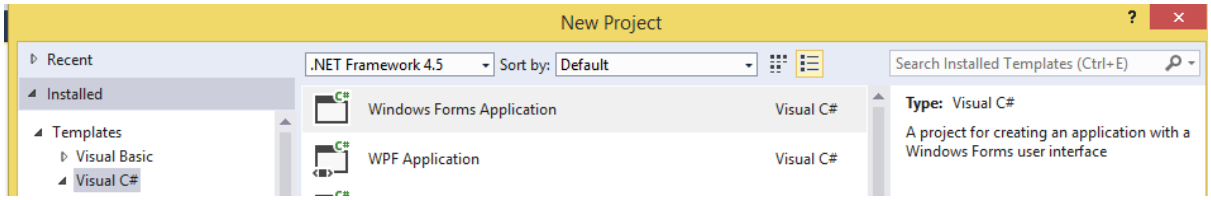
EXERCISE 1

The tasks in the first part of this section provide step by step instructions on how to program in C# using Graphical User Interfaces (GUI's). This is followed by a number of tasks for you to complete by applying the knowledge that you have previously gained.

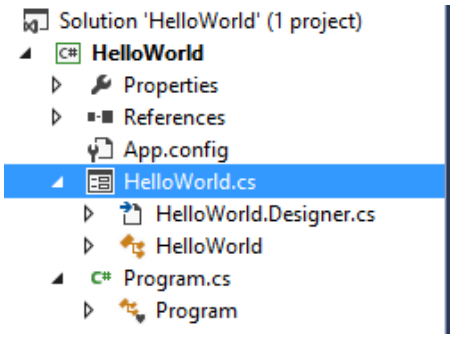
TASK 1: CREATING A SIMPLE GUI PRINTING HELLO WORLD

This task requires to create a GUI which will display "Hello World". We will write a Windows Forms Application to achieve this.

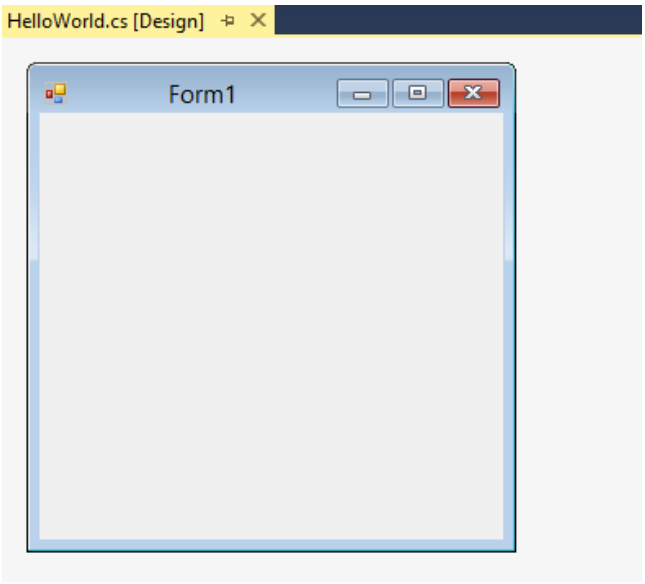
Step 1: Open the Visual Studio File menu, and then select New (or press `Ctrl+Shift+n`). Then click on Project, select C# Windows Forms Application and name it `e.g. Prac06Task1`.



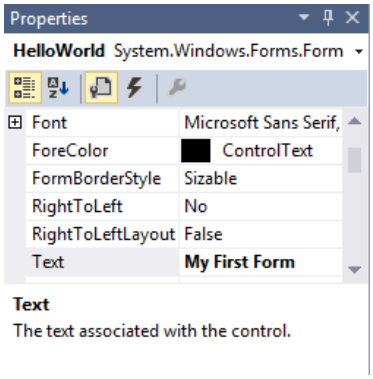
Step 2: Right Click on Form1.cs in the Solution explorer and rename to HelloWorld.cs:

