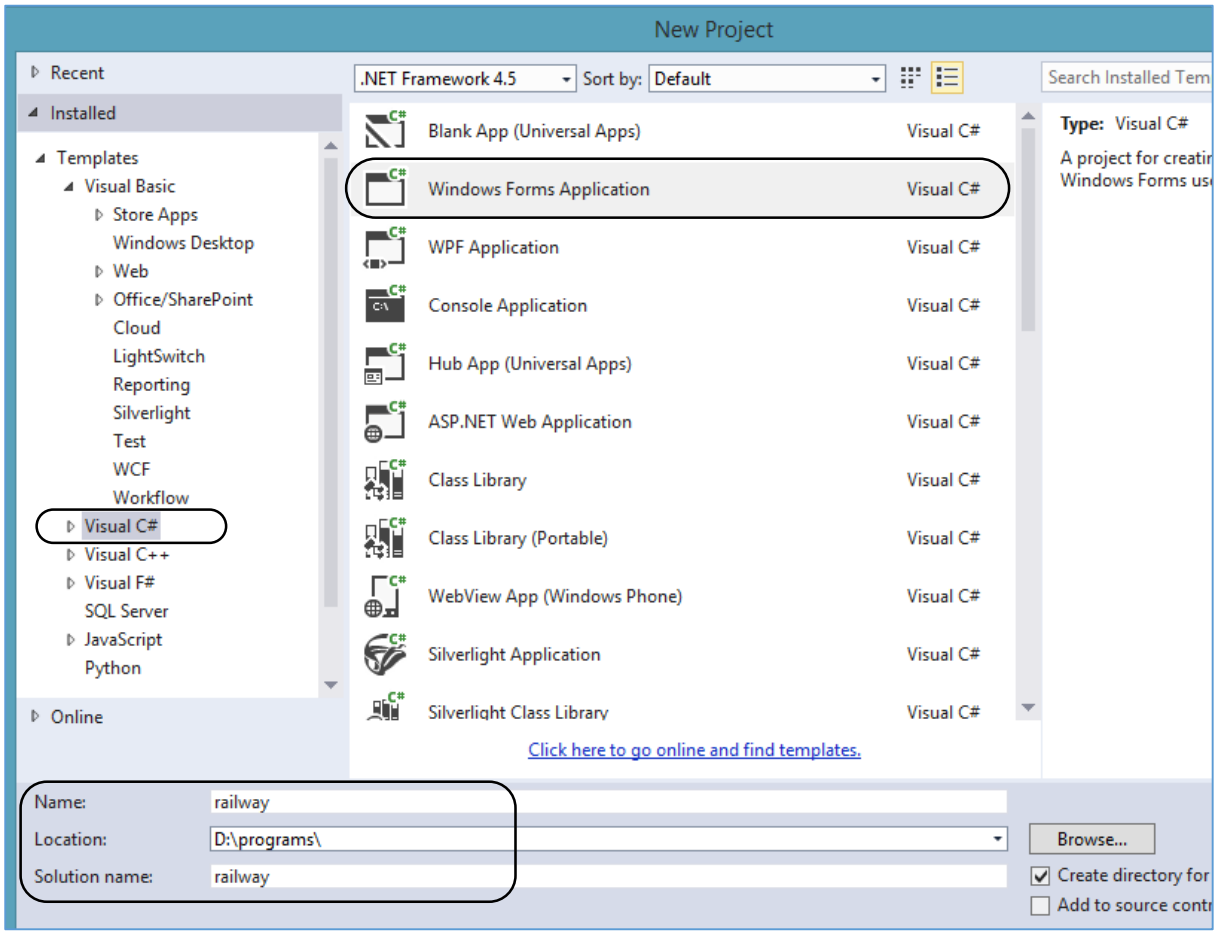
3 Railway Tickets

The next program combines the techniques we have used so far, to produce a ticket program for a narrow gauge railway.

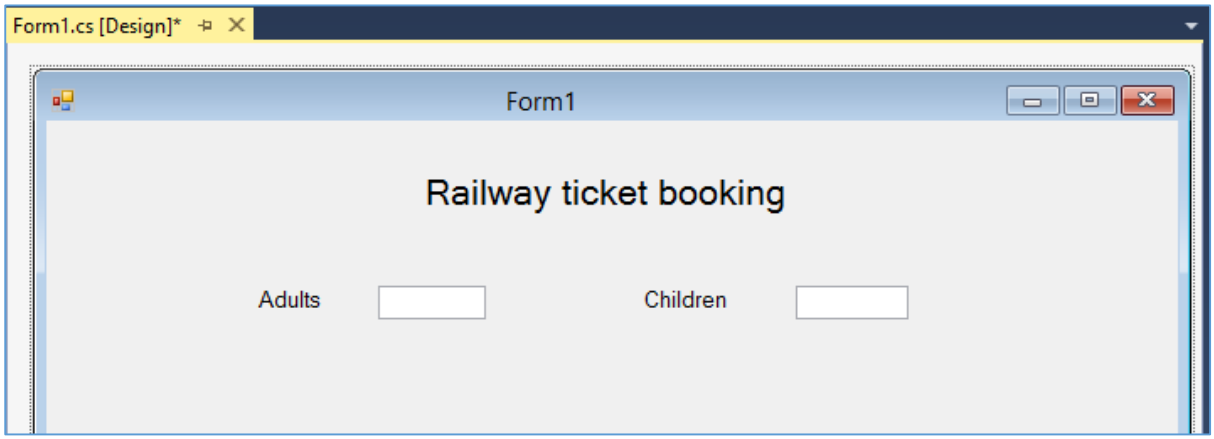
The fare structure for the railway is:

- Single second class adult fare: £6.40
- Return fare is one and a half times the single fare
- All child fares are 60% of the equivalent adult fare
- Each adult or child passenger may pay a £2.00 surcharge to travel First Class for the whole of their single or return journey
- Groups of four or more passengers booking together receive a 20% discount on their total fare

We will begin by starting a new project called '***railway***'

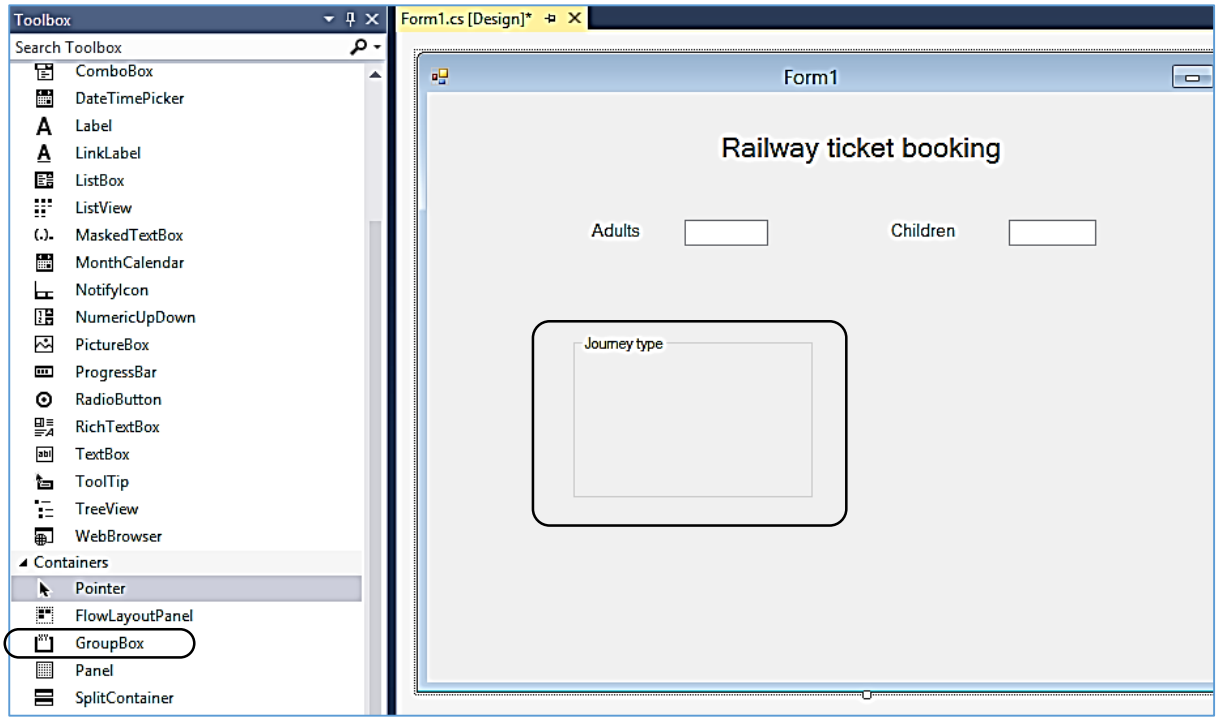


Create two ***text boxes*** for entering the numbers of adult and child passengers. Give the boxes the names ***txtAdult*** and ***txtChild***. Add labels for the screen display:



We will now set up buttons for choosing a *single* or *return* ticket.

Begin by selecting the ***GroupBox*** component in the Containers section of the Toolbox, and drag the mouse to draw a frame. Change the Text property of the GroupBox to '***Journey type***' to create a heading:

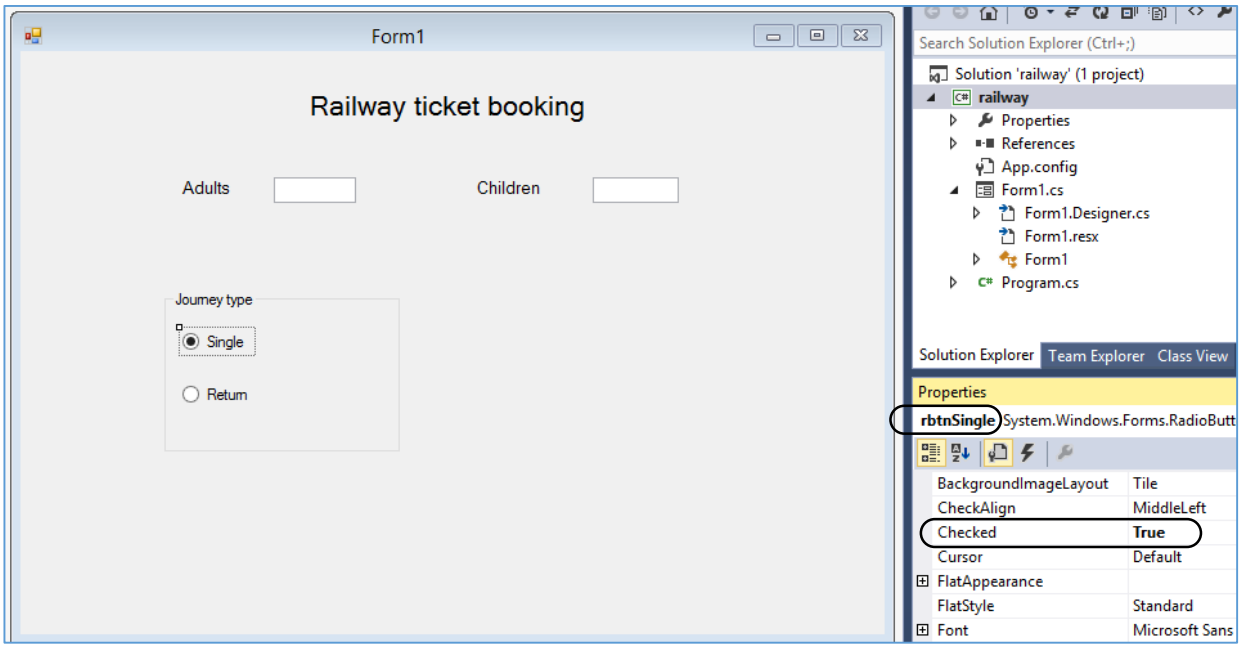


Choose the ***RadioButton*** component from the Toolbox and drag the mouse to place two buttons inside the group box.

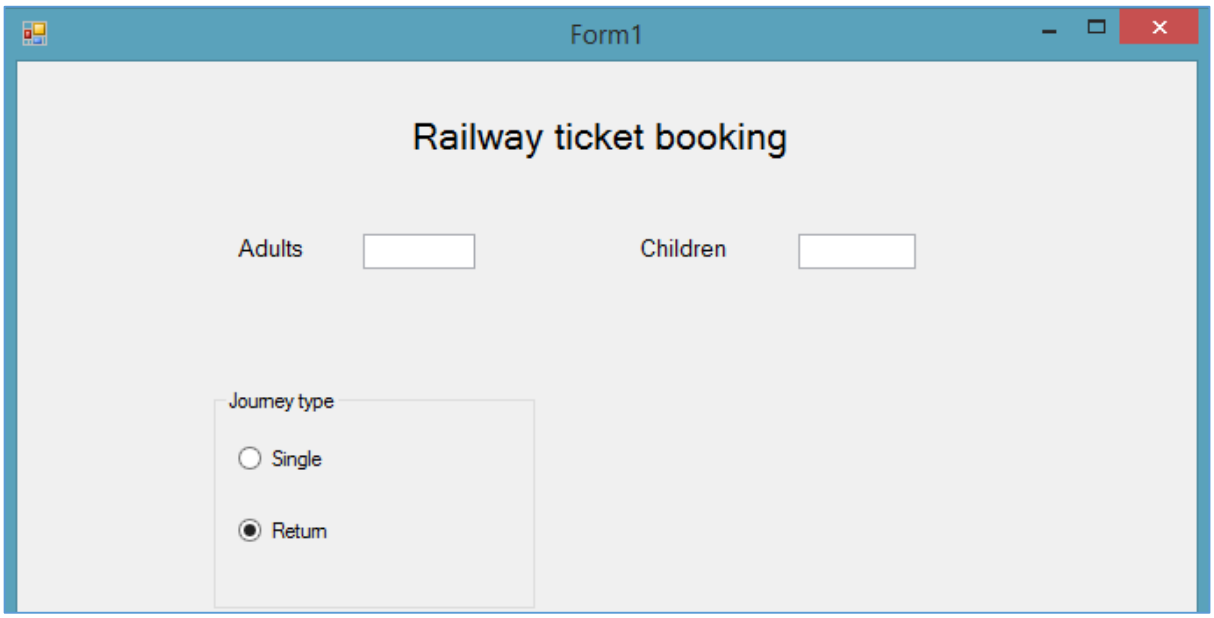
Change the Text properties of the RadioButtons to '***Single***' and '***Return***'.

Give the RadioButtons the names ***rbtnSingle*** and ***rbtnReturn***.

Set the ***Checked*** property of the 'Single' button to ***True***:



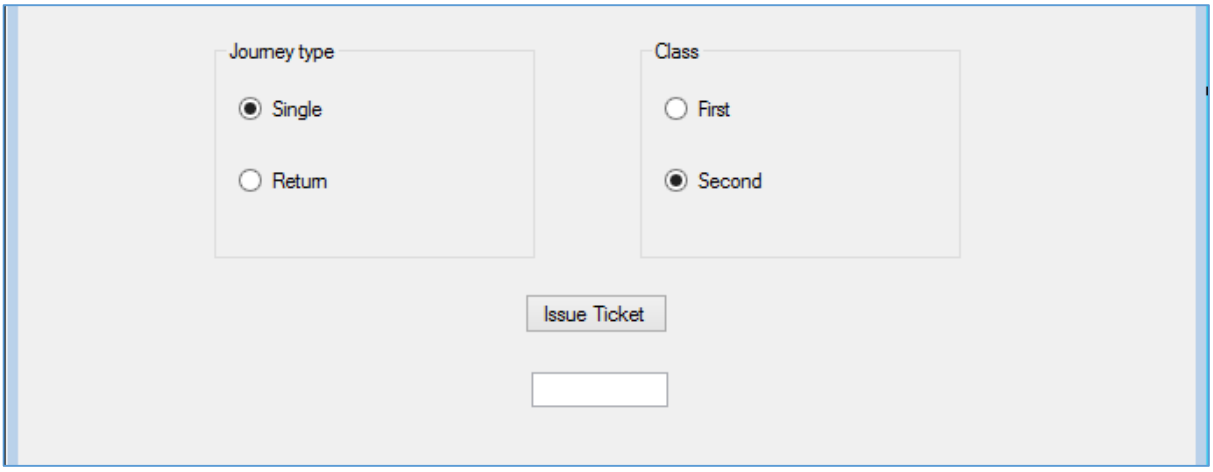
Run the program to test the radio group. If this works correctly, the '***Single***' button should initially be selected. It should only be possible to select one of the options:



Stop the program by clicking the red cross at the top right-hand corner of the form.

Return to the design screen and add another ***GroupBox*** and two ***RadioButtons*** to allow the user to select a '***First class***' or '***Second class***' ticket.

Give the RadioButtons the names ***rbtnFirst*** and ***rbtnSecond***. Set the ***Checked*** property of the '***Second***' RadioButton to ***True***.



Complete the form by adding a ***Button*** for issuing the ticket, and a ***TextBox*** for displaying the ticket price. Give these components the names ***btnTicket*** and ***txtTicketprice***.

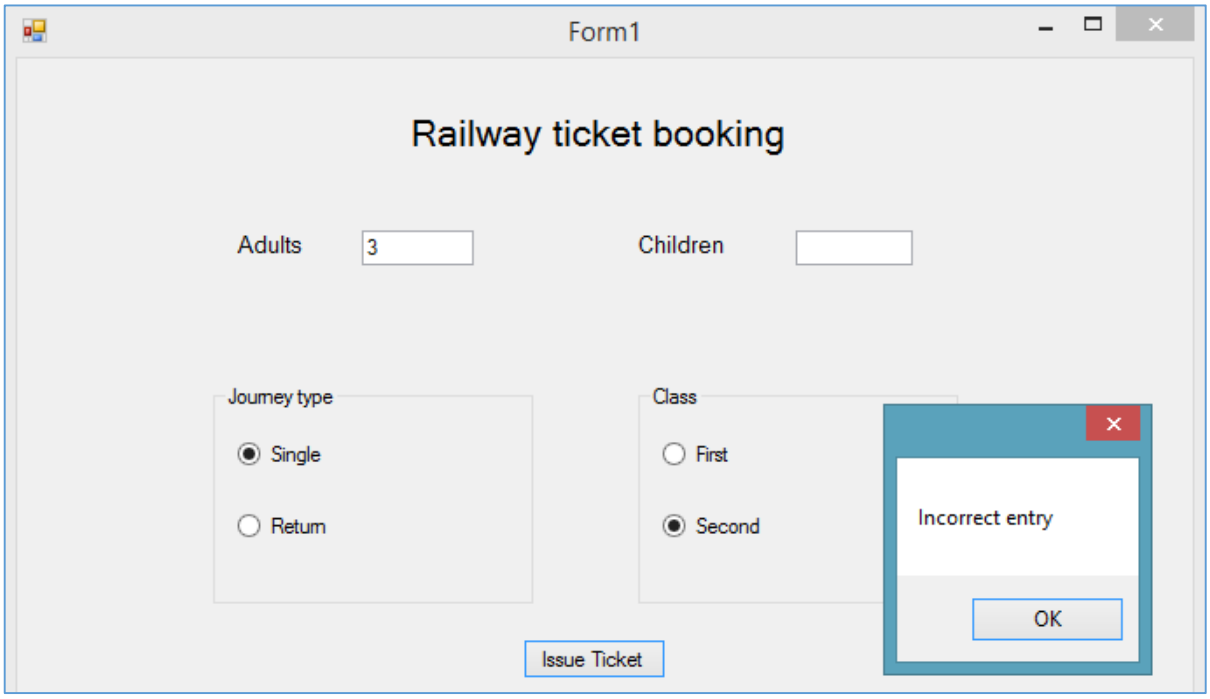
We can now work on the calculations. Right-click on the form, then select '***View Code***'. Add two integer variables '***adult***' and '***child***' at the start of Form1.

Right-click on the form, then select '***View Designer***'. Double-click the '***Issue Ticket***' button to create a button_click method. Add lines of code to obtain the numbers of adults and children from the TextBox inputs.

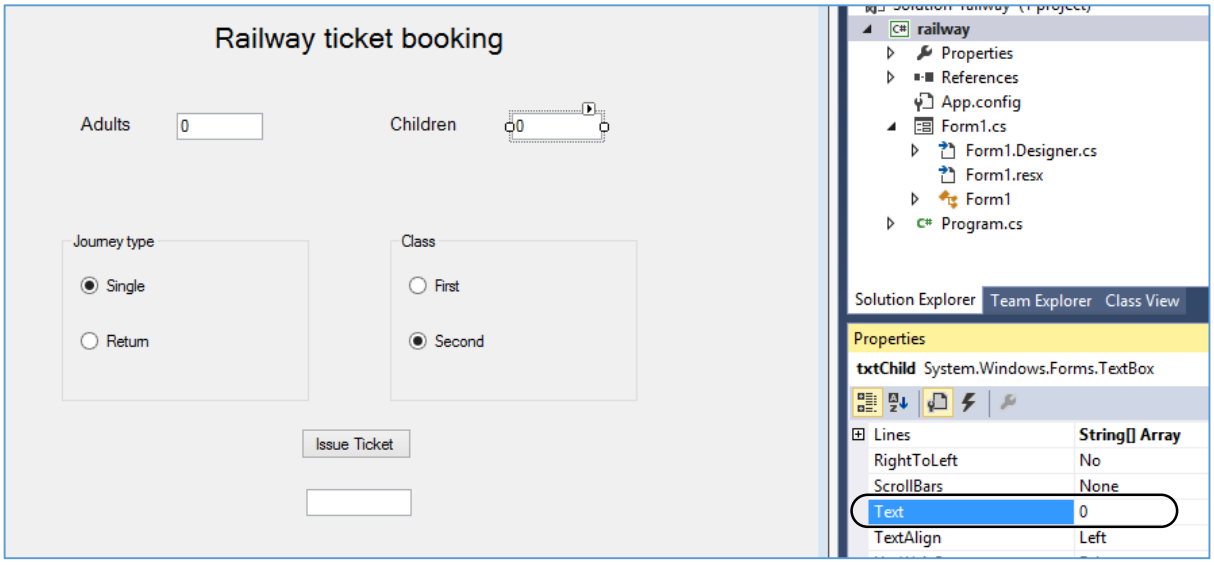
We do not want the program to crash if the user makes an entry error, so we will use a TRY..CATCH structure to provide a warning message:

```
public partial class Form1 : Form
{
    int adult, child;

    public Form1()
    {
        InitializeComponent();
    }

    private void btnTicket_ Click(object sender, EventArgs e)
    {
        try
        {
            adult = Convert.ToInt16(txtAdult.Text);
            child = Convert.ToInt16(txtChild.Text);
        }
        catch
        {
            MessageBox.Show("Incorrect entry");
        }
    }
```

Run the program to test the error trapping. This should give an '***incorrect entry***' message if letters are typed instead of a number, but unfortunately the warning also appears if we leave a box blank – for example, if no children are travelling:



Return to the design screen. Select each of the input TextBoxes in turn, and set the Text property to '***0***'.



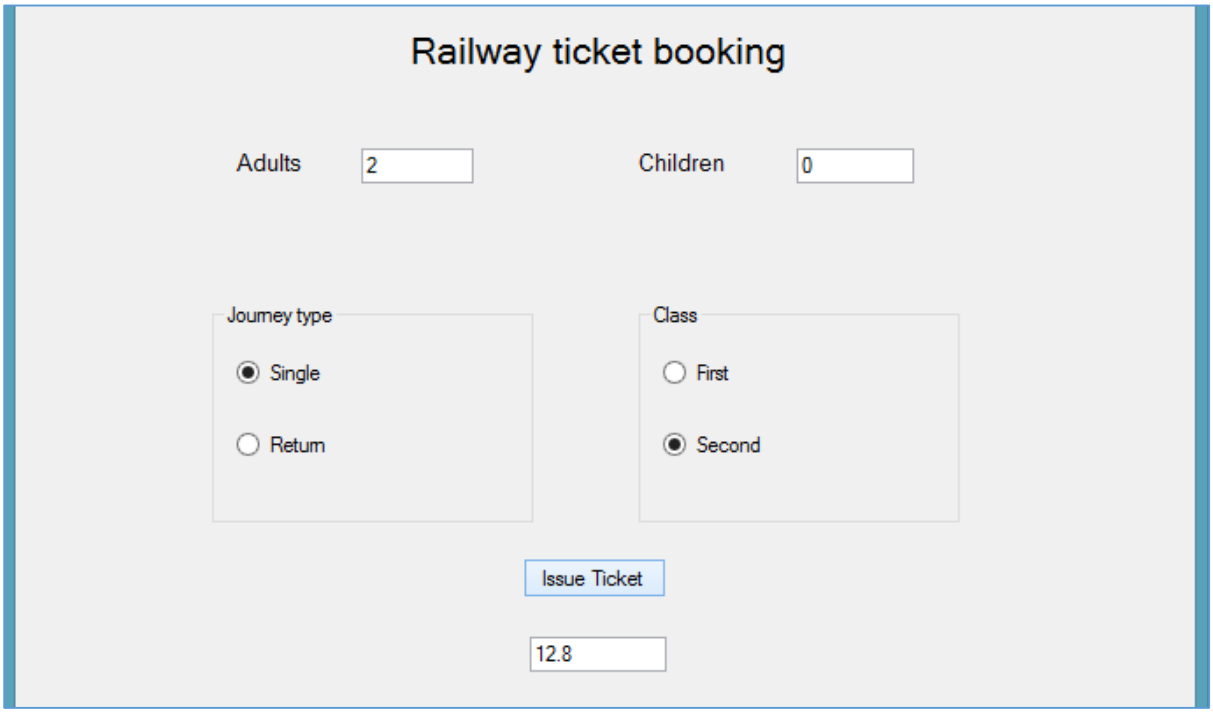
Run the program again and check that no error message appears when booking for one adult travelling on their own. Close the program window and go to the program code.

We can now start work on calculating the ticket cost. Set up a variable '***ticketcost***' which will store this as a decimal number to represent pounds and pence.

The fare structure is quite complex, so it is best to split the calculation into a series of stages, and test the program after each stage. We can begin by working out the cost of the adult and child tickets, assuming a single fare of £6.40, with children paying 60% of the adult cost. The total can then be displayed in the TextBox:

```
public partial class Form1 : Form
{
    int adult, child;
    double ticketcost;

    public Form1()
    {
        InitializeComponent();
    }

    private void btnTicket_Click(object sender, EventArgs e)
    {
        try
        {
            adult = Convert.ToInt16(txtAdult.Text);
            child = Convert.ToInt16(txtChild.Text);
            ticketcost = adult * 6.40;
            ticketcost = ticketcost + (child * 6.40 * 0.60);

            txtTicketprice.Text = Convert.ToString(ticketcost);
        }
```

Run the program and check the calculations for different numbers of adults and children (remembering that all tickets are currently ***second class single***).



The calculations should be correct, but we have a problem with the display format. In some cases, the pence are not shown correctly with two decimal places.

Go back to the program listing, and alter the code to ensure that the output is always displayed to two decimal places:

```
try
{
    adult = Convert.ToInt16(txtAdult.Text);
    child = Convert.ToInt16(txtChild.Text);
    ticketcost = adult * 6.40;
    ticketcost = ticketcost + (child * 6.40 * 0.60);

    string s=ticketcost.ToString("f2");
    txtTicketprice.Text = s;

}
```

We can now consider the other ticket options. Begin by setting up two ***Boolean*** (true/false) variables called '***single***' and '***first***':

```
public partial class Form1 : Form
{
    int adult, child;
    double ticketcost;
    bool single;
    bool first;

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

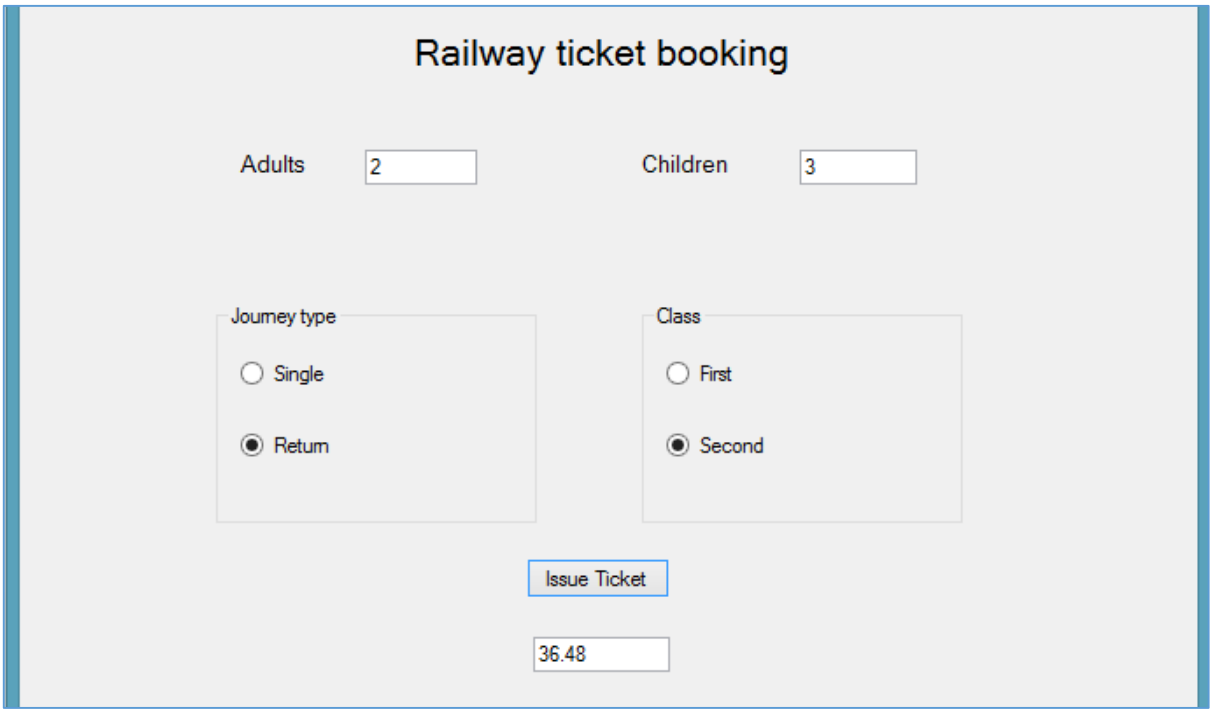
We will use the '***single***' variable to record the type of journey:

- ***single = true*** means 'single ticket wanted'
- ***single = false*** means 'return ticket wanted'

An easy way to set the value of '***single***' is just to see whether the '***Single***' RadioButton has been selected. If not, then the user must require a '***Return***' ticket and we add the extra cost as one and a half times the single fare. Add lines to the program as shown:

```
try
{
    adult = Convert.ToInt16(txtAdult.Text);
    child = Convert.ToInt16(txtChild.Text);
    ticketcost = adult * 6.40;
    ticketcost = ticketcost + (child * 6.40 * 0.60);

    single = rbtnSingle.Checked;
    if (single == false)
    {
        ticketcost = ticketcost * 1.5;
    }

    string s=ticketcost.ToString("f2");
    txtTicketprice.Text = s;
}
```

Run the program and test that second class Single and Return fares are now calculated correctly:



We will use a similar method to add the First Class charge of £2.00 per person:

```
try
{
    adult = Convert.ToInt16(txtAdult.Text);
    child = Convert.ToInt16(txtChild.Text);
    ticketcost = adult * 6.40;
    ticketcost = ticketcost + (child * 6.40 * 0.60);
    single = rbtnSingle.Checked;

    if (single == false)
    {
        ticketcost = ticketcost * 1.5;
    }

    first = rbtnFirst.Checked;

    if (first == true)
    {
        ticketcost = ticketcost + (adult + child) * 2.00;
    }

    string s=ticketcost.ToString("f2");
    txtTicketprice.Text = s;
}
```

Run the program and check that First Class fares are now calculated correctly.

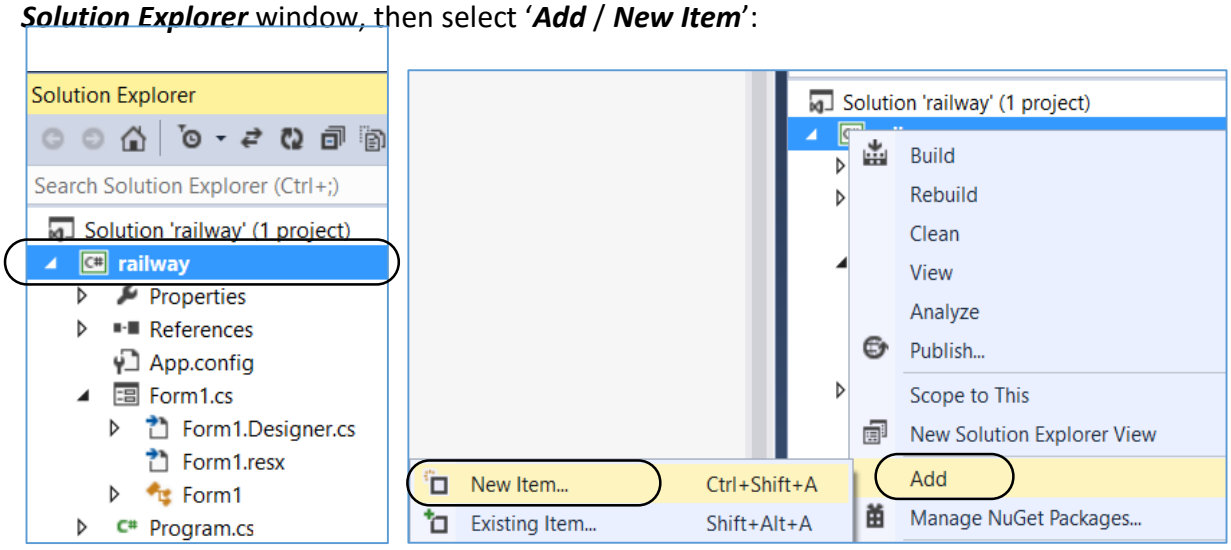
The final step to complete the ticket calculation is to apply the 20% discount offered to groups of four or more passengers:

```
first = rbtnFirst.Checked;
if (first == true)
{
    ticketcost = ticketcost + (adult + child) * 2.00;
}

if ((adult + child) >= 4)
{
    ticketcost = ticketcost - (ticketcost * 0.2);
}

string s=ticketcost.ToString("f2");
txtTicketprice.Text = s;
```

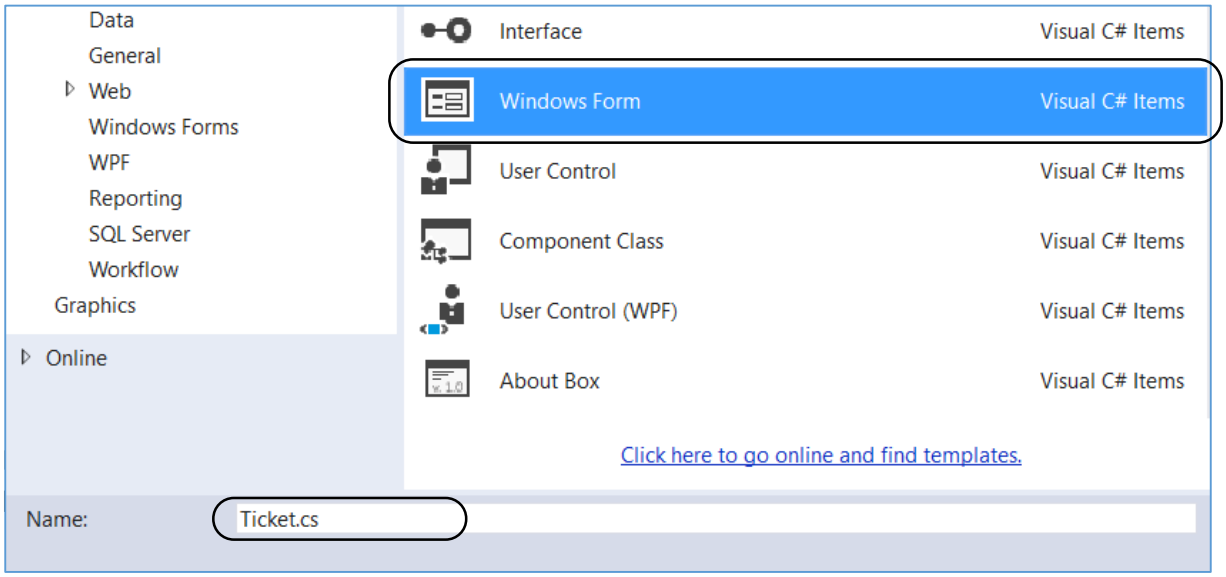
Run the program and check that group discounts are being calculated correctly.

We will now move on to produce the ticket for the customer …

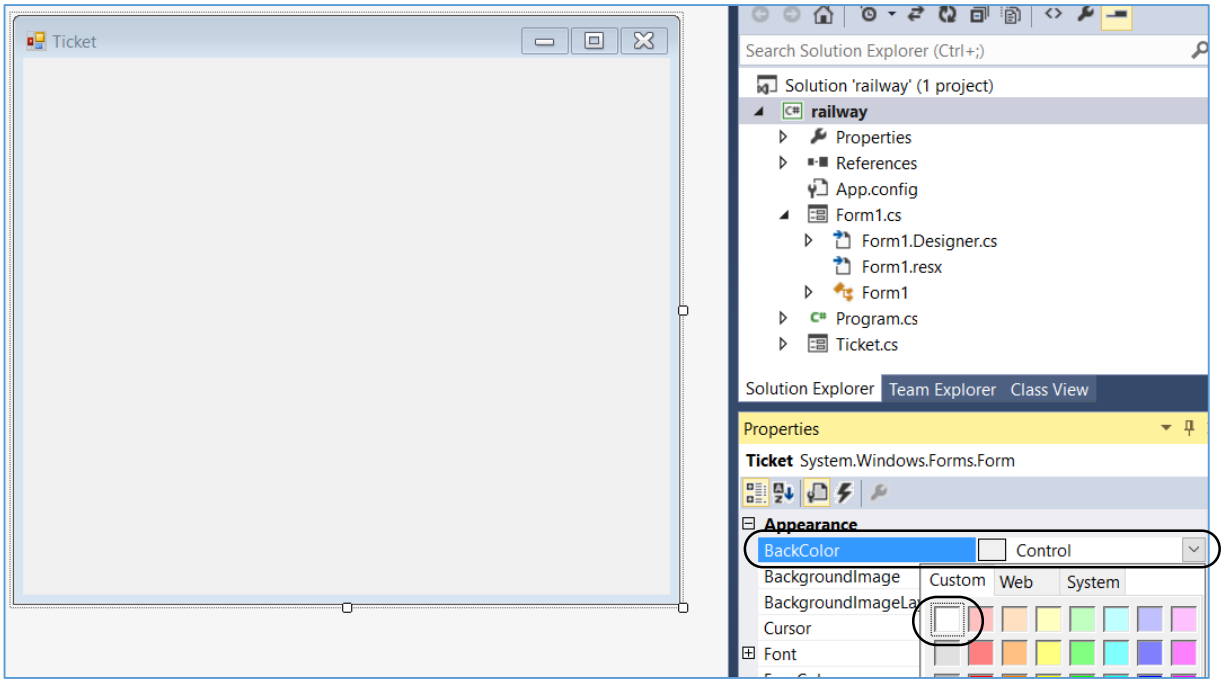
Begin by creating another form. To do this, click-right on the C# '***railway***' icon in the ***Solution Explorer*** window, then select '***Add*** / ***New Item***':



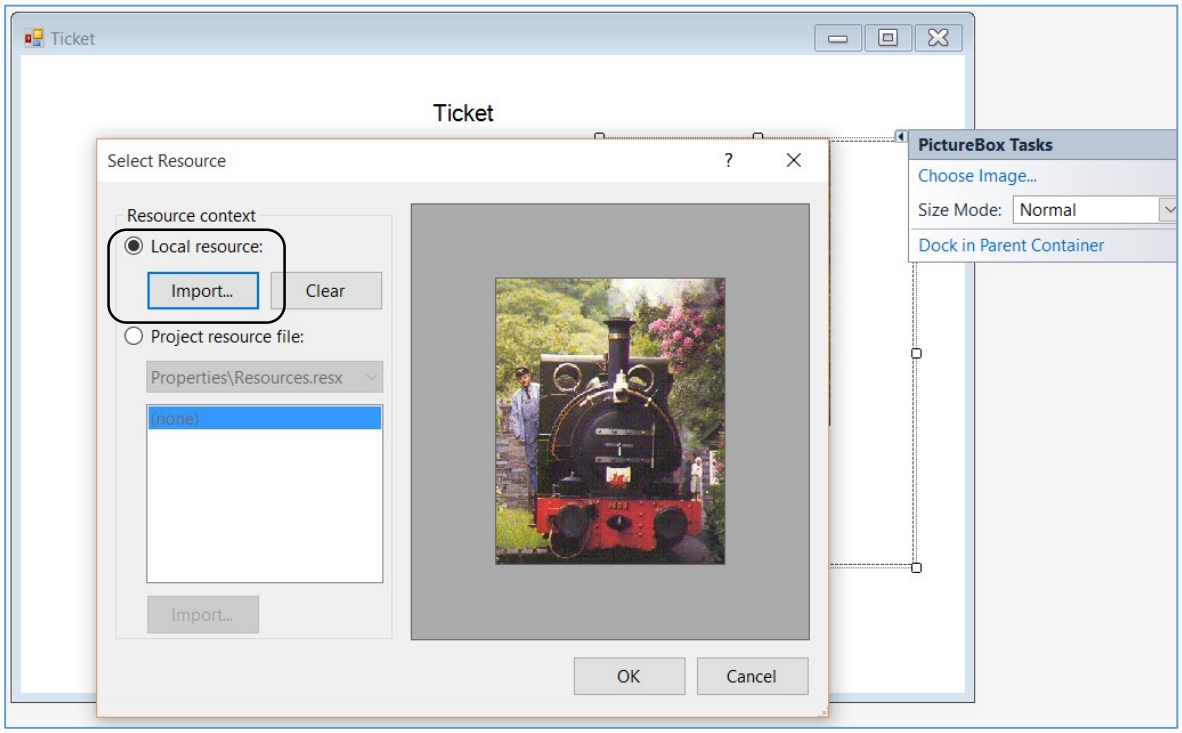
Select '***Windows Form***', and give this the name '***Ticket.cs***':



It will be best to use a white background. Click on the form, then go to ***BackColor*** in the Properties window. Click to open the colour selection palettes, and choose white from the ***Custom*** palette:

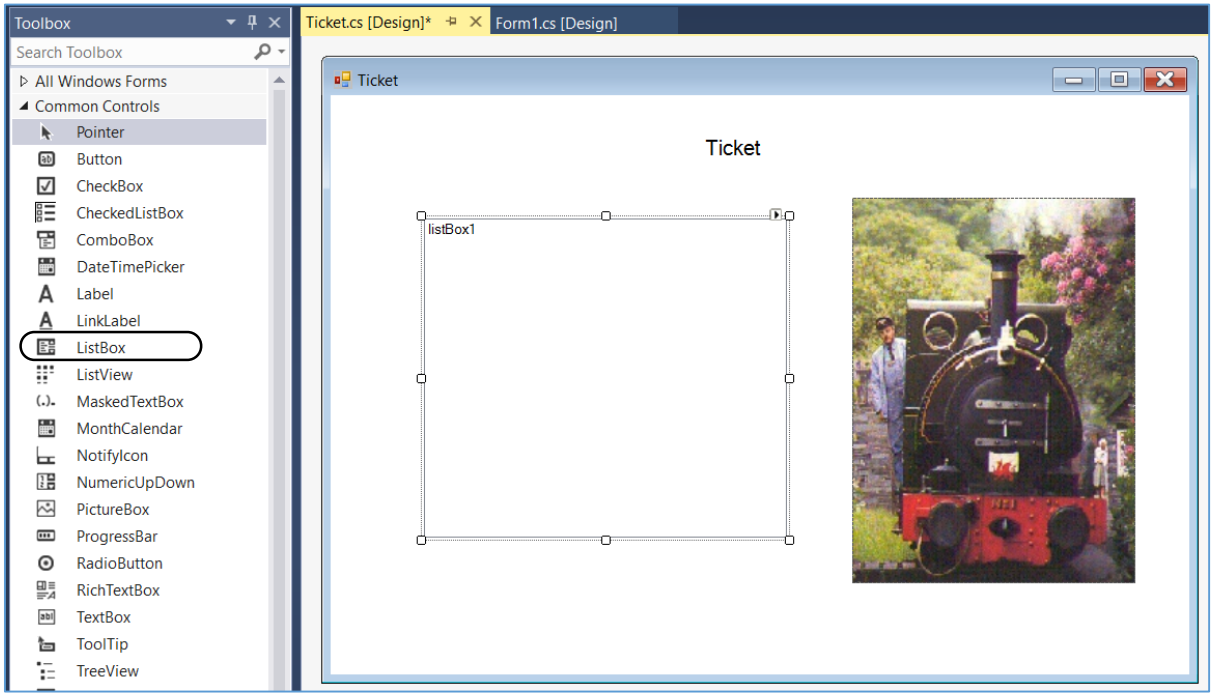


Add a Label '***Ticket***'. Place a ***Picture Box*** in the right hand half of the ticket area, and insert a suitable picture:



The final component we need to produce the ticket is a ***ListBox***, in which we can display the ticket details.

Select the ListBox component in the Toolbox and drag the mouse on the form to produce a box outline. Go to the Properties window, and set the Font property to display a larger point size.

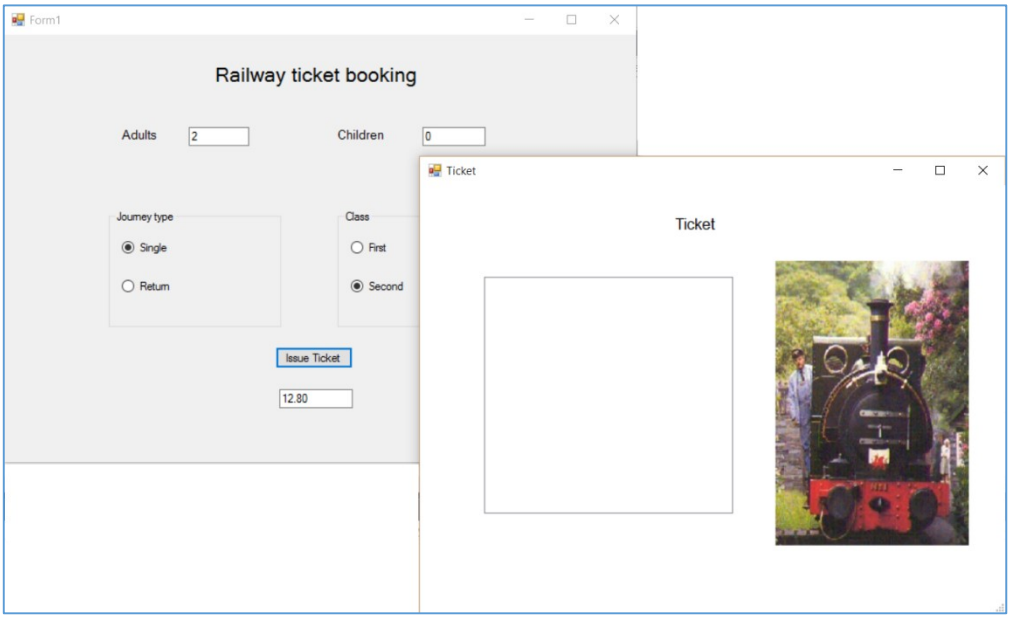


The Ticket form needs to open when the 'Issue Ticket' button is clicked on Form1. To do this, go to the end of the button_click method on Form1, and add the lines of code:

```
        if ((adult + child) >= 4)
        {
            ticketcost = ticketcost - (ticketcost * 0.2);
        }
        string s=ticketcost.ToString("f2");
        txtTicketprice.Text = s;

        Ticket frmticket = new Ticket();
        frmticket.ShowDialog();
    }
    catch
    {
        MessageBox.Show("Incorrect entry");
    }
```

We are using '***Ticket***' to refer to the CLASS we have designed for displaying tickets, whilst '***frmticket***' is one particular OBJECT belonging to this class which we will create when the program is running.

Run the program, enter ticket details, and then click the '***Issue Ticket***' button. If all goes well, the ticket form should open:



In order to produce the ticket, we need to carry over information from ***Form1*** about the number of passengers, type of ticket wanted, and the calculated ticket cost. The best way to do this is to set up a method on the '***Ticket***' form which brings in this data as ***parameters***. Set up the method '***displayTicket***' as shown:

```
public partial class Ticket : Form
{
    public Ticket()
    {
        InitializeComponent();
    }

    public void displayTicket(int adults, int children, bool single,
                                                    bool first, double ticketcost)
    {

    }
}
```

We can now return to ***Form1*** and add a line of code which will call this method, using the values we wish to transfer as parameters:

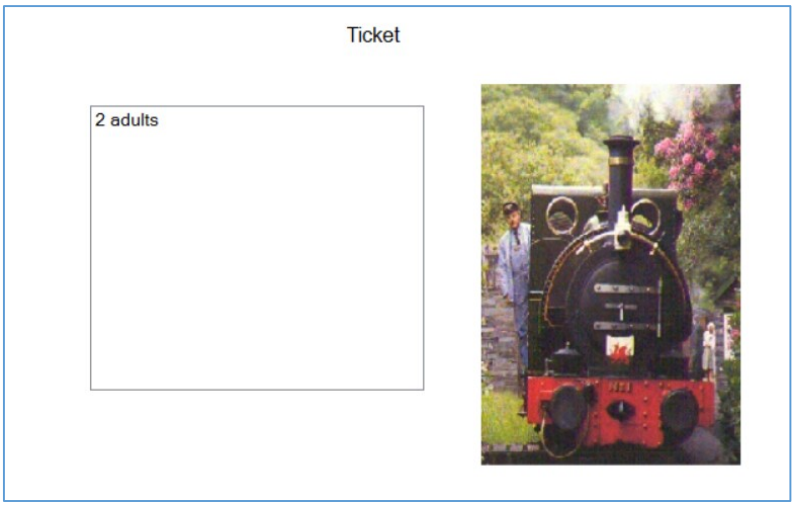
```
            string s=ticketcost.ToString("f2");
            txtTicketprice.Text = s;
            Ticket frmticket = new Ticket();

            frmticket.displayTicket(adult, child, single, first, ticketcost);

            frmticket.ShowDialog();
        }
```

Go back now to the '***Ticket***' program code. The data we need should now be available, using the parameter names we have specified in the header of the '***displayTicket***' method.

We can begin by displaying the number of adults. ListBoxes, like TextBoxes, can only display string data, so it is necessary to convert the number into string format.

```
public void displayTicket(int adults, int children, bool single,
                                          bool first, double ticketcost)
{
    string s = Convert.ToString(adults);
    listBox1.Items.Add(s + " adults");
}
```

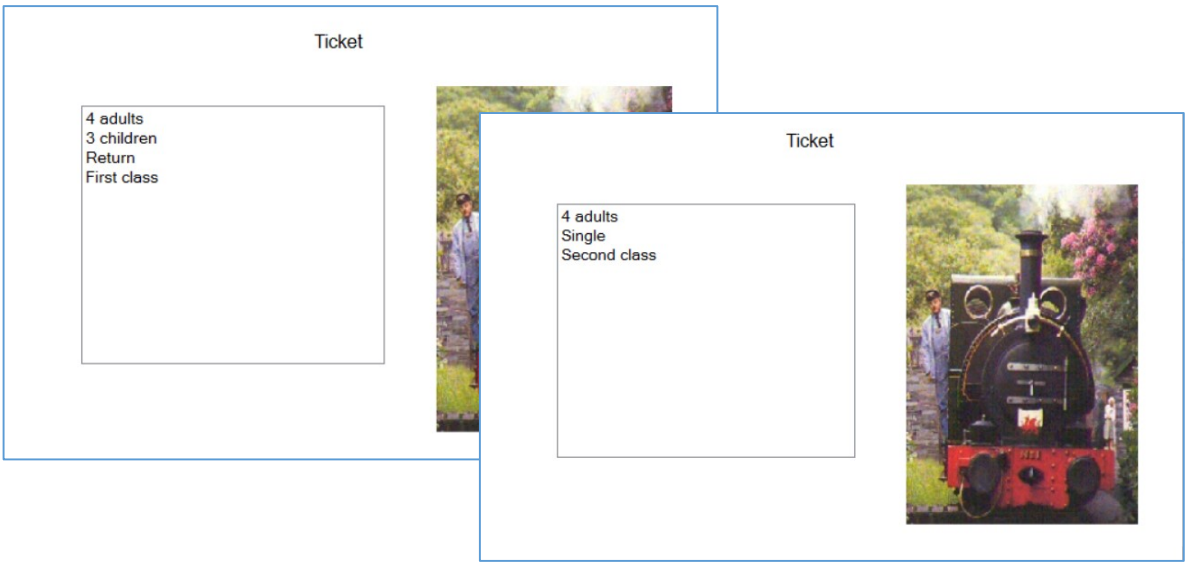
Run the program. Issue a ticket, and check that the correct number of adults is displayed on the ticket:



Return to the '***displayTicket***' method and add lines of code to display the other ticket details. Notice that we have used an ***IF*** condition to only display the number of children when there are actually children included in the ticket:

```
public void displayTicket(int adults, int children, bool single,
                                          bool first, double ticketcost)
{
    string s = Convert.ToString(adults);
    listBox1.Items.Add(s + " adults");

    s = Convert.ToString(children);
    if (children>0)
        listBox1.Items.Add(s + " children");
    if (single==true)
        listBox1.Items.Add("Single");
    else
        listBox1.Items.Add("Return");
    if (first == true)
        listBox1.Items.Add("First class");
    else
        listBox1.Items.Add("Second class");
}
```

Test the program for the different possibilities of: adults plus children or adults only, single or return, first or second class:



Add code to display discount for groups, and to show the overall ticket price:

```
            if (first == true)
                listBox1.Items.Add("First class");
            else
                listBox1.Items.Add("Second class");

            if ((adults+children)>=4)
                listBox1.Items.Add("20% group discount");
            listBox1.Items.Add("");
            s = ticketcost.ToString("f2");
            listBox1.Items.Add("£"+s);
        }
    }
```

Test the finished program. One slight improvement you might try making yourself, is to display the singular word '***adult***' or '***child***' when only one person is travelling, rather than the plural: