click on the fonts option you can spot it by the 3 dots (...)



Change it to the following settings.

Change the fore colour of the font to Yellow.

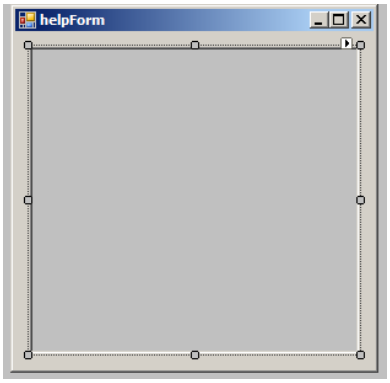


Now one last time we need to change the properties of the form. Click on the form and look for the back colour option in the properties window.

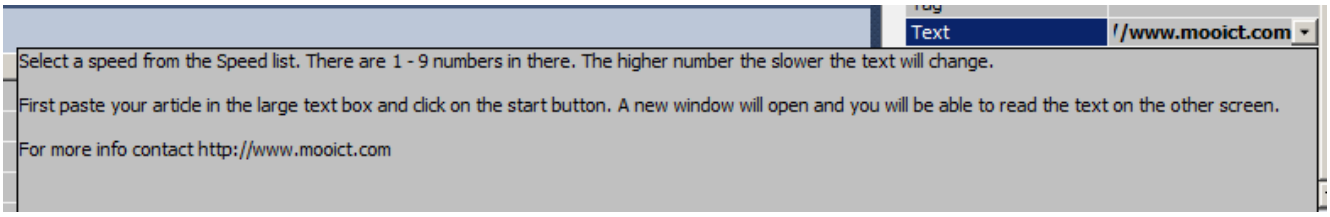
This is what our final design of the second form.

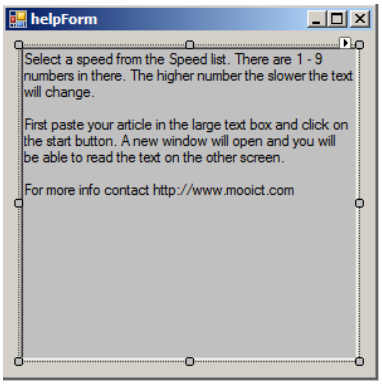
Now let's go to the help screen.

In this screen we will have multi line text box. Where we will explain how to use the app.



Find the text option in the properties window and enter the following text in the help screen.





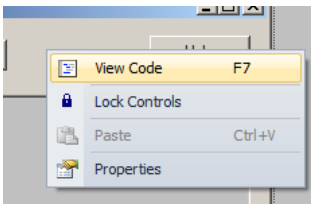
Select a speed from the Speed list. There are 1 - 9 numbers in there. The higher number the slower the text will change.

First paste your article in the large text box and click on the start button. A new window will open and you will be able to read the text on the other screen.

For more info contact http://www.mooict.com

Now we are in the coding phrase

to get to the code view right click on the **form1**.




```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace speedreadingsoftware
{
    public partial class Form1 : Form
    {
```

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

This is the code in your program so far. This is a empty program it will not do much expect just load the first form in to the display.

Let's start adding the codes for the program right now.

We need to add our own variables and link the other two forms together with this.

One the line above the `public Form1()` we will declare our variables

```
        public string words;
        public string[] splitup;
        public int totalWords;
        public int counting = -1;
        public int intervalSetting = 0;
        secondForm readingScreen = new secondForm();
```

Notice we are putting them in as public variables because we will use them in different functions. Lets start with the first one.

1. First variable is called words which is a data type of string. This string will be used to capture all of the data from the text box where the user can paste the information to be displayed later.
2. second variable is an array data of string. Notice that [] brackets after the string declaration. That means we can have multiple variables in one. This variable will be used to split up the article from the words variable.
3. Total words variable is a data type of an integer and it will be used to count the amount of words inside the split up.
4. Second integer we have is called counting. This will keep track of how many words we have read so far into the article. Notice we are starting the variable from -1. We will explain that later.
5. Third integer is called interval setting. This variable will be used to change the speed of the reading. This will be linked to the combo box and it will help to either speed up or slow down the text on the screen.
6. Lastly we are linking the second form to this form. This function will allow us to control the components on the other form and make it visible and invisible.

Let's take a look at how we are going to use the array to read a single word on the screen.

Notice when we mentioned the counting variable we have started from -1. This will make some sense now.

Imagine this is our words array now

Welcome to moo ict website

This whole sentence is a single sentence. Now let's imagine we have split it in the array.

Welcome	To	Moo	Ict	website

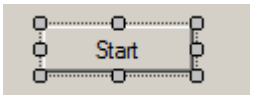
Now instead of one variable we have 5 of them.

The way arrays work is they give each of them an index value which means it starts counting from 0 to end of the array which is 4. counting from 0 to 4 is 5 digits (including the 0).

0	1	2	3	4
Welcome	To	Moo	Ict	website

If we want to check what is in the 3rd index of this array we can say splitup[2], it will return MOO. If we want to check the 5th index of the array we can check it by using splitup[4] and it will return WEBSITE.

Hopefully this clears things up a bit. Now let's get to adding some events to our buttons to make it more interactive.



double click on the start button.

```
        private void startButton_Click(object sender, EventArgs e)
        {

        }
```

In between the brackets add the following code.

```
        private void startButton_Click(object sender, EventArgs e)
        {
            words = mainText.Text;

            splitup = words.Split();

            totalWords = splitup.Length;

            int value = Convert.ToInt32(speedCombo.Text);

            readingTimer.Interval = 100 * value;

            readingScreen.Show();

            readingTimer.Start();
        }
```

First we are giving the variable words a value from the mainText.Text which means the text box values. Then we are using the splitup variable and splitting the words into single strings. words.Split() is a function from the string class which basically will split up all the whole string into single words and disregard the space and whitespaces in the article. It does the job for us.

total words variable is being used after we have split up the words variables and then we count the length of the array. We are using splitup.Length function, it returns the number of arrays inside it.

Int value is being created inside the function. This is a local variable, we won't be using it outside this function so it doesn't need to be public. This open will store the value from the combo box. We have the number 1 - 9 inside it. We are converting the combo box string to integers to use it as the timer interval.

reading timer is our main timer for this software, the value integer we created before we will be using it to multiply the number and it will determine what speed to use. For example by default the timer runs every 100 milliseconds, if the user chooses 3 from the combo box we then multiply 3 x 100 = 300 now then timer will run every 300 milliseconds. In essence the text will change every 300 milliseconds.

Reading screen show function will bring up the second form to the screen by using readingScreen.Show().

lastly we are going to start the timer by using readingTimer.Start().

double click on the timer.

```
        private void readingTimer_Tick(object sender, EventArgs e)
        {

        }
```

enter the following code inside the brackets.

```
        private void readingTimer_Tick(object sender, EventArgs e)
        {
            counting++;

            if (counting == totalWords || readingScreen.isitCLosed == true)
            {
                readingTimer.Stop();
                MessageBox.Show("End of article");
                counting = -1;
            }

            else
            {
                readingScreen.label1.Text = splitup[counting];
            }
        }
```

Inside this function we have a if statement running. The main idea is we want to know when the article has finished so the program stops running. If we make it continue to run it will crash because the array will be empty after a certain point. Remember when we counted the total words. This is why. Also remember when we started the counting variable from - 1, that was because arrays always start from 0 so if we started from 0 it will automatically jump to one so we would have missed the first word of the article.

Now in the beginning we are increasing the counting variable by 1 by adding ++ end of the variable each time it runs it will add 1 to the total number inside it.

Second we started the if statement, this statement means if counting is equals to total words then do the following actions. Since the total words will have the total number of single words inside it stored we can check the counting with it which is increasing by 1 each time the timer runs.

Lets break this if statement down a little to make more sense out of it.

First we are declaring the IF followed by the brackets next to it.

Inside the brackets we have the conditions which must be met in order for the inside actions to be taken. Inside this if statement we have two different condition and this is how they are read:

`counting == totalWords`	`||`	`readingScreen.isitCLosed == true`
Counting is equals to total words	OR	Reading screen is it closed boolean is equals to true

So both of the conditions don't have to meet at the same time, if one or the other meets we do the same thing.

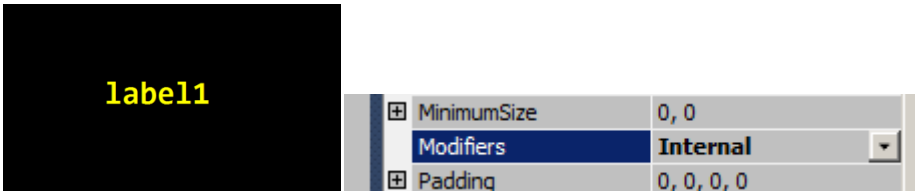
Once either of the conditions are met we are stopping the timer first, then we are going to show the message box and then we are resetting the counter to -1.

Now we move on the else statement, if the first condition hasn't been met we move to the else statement. In this one we continue to show the words coming up on the second form or the reading screen.

By importing the form as we did before we are now able to control the components however you will notice we have highlighted the label1.Text on the code. This is because there is a permission issue between the two forms we need to fix. In your code it should have a red line under the label1. This isn't a major issue we will have to change the modifier value in the second form. In this line of code we are simply adding the split-up array to the label1. Since split up is an array instead of changing the words manually we are changing it dynamically with the counting variable.

Let's move on to the second form.

In this form we only have one label on the screen. Click on the label and go to the properties window.



Change the modifiers to **internal**. This will enable the label to be controlled by the first form.

Now let's go to the code view for the second form.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace speedreadingsoftware
{
    public partial class secondForm : Form
    {
        public secondForm()
        {
            InitializeComponent();
        }
    }
}
```

This is what the code looks like so far. We need to a Boolean variable to the second form. As you remember we did call a variable called is it closed before in the IF statement. That variable belongs in this code. This Boolean will change between true or false once the window has been closed by the user.

Above the line public secondForm() add the following Boolean:

```
        public bool isitCLosed = false;

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

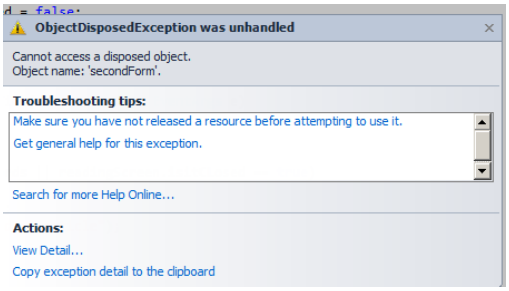
Now go back to the design view of the second form.

Before we go adding the next line of code let me show you something that will help you understand why are we doing it.

Do the following steps

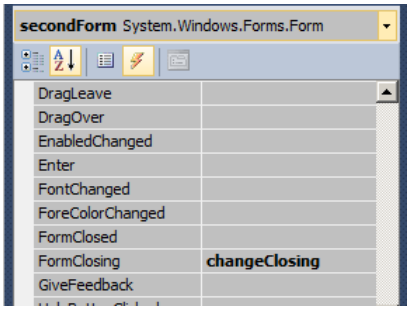
1. Run the program
2. Type few words into the text box
3. Select a speed from the combo box

4. Click on start
5. Once the text finished reading close the second screen
6. Go back to the first screen click start again and BOOM



Some how there is a error for running the program again. Now this is a use case error because if the second form is closed by the user it gets disposed from memory and something dead cannot be resurrected just like that. So initially this is a bug in the system we need to fix and kind of a big one because we cannot let the user run and re run the program to read things right. Let's make it easy for them.

Now click on the form and click on the events window. It's the lightning bolt icon by the properties window.



Find the form closing section and type changeClosing and press enter.

```
        private void changeClosing(object sender, FormClosingEventArgs e)
        {

        }
```

Lets add the following code inside this function. Between the brackets.

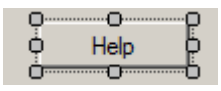
This is the function where we will tell the program what to do when the user has clicked to close the window.

```
        private void changeClosing(object sender, FormClosingEventArgs e)
        {
            //Hiding the window, because closing it makes the window inaccessible.
            this.Hide();
            this.Parent = null;
            e.Cancel = true; //hides the form, cancels closing event
            isitCLosed = true;
        }
```

Instead of closing the window completely and removing it from the memory we will hide it from view. Also we are cancelling any events that might of been triggered so the data is still safe on memory. Until the user closes the main window the program will continue to run. Lastly we will change the Boolean is it closed to true so we can run the function inside the if statement to reset the counter and stop the timer itself.

Help Screen

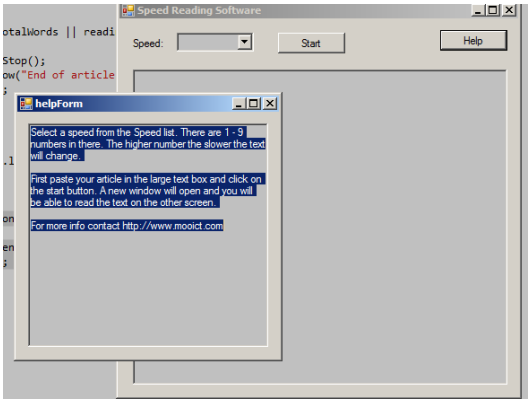
Now go back to the first form and double click on the help button



```
        private void helpButton_Click(object sender, EventArgs e)
        {
            helpForm helpScreen = new helpForm();
            helpScreen.Show();
        }
```

Add the highlighted code in the function. First we are creating a new instance of the help form we created earlier inside the first form. Each time the button is clicked it will create an instance and show it on the screen.

Try it out.

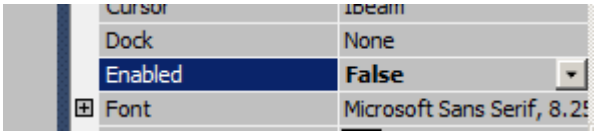


You will notice that the text is editable in the box. Which you can leave it as it is or you can simple use the disable function in the properties window.

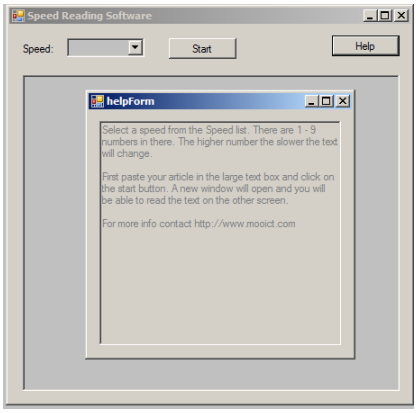
Go to the help form design view

Click on the text box

check the enabled option in the properties window



Change it false. Now run the program and click on the help screen again.



There you go it's not editable now.

Well done on your software development. Hopefully you have learned a thing or two. The full code is available on Mooict.com tutorial page.

See you soon.