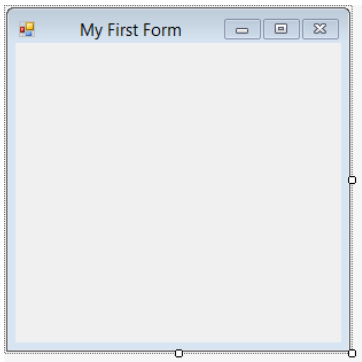


Step 3: Open `HelloWorld.cs` which will display a visual representation of your form as seen below:



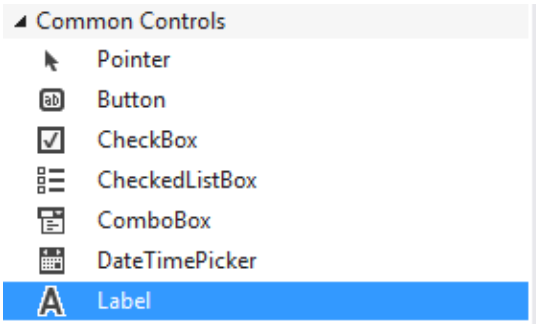
Step 4: The form has a blank default title "Form 1". You can change that by clicking on the form to select it, and then changing the `Text` property in the Properties window.

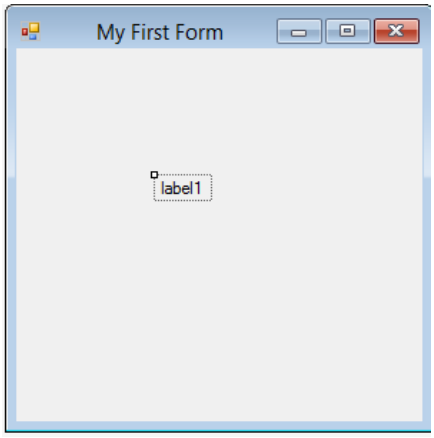




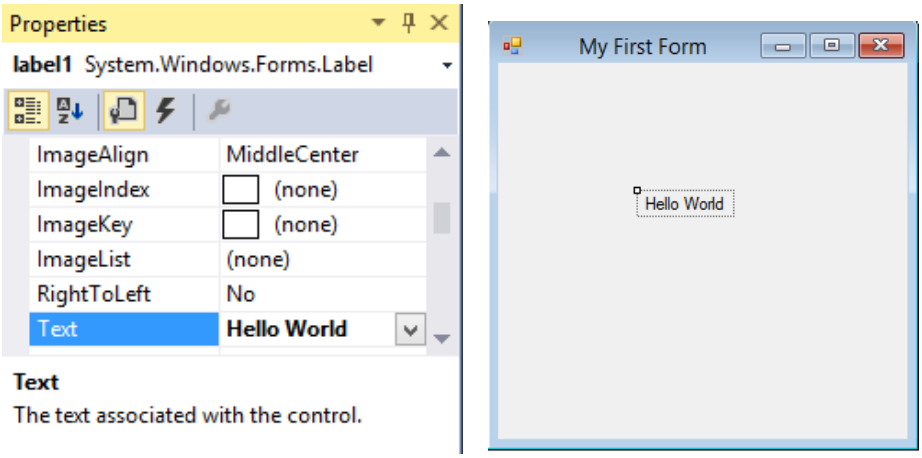
Step 5: On the Menu bar select View and Select Toolbox. The toolbox contains many items and elements you can add to your form e.g. buttons and labels:

Step 6: Using the toolbox, double click on the Label Control which will add a Label object to your form:

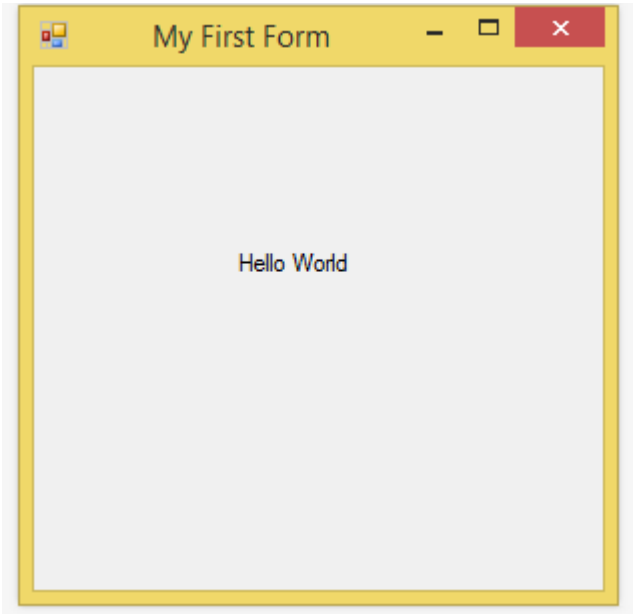




Step 7: Click on the Label on the form and view the properties window to change the text of the label to "Hello World":



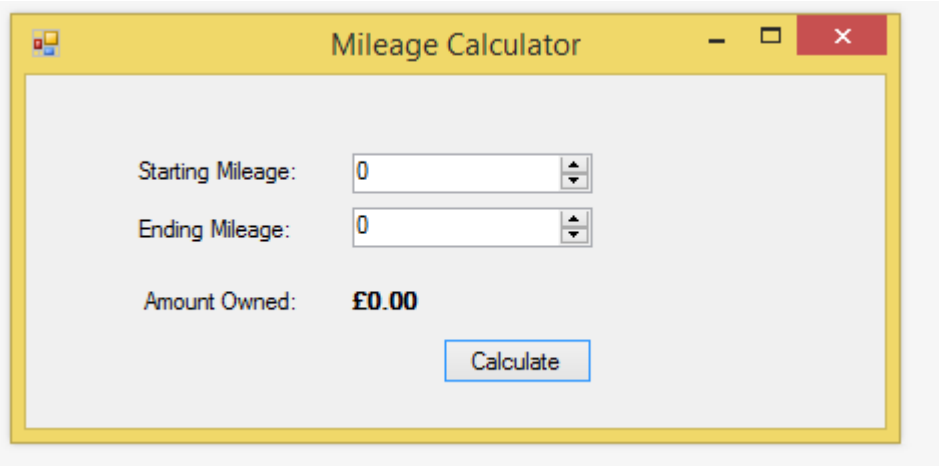
Step 9: Click "Run" or press F5 to run the Windows Forms Application to view your first GUI application:

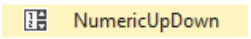


TASK 2: MILEAGE CALCULATOR

This task requires to create a GUI which will create a reimbursement calculator for a business trip. It should allow the user to enter a starting and ending mileage reading from the car's odometer. From those two numbers, it will calculate how many miles the person has travelled and calculate how much the company pays her if the company pays £0.39 for every mile travelled.

Step 1: Create a new windows forms project called `Mileage Calculator` and create the following GUI using the toolbox to add elements to the form e.g. Labels, Buttons. You can reposition any element on the screen by clicking and dragging the element to the desired location:



*Hint: Use the `NumericUpDown` element from the Toolbox for the Miles: 

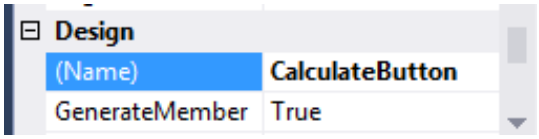
Step 2: To add code to this form, right click on form1.cs and select View Code or press F7 while in the designer view. Create the variables need for the calculator. You will need two whole number values for the starting and ending mileage. Call the variables *`startingMileage`* and *`endingMileage`*. For the calculator, you will need three variables that can hold decimal places and therefore make the variables *doubles* and call them *`milesTraveled`*, *`reimburseRate`* and *`amountOwed`*. Since we know *`reimburseRate`* is 0.39, set this variable to a constant e.g. const.

```
namespace Mileage_Calculator
{
    3 references
    public partial class Form1 : Form
    {
        private int startingMileage = 0;
        private int endingMileage = 0;
        private double milesTraveled = 0;
        private const double reimburseRate = 0.39;
        private double amountOwed = 0;

        1 reference
        public Form1()
        {
            InitializeComponent();
        }

    }
}
```

Step 3: To make your calculator work, we have to add code to your button to perform the calculation. Rename the button to `CalculateButton` using the Name property in the properties window:



Double click on the button in design view which will automatically generate a method which will be called when the button is clicked. This method is where you will place the logic to either output the total amount owed or produce an error message box:

```
1 reference
private void CalculateButton_Click(object sender, EventArgs e)
{

}
```

Step 3: Get the values from the `NumericUpDown` elements and assign the value into our variables for the mileages:

```
//.Value returns a decimal value to use (int) to convert/cast into an int
startingMileage = (int)numericUpDown1.Value;
endingMileage = (int)numericUpDown2.Value;
```

We have to ensure that the number in the *`startingMileage`* field is smaller than the number in the *`endingMileage`* field and if not, show a message box that states "The starting mileage must be less than the ending mileage."

```
if(startingMileage < endingMileage)
{
    //Calculate Logic goes here
}
else
{
    //This will show a message dialog with the message and a title
    MessageBox.Show("The starting mileage must be less than the ending Mileage", "Cannot Calculate Mileage");
}
```

Step 4: Calculate the total miles travelled and work out total amount owed by multiplying the miles by the reimbursement rate i.e. 0.39:

```
//Calculate total miles traveled
milesTraveled = endingMileage - startingMileage;

//Calculate amount owed using Reimburse Rate
amountOwed = milesTraveled * reimburseRate;
```

Step 5: Now we will change the text of the label on the screen to show the amount owed value. First change the name to "`TotalLabel`" and text to "£0.00" of the label element by using the and text properties in the properties windows:

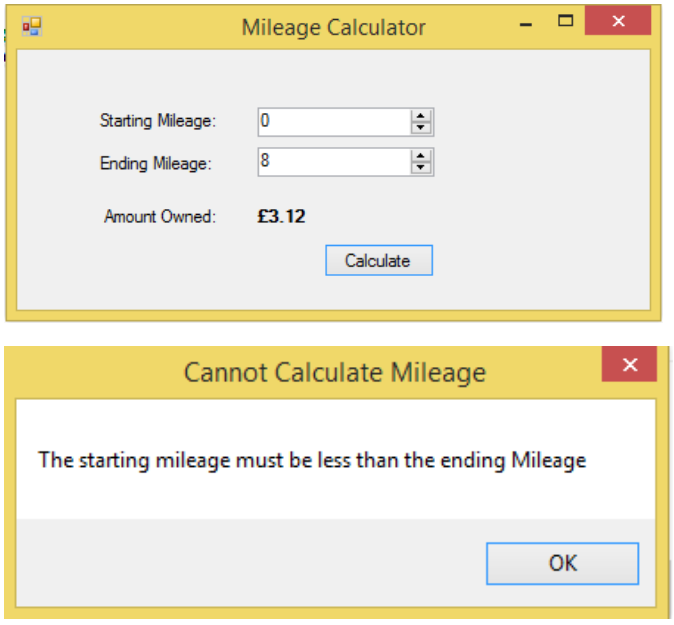
Now using the Labels Text property, show the total amount owed value by assigning the *amountOwed* variable to it:

```
//Assign amountOwned to the text of the label with a £ sign
TotalLabel.Text = "£" + amountOwed;
```

Step 6: Run the application by selecting Start or pressing F5, select a starting and ending mileage and click the Calculate button.

Remember to select the starting mileage more than the ending mileage to check if the error message box appears.

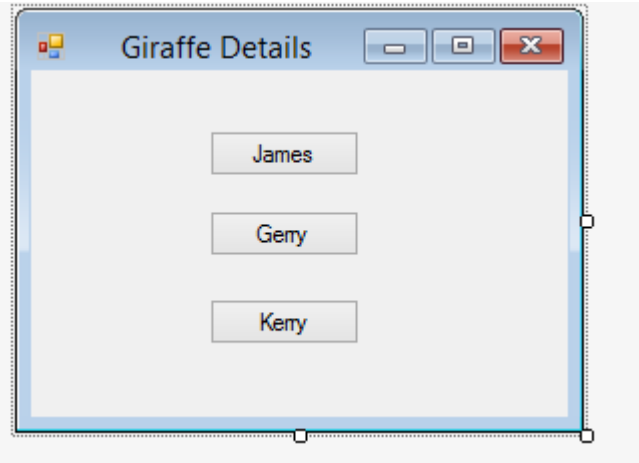
GUI Output:



TASK 3: GIRAFFES

This task requires to create a GUI which will show the details of an Elephant when their button is clicked. This will require a Giraffe class to be created.

Step 1: Create a new windows forms project called Giraffes and create the following GUI using the toolbox to add elements to the form e.g. Buttons. You can reposition any element on the screen by clicking and dragging the element to the desired location. Remember to use the Text and Name properties to correctly name your buttons and change the title of your window:



Step 2: Create a Giraffe Class which will contain 3 variables to store details about the Giraffe. You will need a double called Height, a string called Name and an `int` called `Age`. Your giraffe class will also have a method to show the details of the giraffe in a Message Box like Task 2 and a constructor which will initialise the object with details. You can refer to the previous weeks to see how to create your own classes. A class diagram is shown below:

Giraffe
int Age string Name double Height
WhoAmI()

*HINT: You have to use `System.Windows.Forms` in the Giraffe Class to use the `MessageBox`.

Step 3: Create three Giraffe instances in the Form1.cs class named James, Gerry and Kerry. Use the Giraffe's constructor to create each giraffe object:

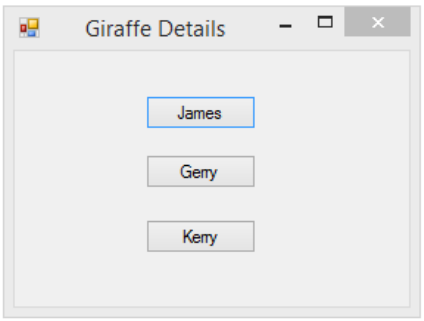
```
//Create the Giraffe Objects using a constructor
James = new Giraffe("James the Giraffe", 6.2, 10);
Gerry = new Giraffe("Gerry the Giraffe", 5.8, 8);
Kerry = new Giraffe("Kerry the Giraffe", 6, 9);
```

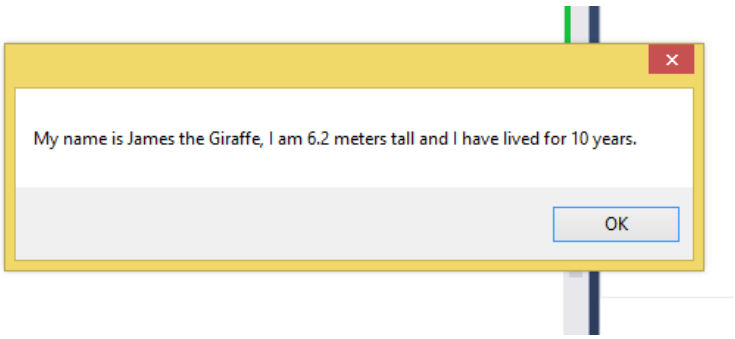
Step 4: Add methods behind each one of your buttons by double clicking on each to generate the click method event. An example method is shown below:

```
private void JamesButton_Click(object sender, EventArgs e)
{
    //Generated Button Event Method which will execute when the button is clicked
}
```

Step 5: Now add a call to the Giraffe's `WhoAmI()` method when the button is clicked to show the Message Box.

The expected output is shown below:





TASK 4: WHAT'S THE OUTPUT?

You should try to work this out on paper and then transfer the code into Visual Studio and run it to see what output you actually get. Remember that if you don't get the answer you expect to check both your workings on paper and the code.

a)

```
string[] Animals = { "Giraffe", "Dolphin", "Elephant", "Dog", "Cat" };

for(int i = Animals.Length - 1; i > 0; i--)
{
    Console.WriteLine(Animals);
}
```

b) What values could be printed if you ran this code multiple times:

```
Random random = new Random();

Console.WriteLine(random.Next(5))
```

EXERCISE 2

TASK 5: SLOPPY JOES SANDWICHES

This task requires to create a GUI which will show a random menu for Joe every day. The menu will contain 6 different sandwich options that will be randomly generated each time the program runs.

Step 1: Create a new windows forms project called Sloppy Joes and create a `MenuMaker` class. For a menu you need ingredients and for Joe's sandwiches we will need Meats, Condiments and Breads and arrays would be perfect for the ingredient lists.

Another thing we will need is a way to randomly choose ingredients to combine together into a sandwich. Luckily you do not have to create one as C# has a built in class called *Random* that generate random numbers.

Create each array by assigning the value to each position as shown below:

```
string[] Meats = new string[6];
Meats[0] = "Roast Beef";
Meats[1] = "Salami";
Meats[2] = "Turkey";
Meats[3] = "Ham";
Meats[4] = "Patrami";
Meats[5] = "Pulled Pork";
```

Or use a collection initialiser to create the array by passing call the values in the declaration as shown below:

```
class MenuMaker
{
    Random Randomiser = new Random();

    string[] Meats = { "Roast Beef", "Salami", "Turkey", "Ham", "Pastrami", "Pulled Pork" };
    string[] Condiments = { "yellow mustard", "brown mustard", "honey mustard", "bbq sauce", "mayo", "relish", "french dressing" };
    string[] Breads = { "rye", "white", "wheat", "italian bread", "pitta", "wrap" };
}
```

The class will use three arrays to story the list of meats, condiments and breads and use a Random object that will generate random numbers.

Step 2: Create a `GetMenuItem()` method to your `MenuMaker` class that generates a random sandwich. The Random object's `Next()` method will choose a random meat, condiment and bread from each array. When you pass an int parameter to `Next()`, the method returns a random number that's less that the parameter but greater or equal to 0.

For example if the Random object is called Randomiser, then calling `Randomiser.Next(7)` will return a number between 0 and 6.

So how do you know what parameter to pass into the `Next()`? Well that's easy, just pass in each array's `Length`.

This method will return a string that contains a sandwich string built from three variables using random elements in the three arrays.

An example for a random meat is shown below:

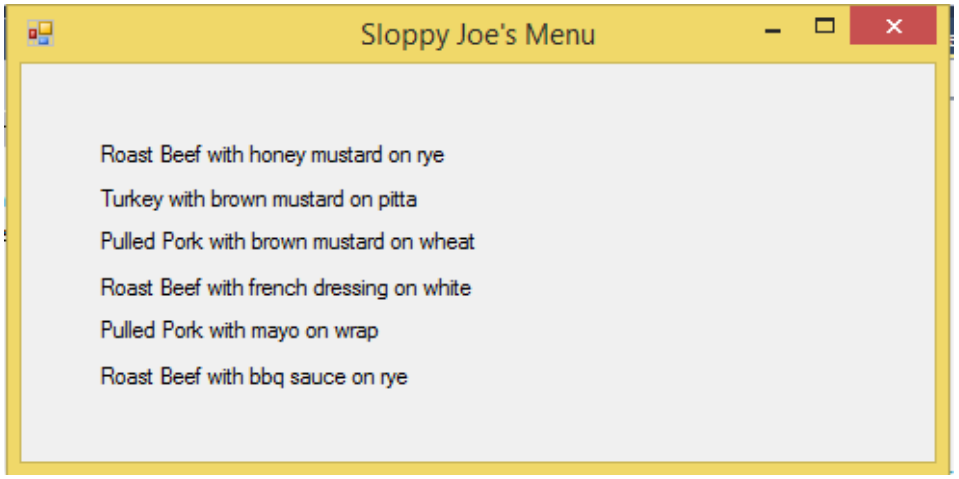
```
//Select a element from the Meats array by selecting the index by using the Random object
//to randomly generate a number between 0 and the length of the array
string randomMeat = Meats[Randomiser.Next(Meats.Length)];
```

Now add a `randomCondiment` and `randomMeat` variables and concatenate all three to create a random Sandwich string to return to be shown on the menu.

For example, the method puts a random item from the Meats array into `randomMeat` by passing `Meats.Length` to the Random object's Next() method. Since there are five items in the Meats array, `Meats.Length` is 5, so `Next(5)` will return a random number between 0 and 4.

Step 3: Now it's time to build the form that will generate the random Menu. Add six labels to the form, Sandwich1 through Sandwich6 using the toolbox and properties window and call to the `MenuMaker` class to generate the random sandwich.

The GUI is shown below showing random menus when ran multiple times:



TASK 6: CAN YOU SPOT THE ERROR?

a) NameLabel.Text() = "Matthew";

b) Message.ShowBox("This is the Box's Description", "This is the Box's Title");

c) AgeLabel.Text = 16;

d) Messagebox.Show("This is the Box's Description", "This is the Box's Title");

e) int[] numbers = {4,6,8,4,1,8};

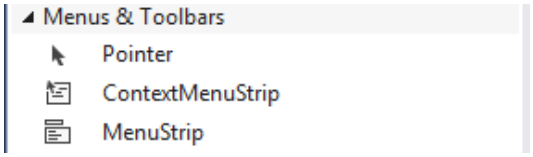
```
for(int i = 0; I < numbers.length, i++)
{
        Console.writeLine("Number: " + i);
}
```

TASK 7: Simple Menu GUI

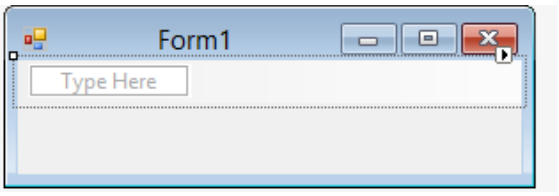
This task will show you how to add a Simple Menu and add several menu items to the menu.

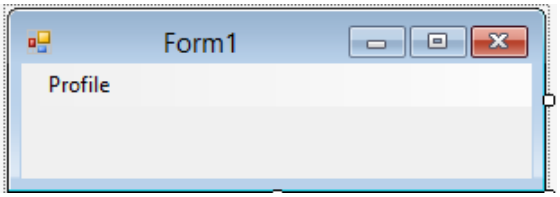
Step 1: Create a new windows forms project called `SimpleMenu` and create the following GUI. Use the toolbox to add a `MenuStrip` to your form i.e. either double click on the `MenuStrip` or drag onto the form.

Click on the MenuStrip to add each Menu Item as shown below:

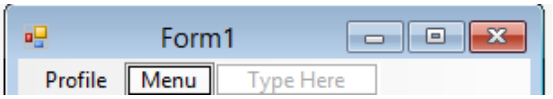


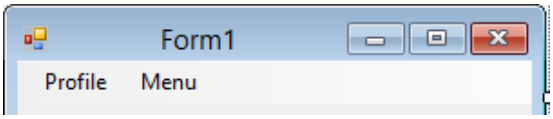
Now you can add an item to the Menu Strip by typing into the placeholder as shown above. Add a Menu Item "Profile":





Add another menu item by repeating the above step by clicking into the bar to add a Contact menu item:

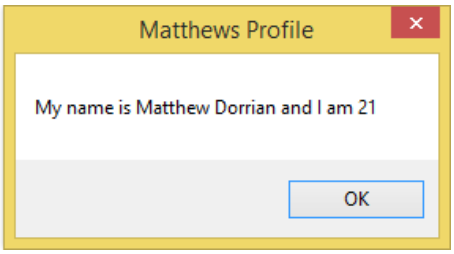




It is time to add a click event to the menu option that gets called when you click on the "Profile" menu option. Double click on the Profile button which will automatically generate the method in the code behind. This is shown below:

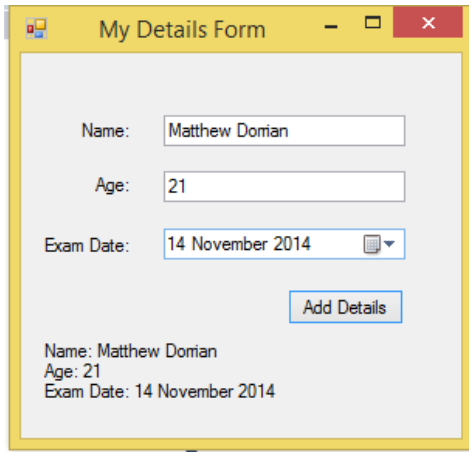
```
private void profileToolStripMenuItem_Click(object sender, EventArgs e)
{
    //Automatically generated button click event method
}
```

Now add code to show your name and age in a Message Box as shown below:



TASK 8: Simple GUI with Menu

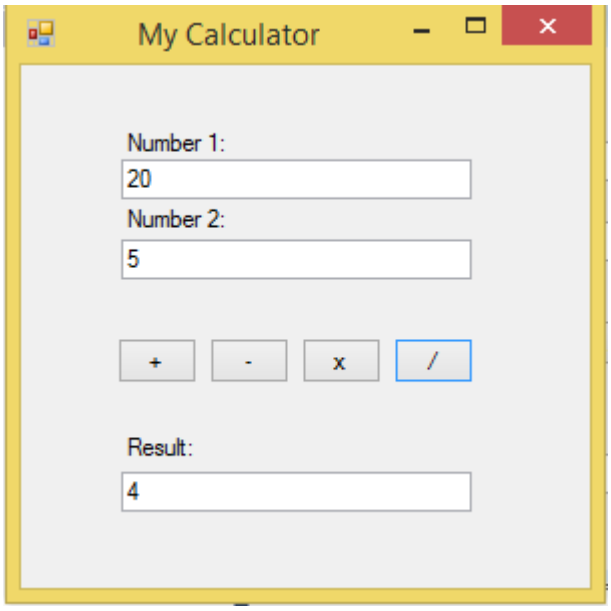
Create a new project called `MyGUI`. Create a form that allows the User to enter in their name, age and the date of their next birthday (Hint: Use a `dateTimePicker` and Textboxes present in the toolbox). Add a button that when clicked displays the details on the form using a Label. Your output should look like below:



Can you now alter this program to add a menu bar which a single menu item which displays a MessageBox displaying your own details when the item is clicked.

TASK 9: Create a Simple Calculator GUI

Create a new project called `Calculator`. Create a form that allows the User to enter in two numbers and then select either an Add, Subtract, Multiple or Divide button to produce the output on screen. An example layout has been show below but don't be afraid to make your own



EXERCISE 3

TASK 10: Extend your simple calculator in Task 9 to add more operations to your calculator. Add buttons to calculate the Modulus, the Power of the two numbers (first number raised to power of second number and also add error handling to ensure if the user types in a value that is not a number, an error dialog is shown on screen.

Hint: Use try catches around converting the values form the text boxes and use the built in Math class for some Maths operation e.g. `Math.Pow()`:

```
try
{

}
catch(Exception e)
{

}
```

```
Math
```

class System.Math
Provides constants and static methods for trigonometric, logarithmic, and other common mathematical functions.

TASK 11: BIG CAT HOUSE GUI

Create a GUI that lists the big cats present in the Cat House in the Zoo. The cats include 2 Tigers, 3 Lions and a Leopard. You will need to create a `BigCat` class that stores details about each cat e.g. Age, Name, Country of Origin, Gender, Species etc.

Have a button that when pressed, shows you all the cats on the Form (Hint: Labels). Also include buttons to view details of each species e.g. Tigers, Lions or Leopards in a `MessageBox` dialog.

If you feel like you want to add more functionality, add a menu bar which shows you stats on the Cat House in a Message Dialog e.g. Number of each Species.