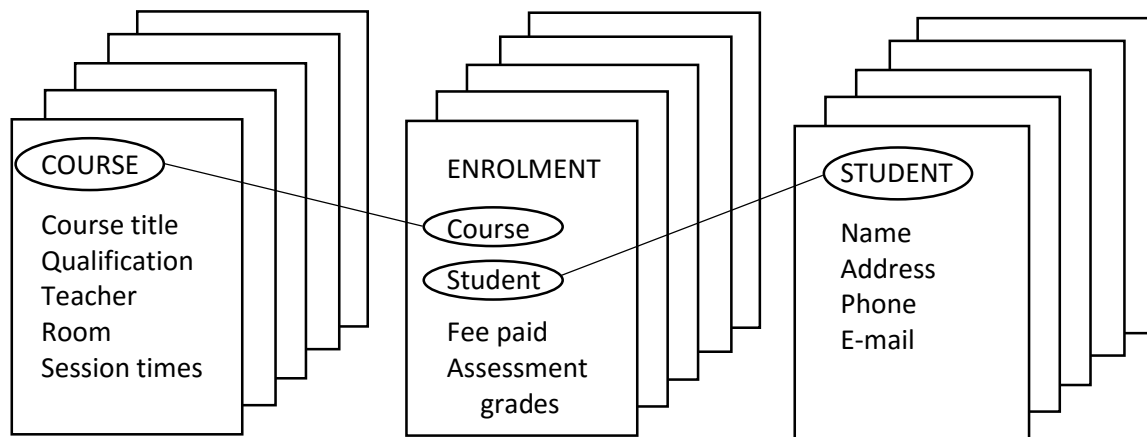


8 College Courses

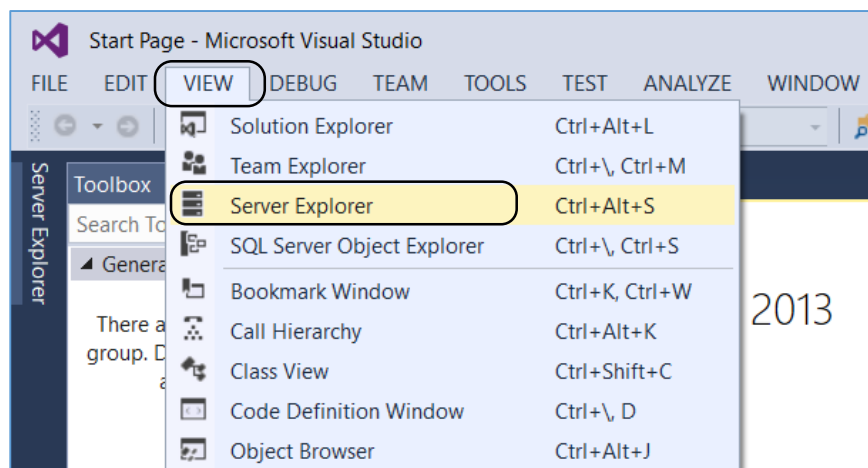
In this program we will introduce **classes** of **objects** as a means of representing entities in a data model. We will design a record keeping system for a college to enter and store details of **courses**, **students**, and **enrolments** of students on particular courses. Three classes of objects will be needed to operate the system:



We will begin by setting up a database containing three tables to hold data for courses, students and enrolments.

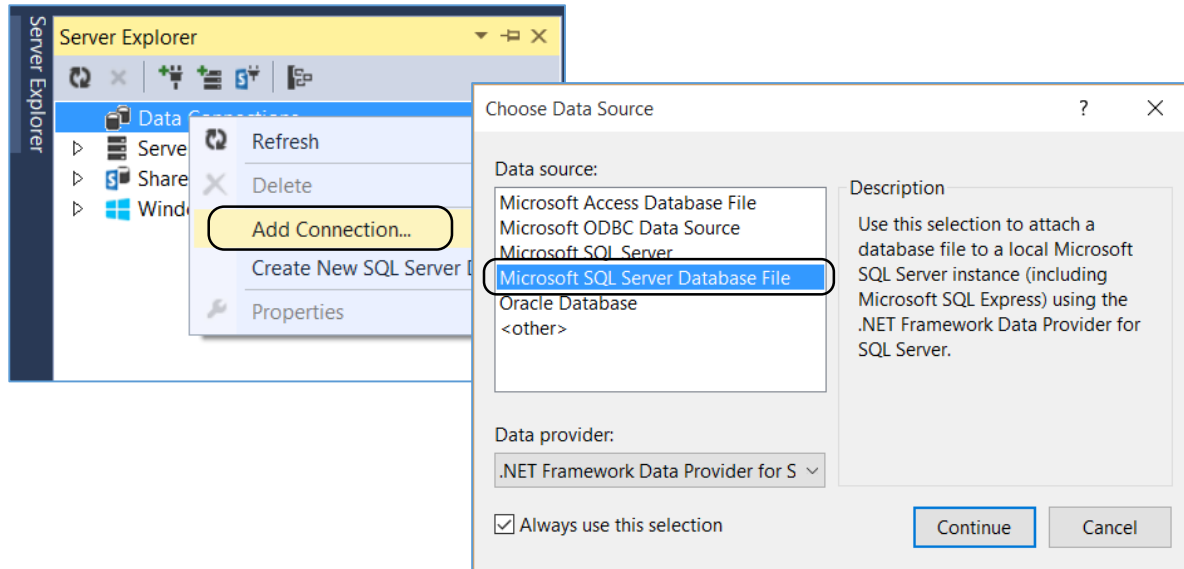
Open *Visual Studio*, but do not start a new project yet.

On the menu line, select '**View / Server Explorer**'

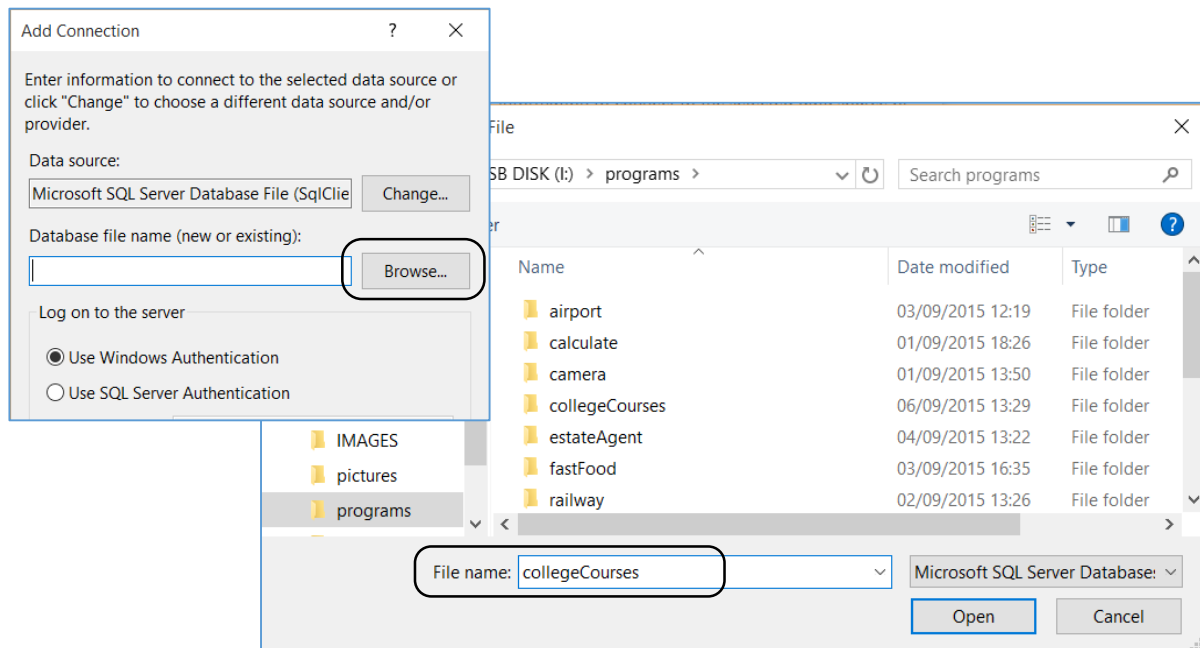


Right-click on '**Data Connections**', then select '**Add Connection**'. If you are asked to choose a Data Source, select:

Microsoft SQL Server Database File

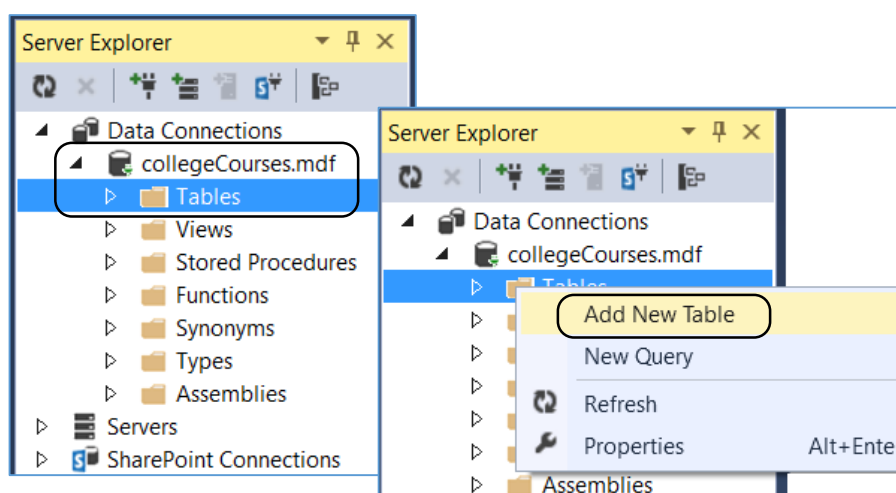


Use the **'Browse'** option to navigate to the location where your C# programs are stored. Give the file name **'collegeCourses'** for the database which will be created.

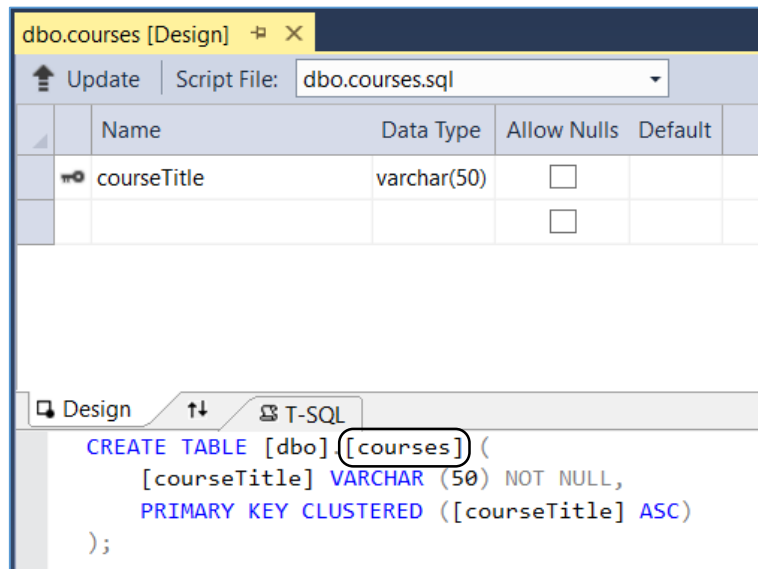


Click **'OK'**, then answer **'Yes'** that you wish to create the database file.

Click-right on **'Tables'** and select **'Add New Table'**:

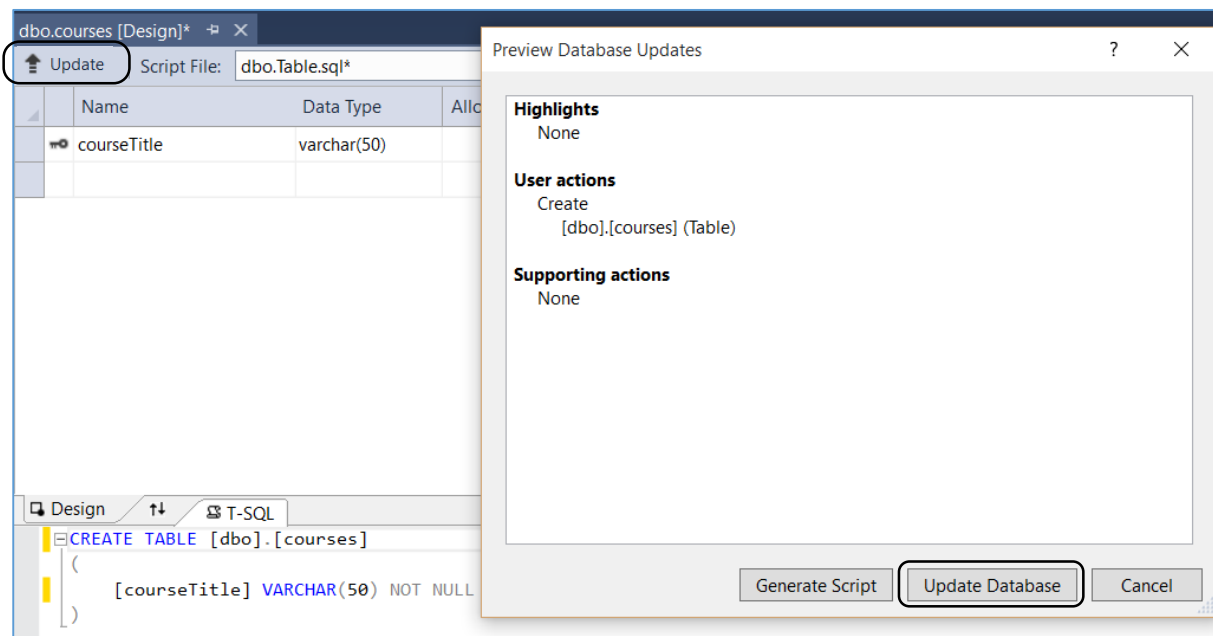


We will begin by setting up a table to store records of the college **courses**. To keep the program simple we will just store the **course title**, although in a real system the database would store further information about each course such as: the tutor, room, days and times of the classes.



Go to the **CREATE TABLE** line and change the name of the table to '**courses**'.

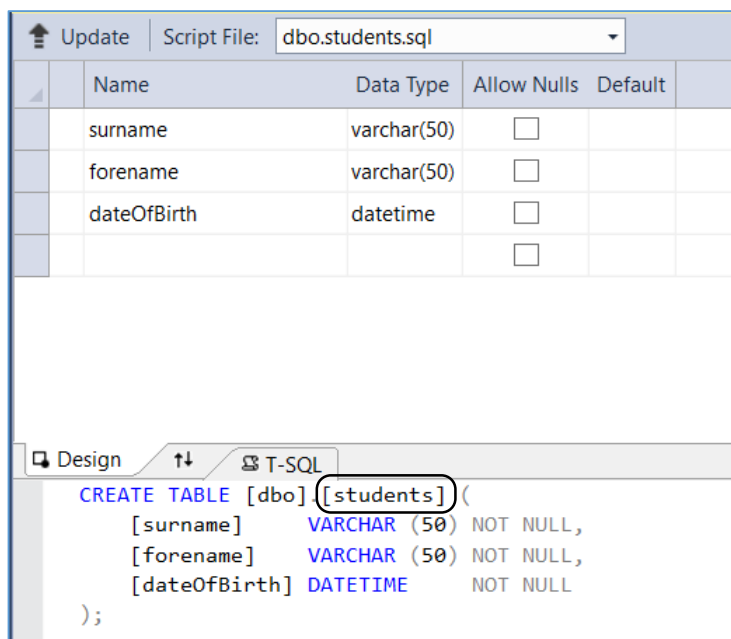
Click the '**Update**' button above the list of fields. When the **Database Update** window appears, click the '**Update Database**' button.



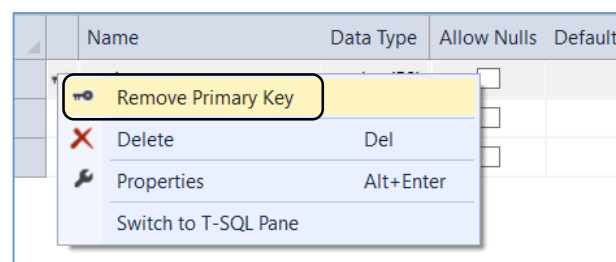
Close the **courses** table design page and re-open the **Server Explorer** window. Right-click on collegeCourses.mdf and select '**Refresh**'.

Click the small arrow to the left of the **Tables** icon. The **courses** table which you created should now be shown.

Repeat the steps above to create a **students** table. For simplicity, we will only add three fields: **surname**, **forename** and **dateOfBirth**. In a real system, this table would include the student's address and other contact details.



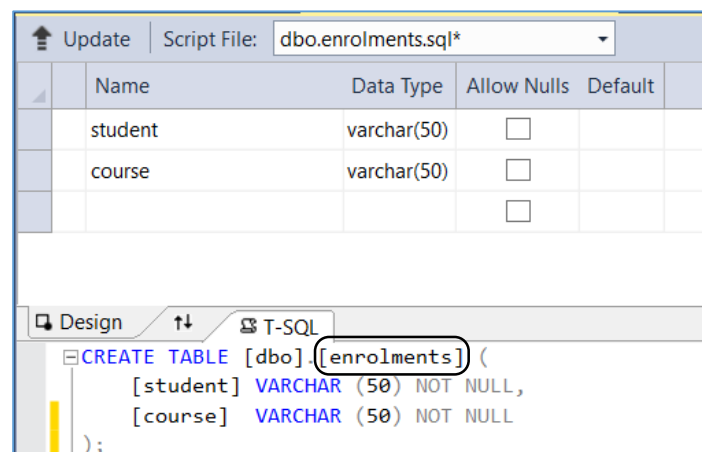
Delete the **Primary Key** by right-clicking on the key icon alongside the first field of the table, then select '**Remove Primary Key**' for the drop down list.



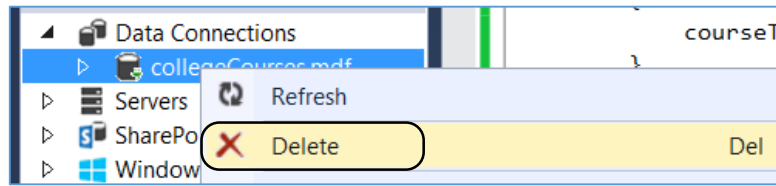
Go to the **CREATE TABLE** line and change the name of the table to '**students**'.

Click the '**Update**' button above the list of fields. When the **Database Update** window appears, click the '**Update Database**' button.

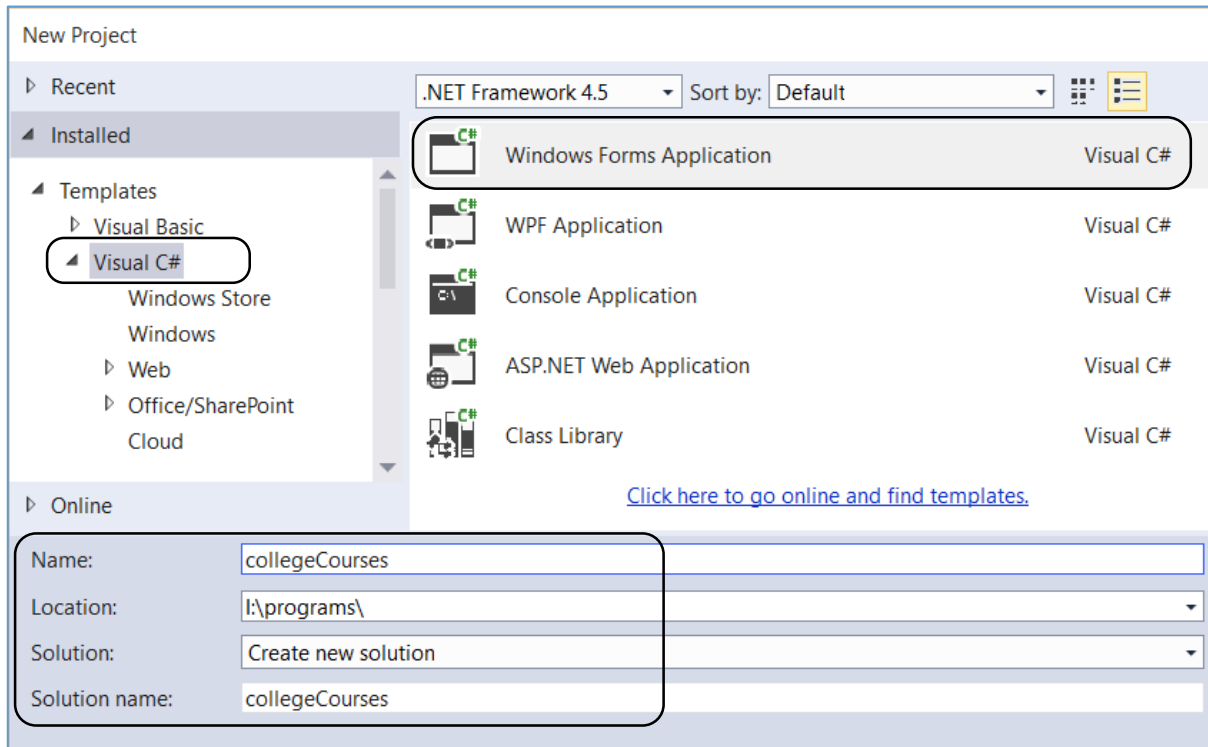
Complete the database structure by adding a third table for **enrolments**. Again delete the **Primary Key** from the table.



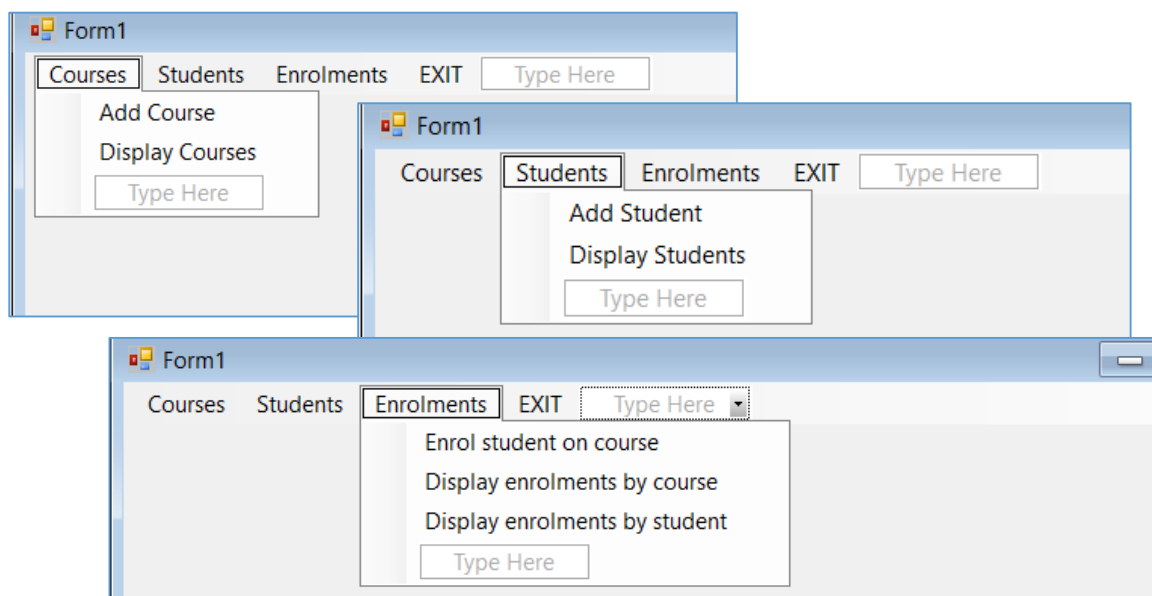
Close the connection to the **collegeCourses** database by right-clicking the , ready to begin work on the program.



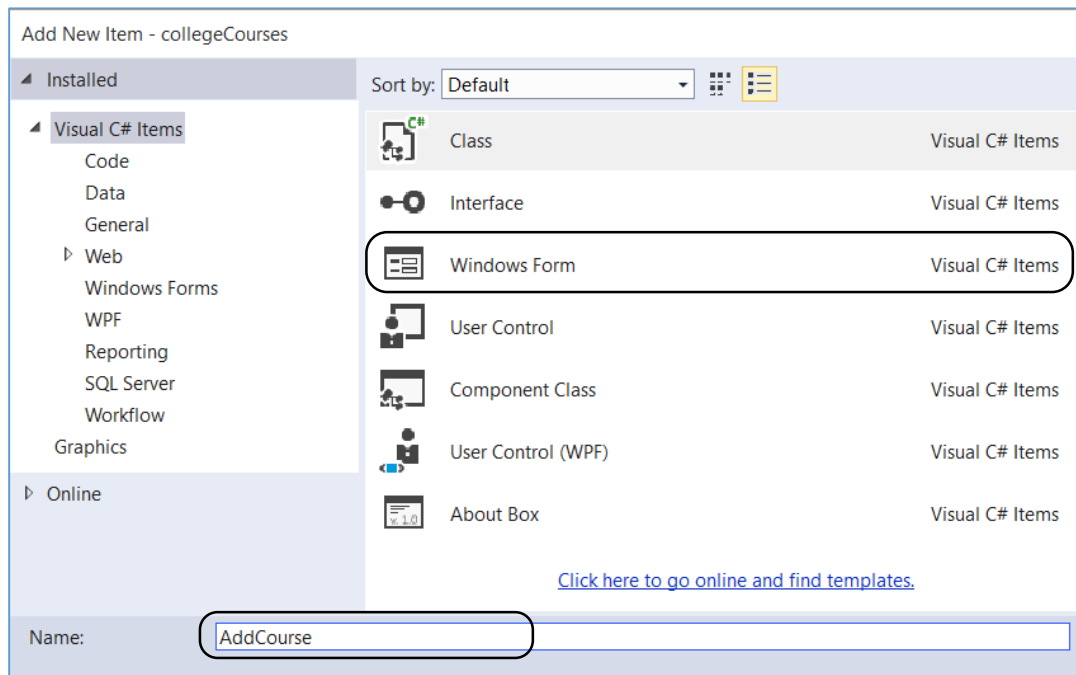
Start a **new C# project**. Select '**Windows Forms Application**', and call the project '**collegeCourses**'.



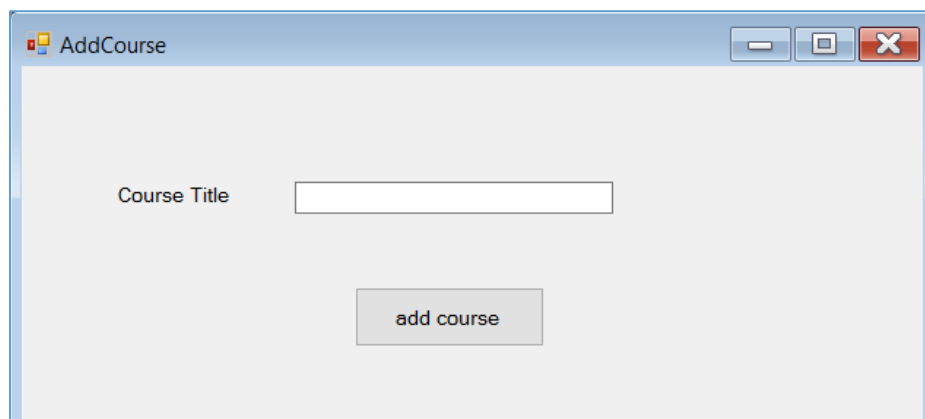
Set up a **menuStrip** on **Form1**, in a similar way to the Estate Agent Database program in Chapter 7. Create options for **adding** and **displaying courses, students** and **enrolments**:



We will create a form for adding courses. Go to the **Solution Explorer** window, right click on the **collegeCourses** program icon, then select '**Add / New Item**'. Choose **Windows Form** and give the name '**AddCourse**'.



Add a **label**, a **textBox** with the name **txtCourseTitle**, and a **button** with the name **btnAddCourse**.



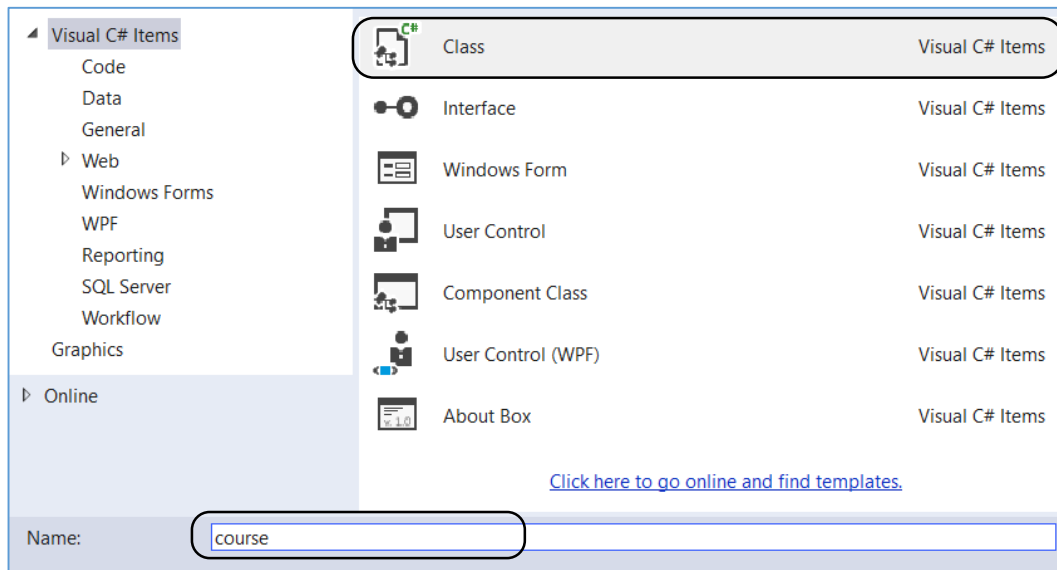
Return to **Form1** and double click the '**Add course**' menu option. Add code to open the AddCourse form.

```
private void addCourseToolStripMenuItem_Click(object sender, EventArgs e)
{
    AddCourse frmAddCourse = new AddCourse();
    frmAddCourse.ShowDialog();
}
```

Run the program and check that the **AddCourse** form opens correctly.

The object oriented way of handling data in this program will be rather different to the procedural approach of previous programs. Each time that a course is entered, a separate **course** object will be created and the **course title** will be assigned to it as a **property** of the object.

We must begin by defining the **course** class. Go to the **Solution Explorer** window, right-click on the **collegeCourses** program, and select '**Add / New Item**'. From the item list, choose '**Class**' and give this the name '**course**'.



The file that you have created has no Form associated with it. Only its code window can be opened.

We will add some useful functions:

- **courseCount** is an integer which will record the number of courses entered.
- **course[]** defines an array of course objects. We are allowing for six courses to be entered, but the array size could be set to any maximum required. No memory space is actually used in the computer until we choose to create course objects when the program is running.
- **courseTitle** is a **property** of the object. In a real system, other properties would be included, such as the name of the course tutor, and the location and times of classes.

```
class course
{
    public static int courseCount;
    public static course[] courseObject = new course[6];
    private string courseTitle;

    public void setTitle(string t)
    {
        courseTitle = t;
    }
    public string getTitle()
    {
        return courseTitle;
    }
}
```

Two methods have been created:

- **setTitle()** allows the course title to be transferred into the **course** object.
- **getTitle()** allows the course title to be transferred back from the **course** object to the outside program.

These two methods are the only means by which the value of **courseTitle** can be accessed or changed. This approach makes object oriented programming more secure and less liable to errors. In a procedural program, data is often accessible from different parts of the program, leading to unexpected logical errors which can be difficult to identify and correct.

When a new **course** object is created, it should be added to the database table so that it is available on future occasions that the program is run. We will add a **saveCourse()** method to do this:

```
using System.Linq;
using System.Text;

using System.Data;
using System.Data.SqlClient;

namespace collegeCourses
{
    class course
    {
        string databaseLocation = "C:\\C#\\collegeCourses.mdf;";

        public static int courseCount;
        public static course[] courseObject = new course[6];
        private string courseTitle;

        public void setTitle(string t)
        {
            courseTitle = t;
        }
        public string getTitle()
        {
            return courseTitle;
        }

        public void saveCourse()
        {
            SqlConnection con = new SqlConnection(@"Data Source=.\\SQLEXPRESS;
            AttachDbFilename=" + databaseLocation + "Integrated Security=True;
            Connect Timeout=30; User Instance=True");
            con.Open();
            SqlCommand cmCourse = new SqlCommand();
            cmCourse.Connection = con;
            cmCourse.CommandType = CommandType.Text;
            cmCourse.CommandText = "INSERT INTO courses(courseTitle)
            VALUES ('" + courseTitle + "')";
            cmCourse.ExecuteNonQuery();
            con.Close();
        }
    }
}
```


The lines beginning:

```
SqlConnection con = new SqlConnection(...  
cmCourse.CommandText = "INSERT INTO...
```

should each be entered as a single line of code with no line breaks.

It is necessary to add '**using Data**' and '**using SqlClient**' directives, and to give the location of the database.

Return to the **AddCourse** form. Double click the '**add course**' button and add code to the event procedure.

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

    private void btnAddCourse_Click(object sender, EventArgs e)
    {
        string title = txtCourseTitle.Text;
        course.courseObject[course.courseCount] = new course();
        course.courseObject[course.courseCount].setTitle(title);
        course.courseObject[course.courseCount].saveCourse();
        course.courseCount++;
        this.Close();
    }
}
```

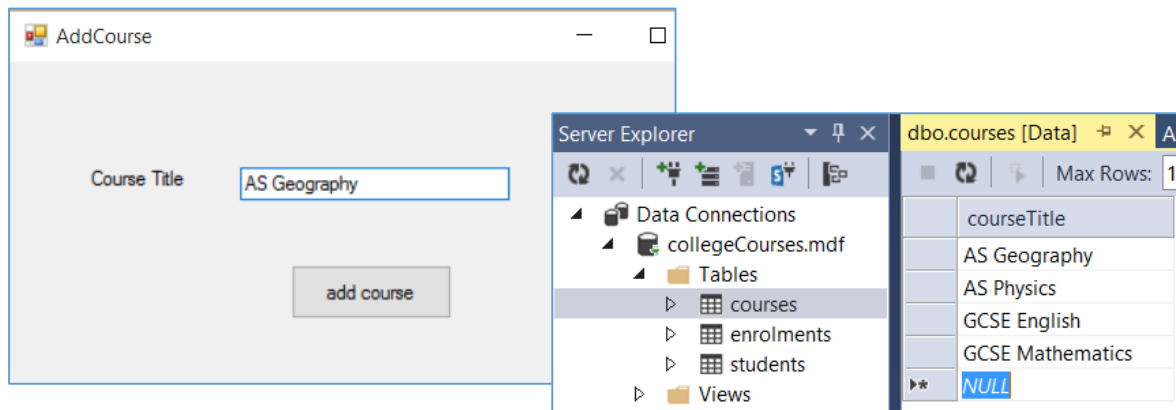
This method carries out a series of actions:

- The course title entered in the textBox is collected.
- A new course object is created. This is assigned a location in the courseObject[] array according to the current value of **courseCount**.
- The course title is transferred to the new object using the setTitle() method.
- The new course object has its **property** details saved into the database table – in this case, just the course title needs to be saved.
- Finally, **courseCount** is increased by one.

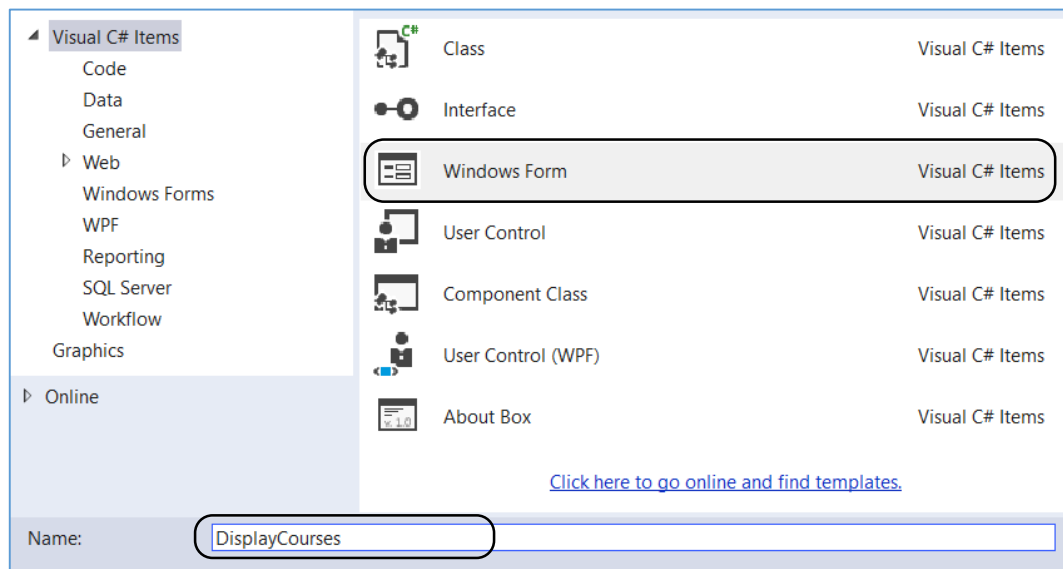
Return to **Form1**. We will initialise the number of courses to zero when the program first starts by adding a line of code to the **Form1()** method:

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
        course.courseCount = 0;
    }
}
```

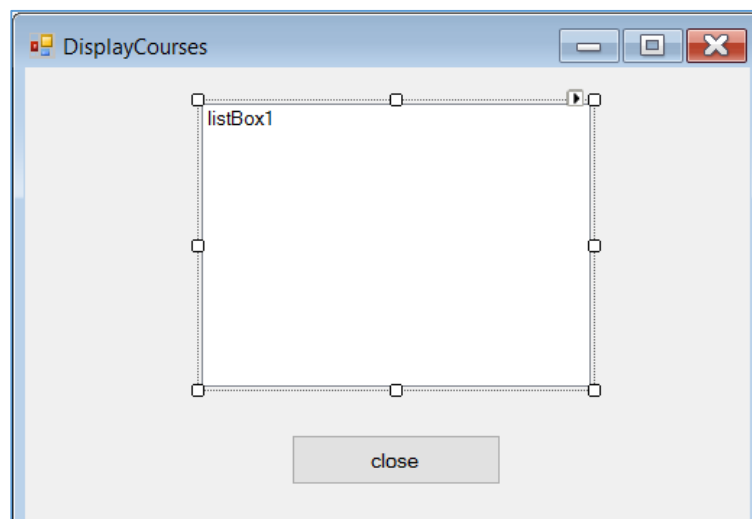
Run the program and enter test data for courses. Connect to the database and check that the course titles appear correctly in the database table, then delete the database connection.



We can now produce a form to display the list of courses. Right click the **collegeCourses** program icon in the **Solution Explorer** window and select 'Add / New item'. Create a **Windows Form**, and name this '**DisplayCourses**':



Add a **ListBox** and **Button** to the form.



Rename the button as **btnClose**. Double click the button and add a line of code to close the form.

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

    private void btnClose_Click(object sender, EventArgs e)
    {
        this.Close();
    }
}
```

Add a **courseList()** method, and call this from the **DisplayCourses()** method.

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

    private void courseList()
    {
        int courseCount = course.courseCount;
        string title;

        listBox1.Items.Clear();
        for (int i = 0; i < courseCount; i++)
        {
            title = course.courseObject[i].getTitle();
            listBox1.Items.Add(title);
        }
    }
}
```

Notice how the **courseList()** method collects **courseCount** from the **course** class. It then uses this variable to operate a loop which loads the title of each course and displays it in the list box. Each course title is obtained from the array of **course** objects using the **getTitle()** method.

Return to **Form1**. Double click the '**Display Courses**' menu option and add code to open the **displayCourses** form:

```
private void displayCoursesToolStripMenuItem_Click(object sender, EventArgs e)
{
    DisplayCourses frmDisplayCourses = new DisplayCourses();
    frmDisplayCourses.ShowDialog();
}
```

We have one more task to complete before courses can be displayed by the program. It is necessary to add a method to **Form1** which will load the course information from the database table and create **course** objects when the program first runs.

Begin by adding the '**using SqlClient**' directive and the database location:

```
using System.Windows.Forms;

using System.Data.SqlClient;

namespace collegeCourses
{
    public partial class Form1 : Form
    {
        string databaseLocation = "C:\\C#\\collegeCourses.mdf";
    }
}
```

Create a **loadCourses()** method to load the records from the database table. Call this from the **Form1()** method:

```
public Form1()
{
    InitializeComponent();
    course.courseCount = 0;

    loadCourses();
}

private void loadCourses()
{
    DataSet dsCourses = new DataSet();

    SqlConnection con = new SqlConnection(@"Data Source=.\SQLEXPRESS;
    AttachDbFilename=" + databaseLocation + "Integrated Security=True;
    Connect Timeout=30; User Instance=True");
    try
    {
        con.Open();
        SqlCommand cmCourses = new SqlCommand();
        cmCourses.Connection = con;
        cmCourses.CommandType = CommandType.Text;
        cmCourses.CommandText = "SELECT * FROM courses";
        SqlDataAdapter daCourses = new SqlDataAdapter(cmCourses);
        daCourses.Fill(dsCourses);
        con.Close();
    }
    catch
    {
        MessageBox.Show("File error");
    }
}
```

Remember that the line beginning:

SqlConnection con = new SqlConnection(...

should be entered as a single line of code with no line breaks.

Add code to find the number of courses loaded, then use a loop to create the correct number of course objects and set the **courseTitle** for each object.

```
SqlDataAdapter daCourses = new SqlDataAdapter(cmCourses);
daCourses.Fill(dsCourses);
con.Close();

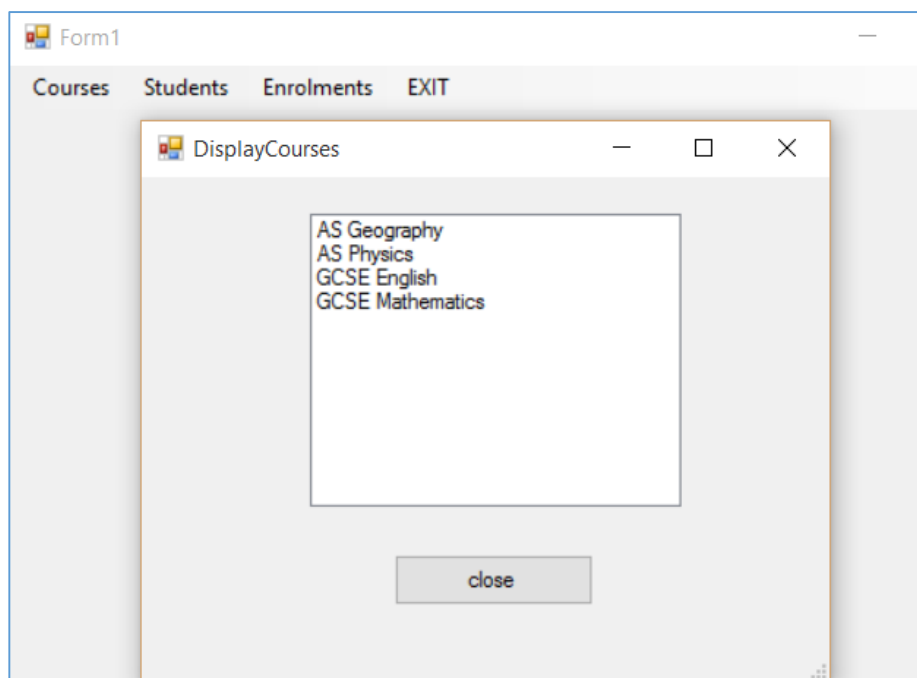
int countRecords = dsCourses.Tables[0].Rows.Count;

for (int i = 0; i < countRecords; i++)
{
    DataRow drCourse = dsCourses.Tables[0].Rows[i];
    string courseTitle = Convert.ToString(drCourse[0]);

    course.courseObject[course.courseCount] = new course();
    course.courseObject[course.courseCount].setTitle(courseTitle);

    course.courseCount++;
}
}
catch
{
```

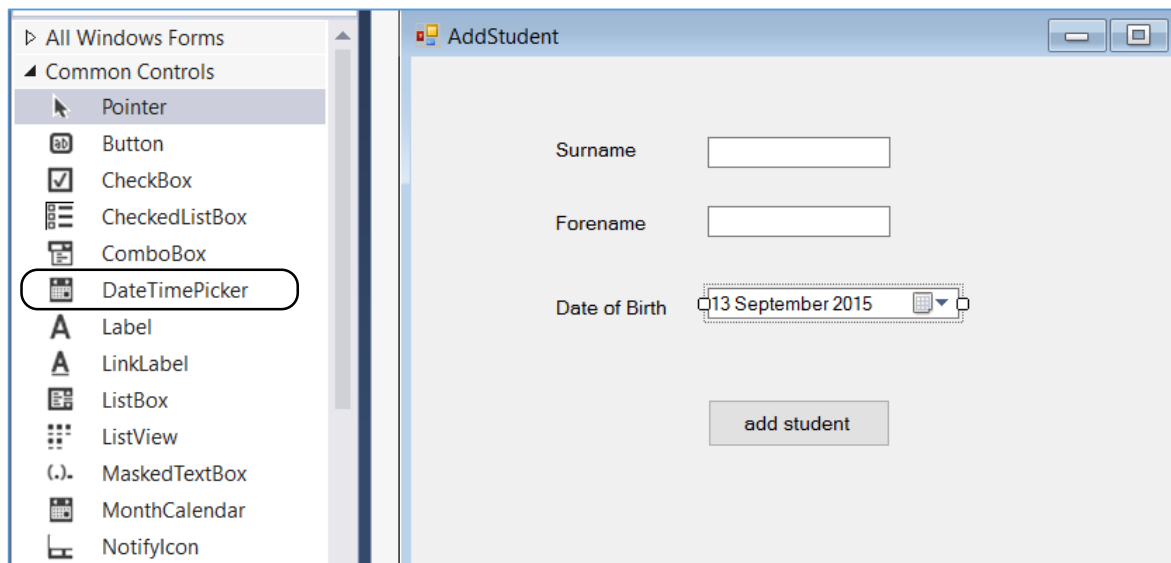
Run the program and select the '**Display courses**' menu option. The course titles which you entered earlier should be displayed.



This completes the **courses** section of the program.

We can now work on the **students** section which will be structured in a very similar way. Begin by creating a **Windows Form** with the name '**AddStudent**'.

- Insert two **textBoxes** and two **labels** for '**Surname**' and '**Forename**'. Name the editBoxes as **txtSurname** and **txtForename**.
- Add a **DateTimePicker** component which will be used to enter the student's date of birth. This should also have a **label**.
- Complete the form with a **button**. Give this the name **btnAddStudent**.




Go to **Form1** and add code to the '**Add student**' menu option to open the **AddStudent** form.

```
private void addStudentToolStripMenuItem_Click(object sender, EventArgs e)
{
    AddStudent frmAddStudent = new AddStudent();
    frmAddStudent.ShowDialog();
}
```

Run the program and check that the '**Add student**' option opens the form correctly. **Date of birth** can be selected by clicking first on the **calendar drop down arrow**, then on the **month heading**, and finally on the **year heading** to reach the scrolling year display, as shown below.

Surname

Forename

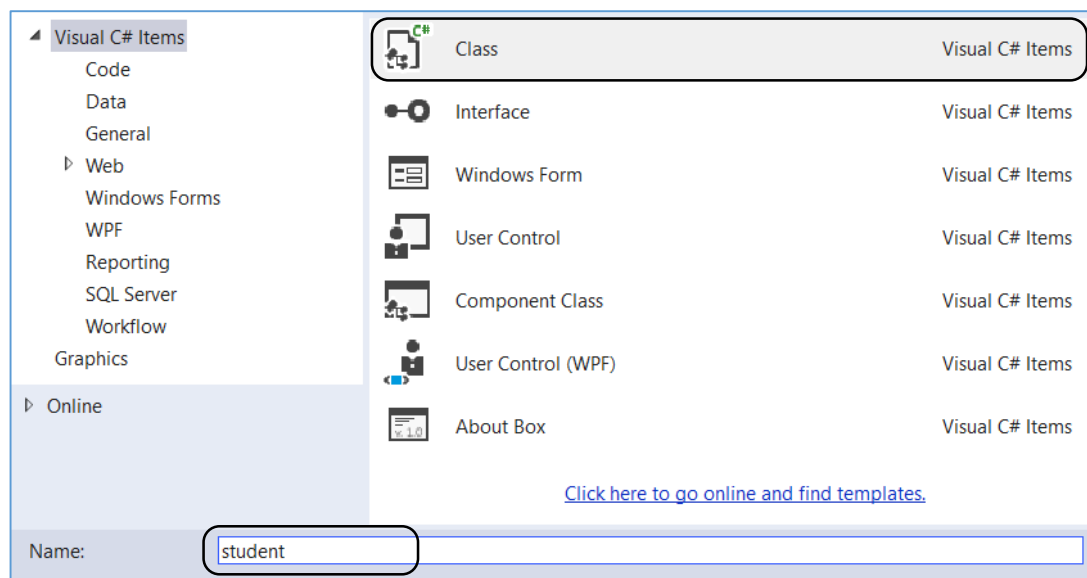
Date of Birth 

◀ 2000-2009 ▶

1999	2000	2001	2002
2003	2004	2005	2006
2007	2008	2009	2010

It is now necessary to create a **student** class file. Go to the **Solution Explorer** window, right-click the **collegeCourses** program icon, and select '**Add / New Item**'.

Choose '**Class**', and give the name '**student**':



We will first set up a class property to record the total number of students input, and an array to hold up to 12 *student* objects. We can then add the *surname*, *forename* and *date of birth* properties needed for a student object.

```
class student
{
    public static int studentCount;
    public static student[] studentObject = new student[12];

    private string surname;
    private string forename;
    private DateTime dateOfBirth;
}
```

Methods can then be added to transfer the *surname*, *forename* and *date of birth* into and out of the *student* object:

```
class student
{
    public static int studentCount;
    public static student[] studentObject = new student[12];

    private string surname;
    private string forename;
    private DateTime dateOfBirth;

    public void setSurname(string s)
    {
        surname = s;
    }

    public string getSurname()
    {
        return surname;
    }

    public void setForename(string f)
    {
        forename = f;
    }

    public string getForename()
    {
        return forename;
    }

    public void setDateOfBirth(DateTime d)
    {
        dateOfBirth = d;
    }

    public DateTime getDateOfBirth()
    {
        return dateOfBirth;
    }
}
```

We will complete the student class with a *saveStudent()* method, to transfer a student record into the database table. Add '*using Data*' and '*using SqlConnection*' directives, and give the database location:

```
using System.Linq;
using System.Text;

using System.Data;
using System.Data.SqlClient;

namespace collegeCourses
{
    class student
    {
        string databaseLocation = "C:\\C#\\collegeCourses.mdf;";

        public static int studentCount;
        public static student[] studentObject = new student[12];
    }
}
```


Insert the ***saveStudent()*** method into the class file after the list of properties:

```
private string surname;
private string forename;
private DateTime dateOfBirth;

public void saveStudent()
{
    SqlConnection con = new SqlConnection(@"Data Source=.\SQLEXPRESS;
        AttachDbFilename=" + databaseLocation + "Integrated Security=True;
        Connect Timeout=30;User Instance=True");

    con.Open();
    SqlCommand cmStudent = new SqlCommand();
    cmStudent.Connection = con;
    cmStudent.CommandType = CommandType.Text;
    cmStudent.CommandText = "INSERT INTO students(surname, forename, dateOfBirth)
        VALUES ('" + surname + "', '" + forename + "', '" +
        dateOfBirth.ToString("MM/dd/yyyy") + "')";
    cmStudent.ExecuteNonQuery();
    con.Close();
}
```

Return to ***Form1*** and add a line of code to the ***Form1()*** method to initialise the number of students to zero when the program first runs:

```
public Form1()
{
    InitializeComponent();
    course.courseCount = 0;

    student.studentCount = 0;

    loadCourses();
}
```

Go to the ***AddStudent*** form and double click the '***Add student***' button. Insert code into the `button_click` method to create a new ***student*** object:

```
private void btnAddStudent_Click(object sender, EventArgs e)
{
    string surname = txtSurname.Text;
    string forename = txtForename.Text;
    string d = Convert.ToString(dateTimePicker1.Value);
    DateTime dateOfBirth = Convert.ToDateTime(d);

    student.studentObject[student.studentCount] = new student();
    student.studentObject[student.studentCount].setSurname(surname);
    student.studentObject[student.studentCount].setForename(forename);
    student.studentObject[student.studentCount].setDateOfBirth(dateOfBirth);
    student.studentObject[student.studentCount].saveStudent();

    student.studentCount++;

    this.Close();
}
```

Run the program and enter test data for several students:

The screenshot shows a window titled 'AddStudent'. It has three input fields: 'Surname' with the text 'Young', 'Forename' with the text 'Sally', and 'Date of Birth' with a calendar picker showing '19 November 1996'. The calendar is open, displaying the month of November 1996, with the 19th highlighted.

Exit from the program and use the Server Explorer to open the students table in the database. Check that the student data has been saved correctly, then close the connection to the database.

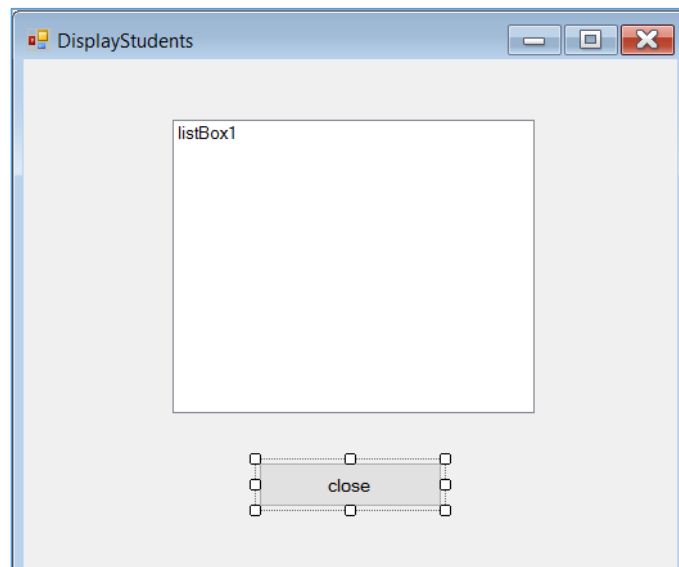
The screenshot shows the 'Server Explorer' window in SQL Server. Under 'Data Connections', 'collegeCourses.mdf' is expanded, and 'Tables' is selected. The 'students' table is highlighted. The table's data is displayed in the right pane:

surname	forename	dateOfBirth
Evans	Thomas	25/04/1992 00:00...
Pritchard	Stephen	06/06/1998 00:00...
Young	Sally	19/11/1996 00:00...
* NULL	NULL	NULL

With the student input form working, we can now create a **DisplayStudents** form. Go to the Solution Explorer window and add new item. Select **Windows Form** and give the name '**DisplayStudents**'.

The screenshot shows the 'Add New Item' dialog box in Visual Studio. The 'Visual C# Items' category is selected on the left. In the center, the 'Windows Form' template is highlighted. At the bottom, the 'Name' field contains the text 'DisplayStudents'.

Add a **listBox** and **button** to the form.



Right-click on the form and select the program code view.

We will display students' details in the list box in the format: **date of birth, surname, forename**. For example: **19/11/1996 Young, Sally**

Create a **studentList()** method, and call this from the **DisplayStudents()** method:

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

    private void studentList()
    {
        int studentCount = student.studentCount;
        string surname;
        string forename;
        DateTime dateOfBirth;

        listBox1.Items.Clear();
        for (int i = 0; i < studentCount; i++)
        {
            surname = student.studentObject[i].getSurname();
            forename = student.studentObject[i].getForename();
            dateOfBirth = student.studentObject[i].getDateOfBirth();

            listBox1.Items.Add(dateOfBirth.ToString("dd/MM/yyyy")
                               + " " + surname + ", " + forename);
        }
    }
}
```

Notice how the number of students is obtained from the **student** class, then a loop is used to access the **surname, forename** and **date of birth** from each of the objects in the **studentObject** array.

To complete the display of student details, return to **Form1** and add a **loadStudents()** method. Call this from the **Form1()** method:

```
public Form1()
{
    InitializeComponent();
    course.courseCount = 0;
    student.studentCount = 0;
    loadCourses();

    loadStudents();
}

private void loadStudents()
{
    DataSet dsStudents = new DataSet();

    SqlConnection con = new SqlConnection(@"Data Source=.\SQLEXPRESS;
        AttachDbFilename=" + databaseLocation + "Integrated Security=True;
        Connect Timeout=30; User Instance=True");
    try
    {
        con.Open();
        SqlCommand cmStudents = new SqlCommand();
        cmStudents.Connection = con;
        cmStudents.CommandType = CommandType.Text;
        cmStudents.CommandText = "SELECT * FROM students";
        SqlDataAdapter daStudents = new SqlDataAdapter(cmStudents);
        daStudents.Fill(dsStudents);
        con.Close();

        int countRecords = dsStudents.Tables[0].Rows.Count;

        for (int i = 0; i < countRecords; i++)
        {
            DataRow drStudent = dsStudents.Tables[0].Rows[i];
            string surname = Convert.ToString(drStudent[0]);
            string forename = Convert.ToString(drStudent[1]);
            DateTime dateOfBirth = Convert.ToDateTime(drStudent[2]);

            student.studentObject[student.studentCount] = new student();
            student.studentObject[student.studentCount].setSurname(surname);
            student.studentObject[student.studentCount].setForename(forename);
            student.studentObject[student.studentCount].setDateOfBirth(dateOfBirth);

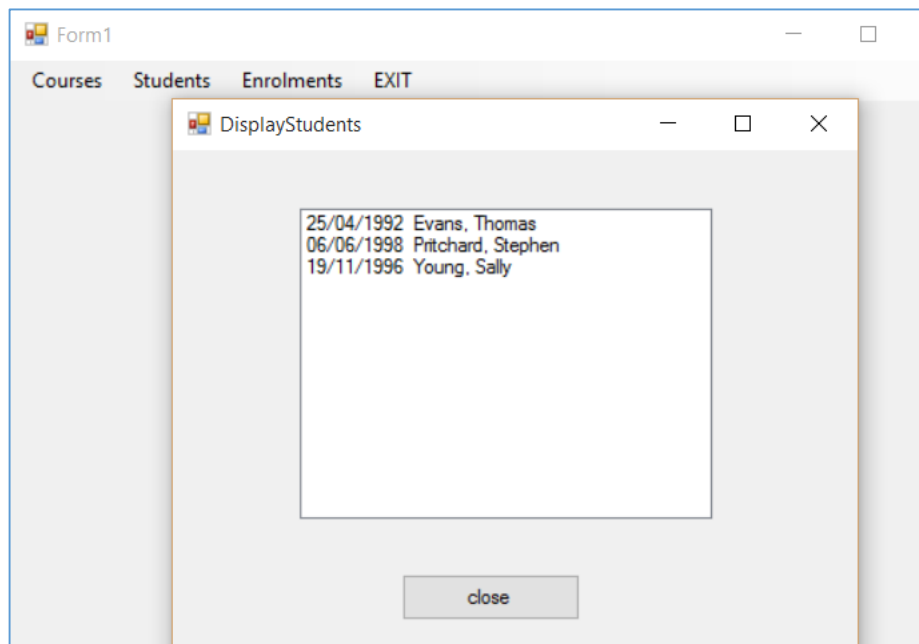
            student.studentCount++;
        }
    }
    catch
    {
        MessageBox.Show("File error");
    }
}
```

This method works in a similar way to **loadCourses()** which you wrote earlier. A loop is used to collect the **surname**, **forename** and **date of birth** from each student record in the data set. A new **studentObject** is created, and we set its properties using this data.

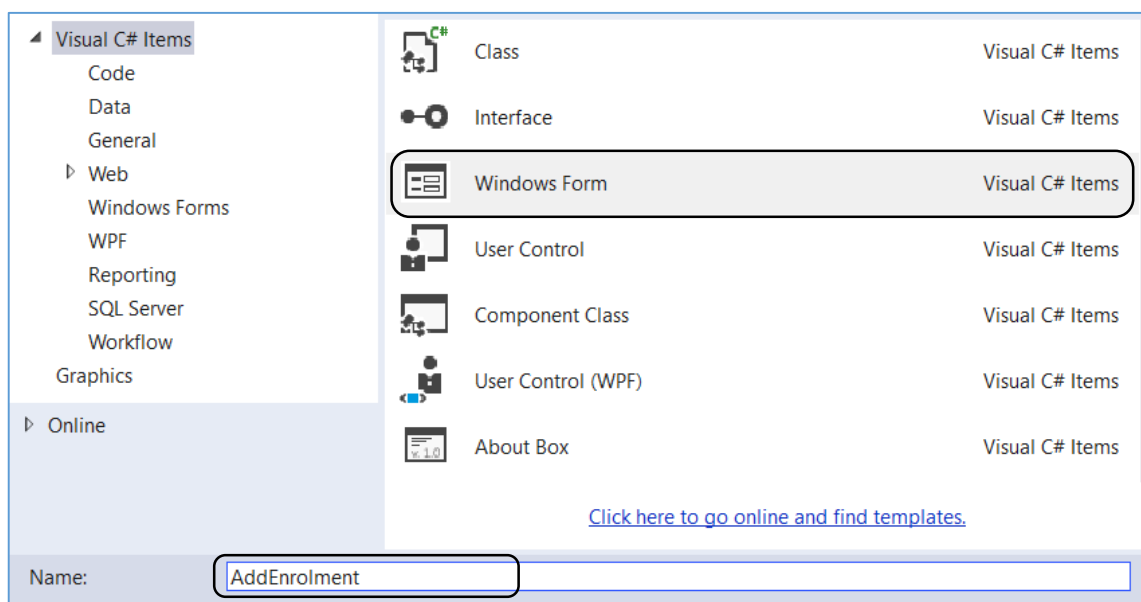
Double click the '**Display students**' menu option and add code to open the **DisplayStudents** form:

```
private void displayStudentsToolStripMenuItem_Click(object sender, EventArgs e)
{
    DisplayStudents frmDisplayStudents = new DisplayStudents();
    frmDisplayStudents.ShowDialog();
}
```

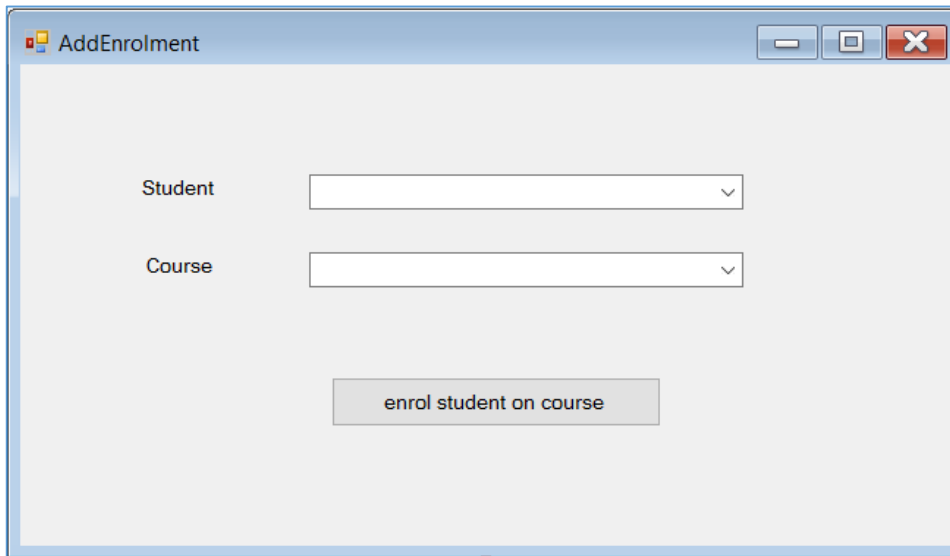
Run the program. Select the '**Display students**' option, and check that your student test data is shown correctly.



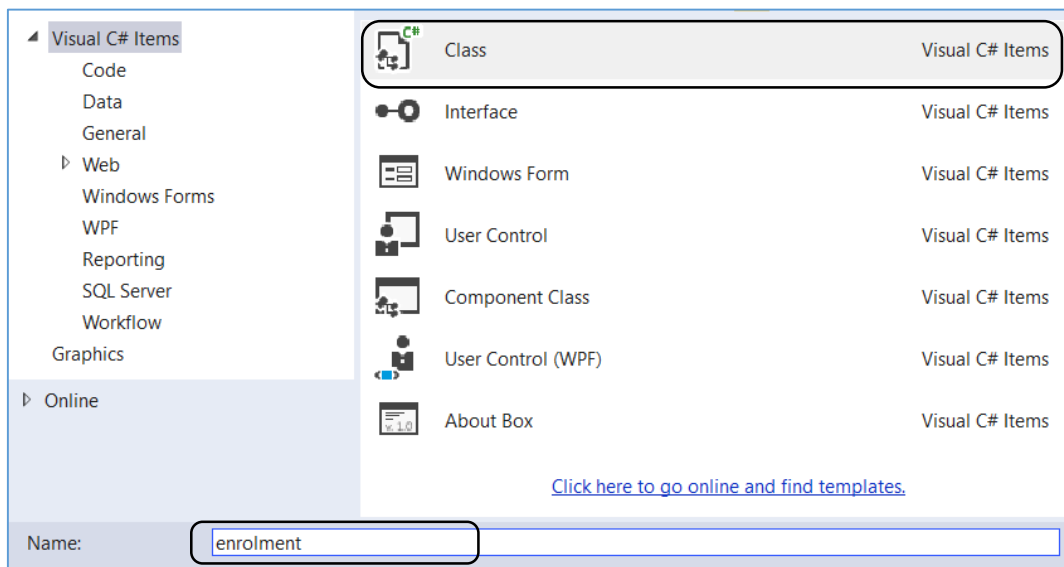
The final stage of the program is to handle the enrolment of students on courses. Create a new **Windows Form** and give this the name '**AddEnrolment**'.



Add two **ComboBoxes** to the form, along with **labels**. Complete the form with a **button** to enrol a student on a course. Name the button as **btnAddEnrolment**.



We will create an **enrolment** class, as we did for courses and students. Go to the **Solution Explorer** and right-click on the **collegeCourses** program to select '**Add /New item**'. Choose '**Class**', and give the name '**enrolment**'.



Add a property to store the **total number of enrolments**, an array for **enrolment objects**, and **course title** and **student name** properties for each enrolment recorded:

```
class enrolment
{
    public static int enrolCount;
    public static enrolment[] enrolObject = new enrolment[16];

    private string course;
    private string student;
}
```

Add methods to the **enrolment** class for transferring the **course title** and **student name** into and out of an enrolment object:

```
class enrolment
{
    public static int enrolCount;
    public static enrolment[] enrolObject = new enrolment[16];

    private string course;
    private string student;

    public void setCourse(string c)
    {
        course = c;
    }

    public string getCourse()
    {
        return course;
    }

    public void setStudent(string s)
    {
        student = s;
    }

    public string getStudent()
    {
        return student;
    }
}
```

The final requirement for the **enrolment** class is a method to store enrolment records in the database. Begin by adding '**using Data**' and '**using SqlClient**' directives, and giving the database location:

```
using System.Linq;
using System.Text;

using System.Data;
using System.Data.SqlClient;

namespace collegeCourses
{
    class enrolment
    {
        string databaseLocation = "C:\\C#\\collegeCourses.mdf;";

        public static int enrolCount;
        public static enrolment[] enrolObject = new enrolment[16];
    }
}
```

The ***saveEnrolment()*** method can now be written:

```
private string course;
private string student;

public void saveEnrolment()
{
    SqlConnection con = new SqlConnection(@"Data Source=.\\SQLEXPRESS;
        AttachDbFilename=" + databaseLocation + "Integrated Security=True;
        Connect Timeout=30; User Instance=True");

    con.Open();
    SqlCommand cmEnrol = new SqlCommand();
    cmEnrol.Connection = con;
    cmEnrol.CommandType = CommandType.Text;
    cmEnrol.CommandText = "INSERT INTO enrolments(student, course)
        VALUES ('" + student + "', '" + course + "')";
    cmEnrol.ExecuteNonQuery();
    con.Close();
}

public void setCourse(string c)
{
    course = c;
}
```

Go to ***Form1*** and add a line of code to the ***Form1()*** method to initialise the number of enrolments to zero when the program begins:

```
public Form1()
{
    InitializeComponent();
    course.courseCount = 0;
    student.studentCount = 0;

    enrolment.enrolCount = 0;

    loadCourses();
    loadStudents();
}
```

Double click the '***Enrol student on course***' menu option, then add code to open the ***addEnrolment*** form.

```
private void enrolStudentToolStripMenuItem_Click(object sender, EventArgs e)
{
    AddEnrolment frmAddEnrolment = new AddEnrolment();
    frmAddEnrolment.ShowDialog();
}
```


Open the program code view for the **addEnrolment** form. This form has two **comboBoxes** which should provide drop-down lists of student names and course titles. It will then be possible to make an enrolment by selecting the relevant **student** and **course** from the lists, as in the design below:

Begin by producing a **loadData()** method. This uses loops to obtain the student and course data from the arrays of objects, and then inserts this data into the **comboBox** drop-down lists. Add a line of code to call the **loadData()** method from the **AddEnrolment()** method when the form is opened.

```
public AddEnrolment()
{
    InitializeComponent();
    loadData();
}

private void loadData()
{
    int studentCount = student.studentCount;
    string surname;
    string forename;
    DateTime dateOfBirth;

    comboBox1.Items.Clear();
    for (int i = 0; i < studentCount; i++)
    {
        surname = student.studentObject[i].getSurname();
        forename = student.studentObject[i].getForename();
        dateOfBirth = student.studentObject[i].getDateOfBirth();

        comboBox1.Items.Add(dateOfBirth.ToString("dd/MM/yyyy")
            + " " + surname + ", " + forename);
    }

    int courseCount = course.courseCount;
    string title;

    comboBox2.Items.Clear();
    for (int i = 0; i < courseCount; i++)
    {
        title = course.courseObject[i].getTitle();
        comboBox2.Items.Add(title);
    }
}
```

Move to the form design view and double click the '**enrol student on course**' button. Add code to the `button_click` method to collect the selected **student** and **course** names from the **comboBoxes**, create a new **enrolment** object, set the properties of the object, then save the data into the database file.

```
private void btnAddEnrolment_Click(object sender, EventArgs e)
{
    string student = comboBox1.Text;
    string course = comboBox2.Text;

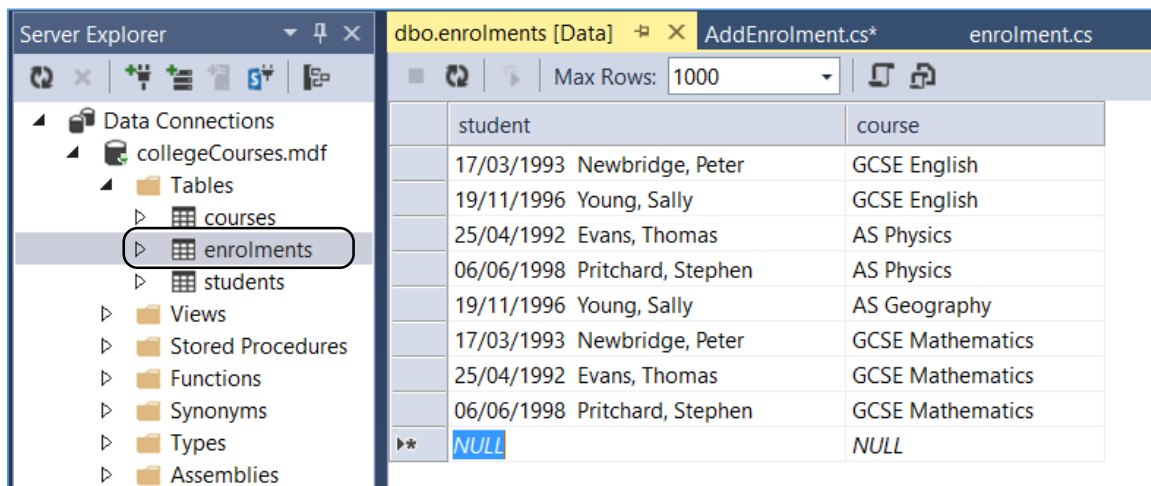
    enrolment.enrolObject[enrolment.enrolCount] = new enrolment();
    enrolment.enrolObject[enrolment.enrolCount].setStudent(student);
    enrolment.enrolObject[enrolment.enrolCount].setCourse(course);
    enrolment.enrolObject[enrolment.enrolCount].saveEnrolment();

    enrolment.enrolCount++;

    this.Close();
}
```

Run the program. Go to the **Add enrolment** form and select combinations of **students** and **courses**, then click the button to enter each enrolment.

Exit from the program and use the Server Explorer to open the collegeCourses database. Check that the enrolments have been recorded correctly in the enrolments table, then close the database connection.



student	course
17/03/1993 Newbridge, Peter	GCSE English
19/11/1996 Young, Sally	GCSE English
25/04/1992 Evans, Thomas	AS Physics
06/06/1998 Pritchard, Stephen	AS Physics
19/11/1996 Young, Sally	AS Geography
17/03/1993 Newbridge, Peter	GCSE Mathematics
25/04/1992 Evans, Thomas	GCSE Mathematics
06/06/1998 Pritchard, Stephen	GCSE Mathematics
NULL	NULL

We can now consider how the enrolments can be displayed by the program. It will be useful for the staff of the college if this can be done in two different ways:

- As a **list of the courses**, showing the students enrolled on each course.
- As a **list of students**, showing the courses for which each student is enrolled.

We will provide these options as two separate forms.

Return to **Form1** and add a **loadEnrolments()** method. This will work in a very similar way to the methods which you wrote earlier to load course and student records.

Call the **loadEnrolments()** method from the **Form1()** method.

```
public Form1()
{
    InitializeComponent();
    course.courseCount = 0;
    student.studentCount = 0;
    enrolment.enrolCount = 0;
    loadCourses();
    loadStudents();

    loadEnrolments();
}

private void loadEnrolments()
{
    DataSet dsEnrol = new DataSet();

    SqlConnection con = new SqlConnection(@"Data Source=.\\SQLEXPRESS;
        AttachDbFilename=" + databaseLocation + "Integrated Security=True;
        Connect Timeout=30; User Instance=True");
    try
    {
        con.Open();
        SqlCommand cmEnrol = new SqlCommand();
        cmEnrol.Connection = con;
        cmEnrol.CommandType = CommandType.Text;
        cmEnrol.CommandText = "SELECT * FROM enrolments";
        SqlDataAdapter daEnrol = new SqlDataAdapter(cmEnrol);
        daEnrol.Fill(dsEnrol);
        con.Close();

        int countRecords = dsEnrol.Tables[0].Rows.Count;

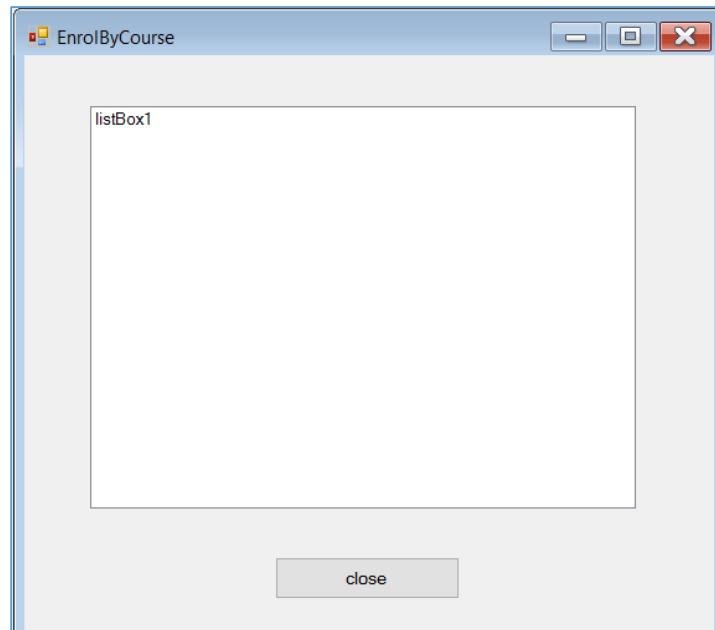
        for (int i = 0; i < countRecords; i++)
        {
            DataRow drEnrol = dsEnrol.Tables[0].Rows[i];
            string student = Convert.ToString(drEnrol[0]);
            string course = Convert.ToString(drEnrol[1]);

            enrolment.enrolObject[enrolment.enrolCount] = new enrolment();
            enrolment.enrolObject[enrolment.enrolCount].setStudent(student);
            enrolment.enrolObject[enrolment.enrolCount].setCourse(course);

            enrolment.enrolCount++;
        }
    }
    catch
    {
        MessageBox.Show("File error");
    }
}
```

Create a new **Windows Form** and give this the name '**EnrolByCourse**'.

Add a **listBox** and **button** to the form:



Go to **Form1**, double click the '**Display enrolments by course**' menu option and add lines of code to open the form:

```
private void displayEnrolmentsByCourseToolStripMenuItem_Click
(object sender, EventArgs e)
{
    EnrolByCourse frmEnrolByCourse = new EnrolByCourse();
    frmEnrolByCourse.ShowDialog();
}
```

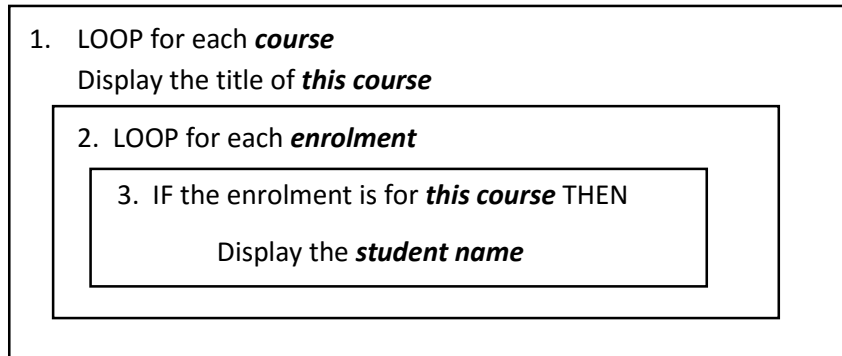
Change to the program code view for the **EnrolByCourse** form. Create a **courseList()** method and call this from the **EnrolByCourse()** method. Add variables to hold the **student** and **course** properties when they are accessed from the object classes.

```
public EnrolByCourse()
{
    InitializeComponent();
    courseList();
}

private void courseList()
{
    int courseCount = course.courseCount;
    string title;

    int enrolCount = enrolment.enrolCount;
    string student;
    string courseName;
}
```

The strategy we will use to display the courses, showing the students enrolled on each course, is:



Add lines of code to the **courseList()** method to operate the loops and IF condition:

```

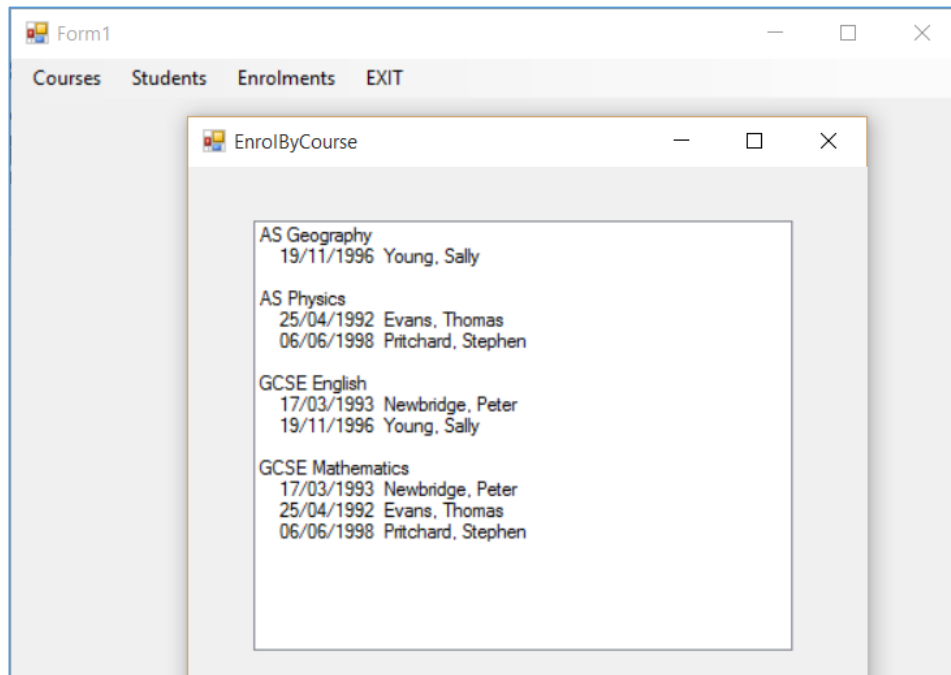
private void courseList()
{
    int courseCount = course.courseCount;
    string title;

    int enrolCount = enrolment.enrolCount;
    string student;
    string courseName;

    listBox1.Items.Clear();
    for (int i = 0; i < courseCount; i++)
    {
        title = course.courseObject[i].getTitle();
        listBox1.Items.Add(title);
        for (int j = 0; j < enrolCount; j++)
        {
            student = enrolment.enrolObject[j].getStudent();
            courseName = enrolment.enrolObject[j].getCourse();
            if (courseName == title)
            {
                listBox1.Items.Add("    "+student);
            }
        }
        listBox1.Items.Add("");
    }
}
  
```

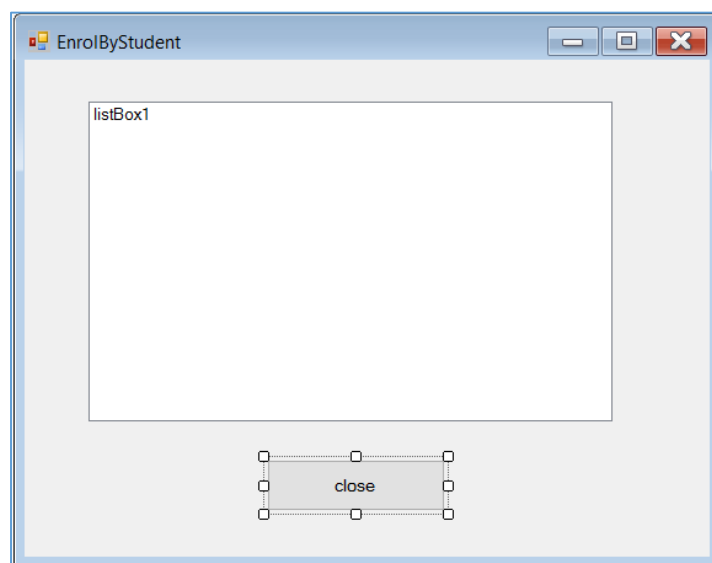
Run the program and select the '**Display enrolments by course**' menu option.

The **EnrolByCourse** form should open, with your test data displayed in the listBox in a similar way to the example below.



The final task in this program is to display a list of students, showing the courses for which each student is enrolled.

Create another **Windows Form** and name this '**EnrolByStudent**'. Add a list box and button as before:



The strategy we will use to display the students, showing the courses for which they are enrolled, is:

1. LOOP for each **student**
Display the name of **this student**
 2. LOOP for each **enrolment**
 3. IF the enrolment is for **this student** THEN
Display the **course title**

Right-click on the **EnrolByStudent** form to change to the Code view. Create a **studentList()** method to implement the algorithm. Call the **studentList()** method from the **EnrolByStudent()** method.

```
public EnrolByStudent()
{
    InitializeComponent();

    studentList();
}

private void studentList()
{
    int studentCount = student.studentCount;
    string surname;
    string forename;
    DateTime dateOfBirth;

    int enrolCount = enrolment.enrolCount;
    string studentName;
    string courseName;

    listBox1.Items.Clear();
    for (int i = 0; i < studentCount; i++)
    {
        surname = student.studentObject[i].getSurname();
        forename = student.studentObject[i].getForename();
        dateOfBirth = student.studentObject[i].getDateOfBirth();
        string s = dateOfBirth.ToString("dd/MM/yyyy") + " "
            + surname + ", " + forename;
        listBox1.Items.Add(s);
        for (int j = 0; j < enrolCount; j++)
        {
            studentName = enrolment.enrolObject[j].getStudent();
            courseName = enrolment.enrolObject[j].getCourse();
            if (studentName == s)
            {
                listBox1.Items.Add("    " + courseName);
            }
        }
        listBox1.Items.Add("");
    }
}
```

Go to **Form1**, double click the '**Display enrolments by student**' menu option and add lines of code to open the form:

```
private void displayEnrolmentsByStudentToolStripMenuItem_Click
    (object sender, EventArgs e)
{
    EnrolByStudent frmEnrolByStudent = new EnrolByStudent();
    frmEnrolByStudent.ShowDialog();
}
```

Run the program and check that a correct list of students is produced from your enrolment data:

