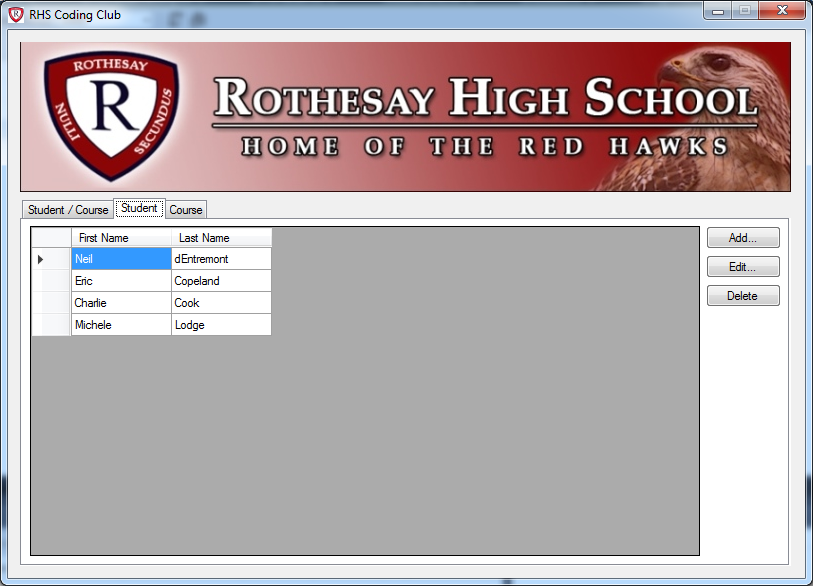
RHS Coding Club

Put together a small app that allows the developer to work with databases and visual development using the development language C# (C-Sharp).



In order to put an application together, a good place to start is from the ground up, so we’ll begin with the data and database.

Database

The database we are using is SQL Server. SQL Server is a relational database. A relational database management system (RDBMS) is a database management system (DBMS) that is based on the relational model as invented by E. F. Codd, of IBM's San Jose Research Laboratory. Many popular databases currently in use are based on the relational database model.

RDBMSs are a common choice for the storage of information in new databases used for financial records, manufacturing and logistical information, personnel data, and other applications since the 1980s. Relational databases have often replaced legacy hierarchical databases and network databases because they are easier to understand and use. However, relational databases have been challenged by object databases, which were introduced in an attempt to address the object-relational impedance mismatch in relational databases, and XML databases.

<http://en.wikipedia.org/wiki/Relational_database_management_system>

W3Schools.com is a great site for SQL documentation.

<http://www.w3schools.com/sql/>

**Tables**

We created tables last class. In this class use the SQL provided to regenerate your table and level set with the rest of the class.

**Inserting Data**

Use the SQL provided to load data into your Student and Course tables

**Querying Data**

Using the W3Schools site, research how to select (query) data.

**Exercise:**

* Select all Student records.
  + Hint: W3Schools SQL SELECT, SELECT \*
* Select the first name and last name of all Student records.
  + Hint: W3Schools SQL SELECT
* Select (and combine) the first name and last name of all Student records.
  + Hint: W3Schools SQL SELECT, field1 + field2
* Select Student records whose last name begins with ‘C’.
  + Hint: W3Schools SQL LIKE
* Select all Course records.
* Select all Computer Science courses.
* Select all 112 courses.
* Select all Computer Science AND 112 courses.
  + Hint: W3Schools SQL SELECT, AND
* Select all Computer Science OR 112 courses.
  + Hint: W3Schools SQL SELECT, OR

**Joining Tables**

Using the W3Schools site, research how to do joins. SQL Joins

**Exercise**

Using the StudentCourse table and joining to the Student and Course table, create a Select query to return the following data. To create column names like below, see **SQL Aliases**.

|  |  |  |
| --- | --- | --- |
| Student Name | Course (Level) | Mark |

For Example,

|  |  |  |
| --- | --- | --- |
| Neil dEntremont | Computer Science (112) | 95 |

**Stored Procedures**

<http://databases.about.com/od/sqlserver/a/storedprocedure.htm>

<https://www.youtube.com/watch?v=fjNsRV4zLdc> – by WiseOwl (there are my other good videos)

<http://stackoverflow.com/questions/459457/what-is-a-stored-procedure> - StackOverflow is another great technical resource.

**Exercise**

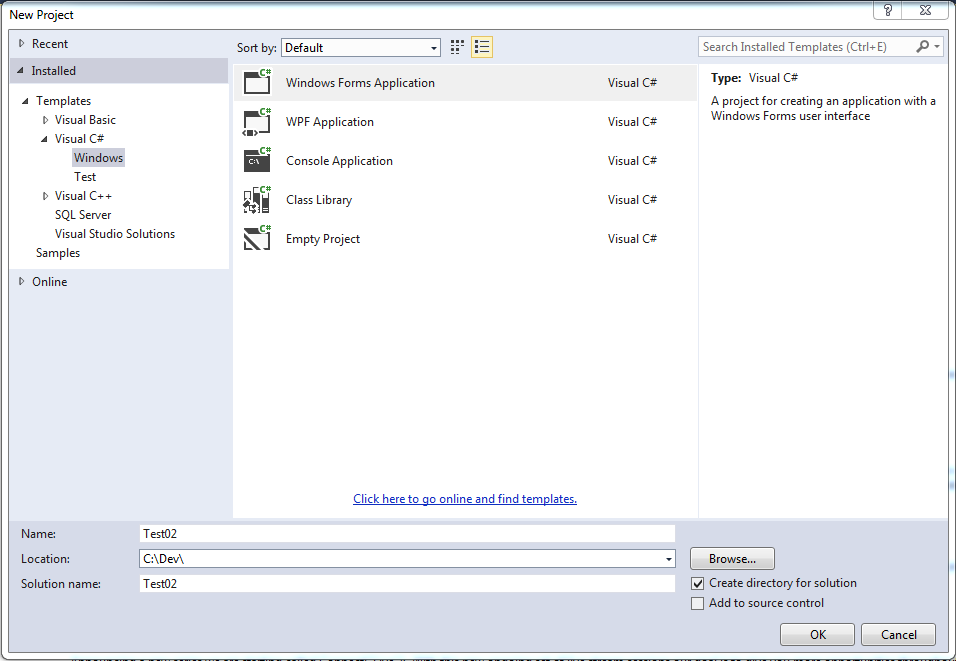
* Create a stored procedure to…
  + Return all Student records.
  + Insert a Student record by taking first name and last name as parameters.
  + Update a Student record by taking id, first name and last name as parameters.
    - Hint – look at SQL Update to get an understanding of Update.

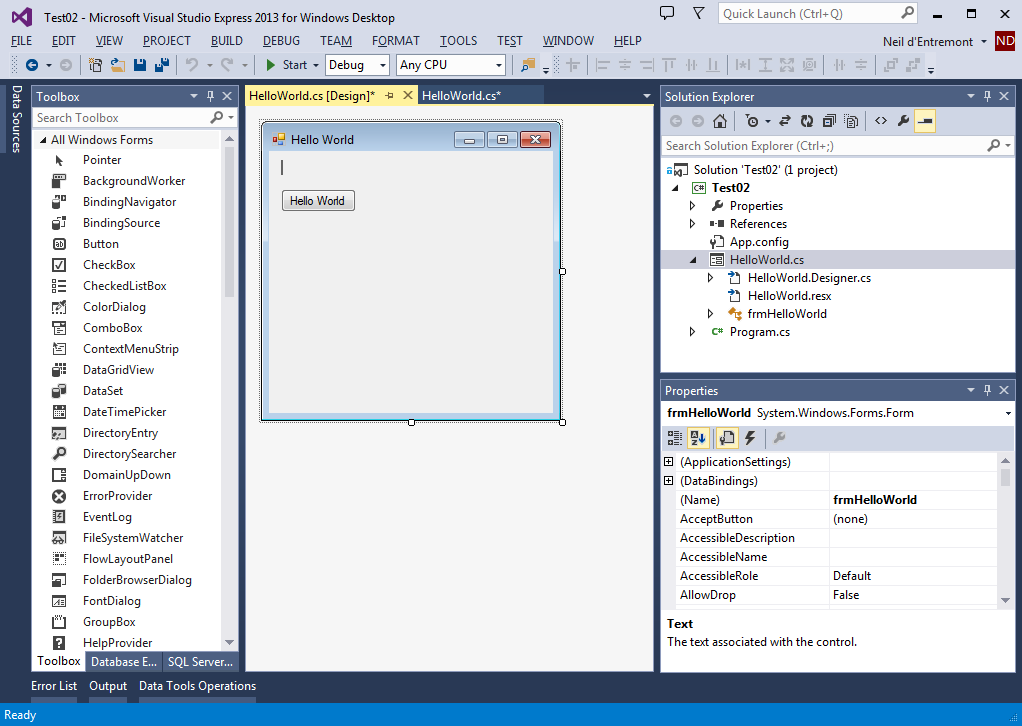
Visual Studio Application

Time to Create a Visual Studio Project

Let’s start with a test app to gain some basic understanding of creating an app, adding some controls and coding for an event.

Within Visual Studio select File / New Project. Pick Windows Form Application and give your app a name.





* Rename Form1 to HelloWorld.
* HelloWorld Form

|  |  |
| --- | --- |
| Property | Value |
| (Name) | frmHelloWorld |
| Text | Hello World |

* Add a label

|  |  |
| --- | --- |
| Property | Value |
| (Name) | lblHelloWorld |
| BorderStyle | FixedSingle |

* Add a button

|  |  |
| --- | --- |
| Property | Value |
| (Name) | btnHelloWorld |
| Text | Hello World |

Modify the Click event of the button to load the text string ‘Hello World’ into the Text property of the label.

* Double click the Hello World button

private void btnHelloWorld\_Click(object sender, EventArgs e)

{

lblHelloWorld.Text = "Hello World";

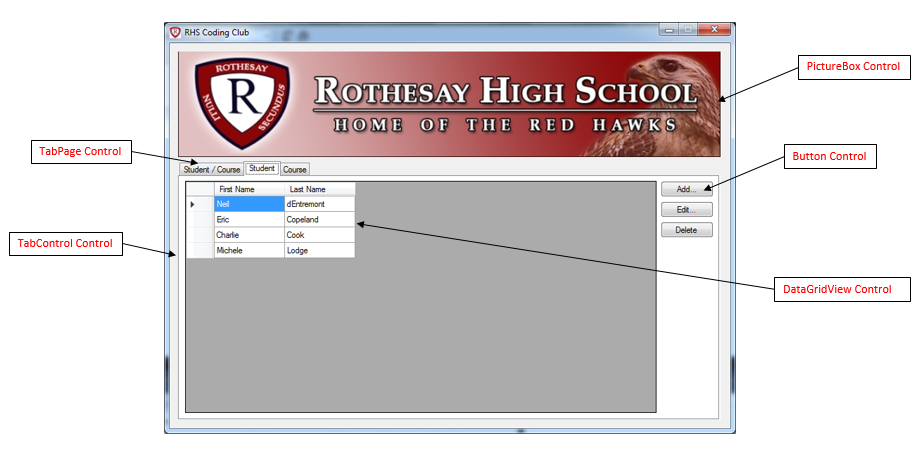
}

Student / Course Application

Time to create the Student / Course application.

Within Visual Studio select File / New Project. Pick Windows Form Application and give your app a name.

Build the main application page.



* Rename Form1 to Main.
* Main Form

|  |  |
| --- | --- |
| Property | Value |
| (Name) | frmMain |
| Text | RHS Coding Club |

* PictureBox Control

|  |  |
| --- | --- |
| Property | Value |
| (Name) | picRHS |
| Image | Select an image to load here |

* TabControl Control

|  |  |
| --- | --- |
| Property | Value |
| (Name) | tabStudentCourse |
| TabPages | Add 3 tab pages |

* TabPage Control

|  |  |
| --- | --- |
| Property | Value |
| (Name) | tabPageStudentCourse |

* TabPage Control

|  |  |
| --- | --- |
| Property | Value |
| (Name) | tabPageStudent |

* TabPage Control

|  |  |
| --- | --- |
| Property | Value |
| (Name) | tabPageCourse |

* Within the Student tab
* DataGridView Control

|  |  |
| --- | --- |
| Property | Value |
| (Name) | grdStudent |

* Button Control

|  |  |
| --- | --- |
| Property | Value |
| (Name) | btnStudentAdd |
| Text | Add… |

* Button Control

|  |  |
| --- | --- |
| Property | Value |
| (Name) | btnStudentEdit |
| Text | Edit… |

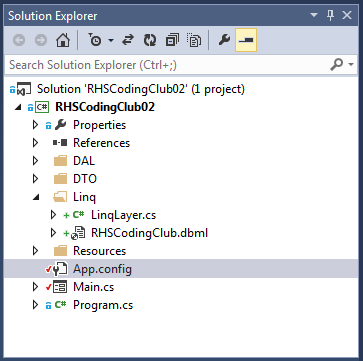
* Button Control

|  |  |
| --- | --- |
| Property | Value |
| (Name) | btnStudentDelete |
| Text | Delete |

**Database Layer**

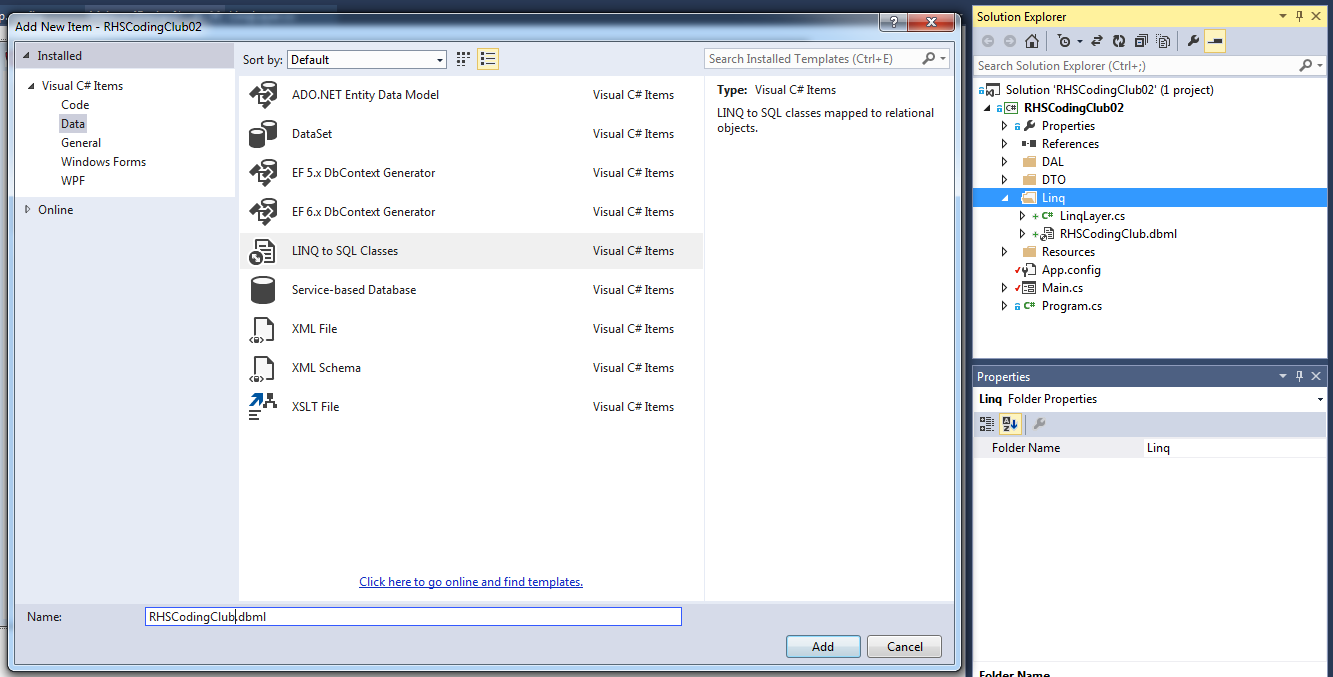
In order for the application to communicate to the database we need to configure a database connection framework. For this application we’ll use the Microsoft Linq Framework. Linq is a data query language used for access data. Its concepts are similar to SQL that we’ve been learning for access data.

Within Solution Explorer create a Linq folder where we will some Linq framework files.

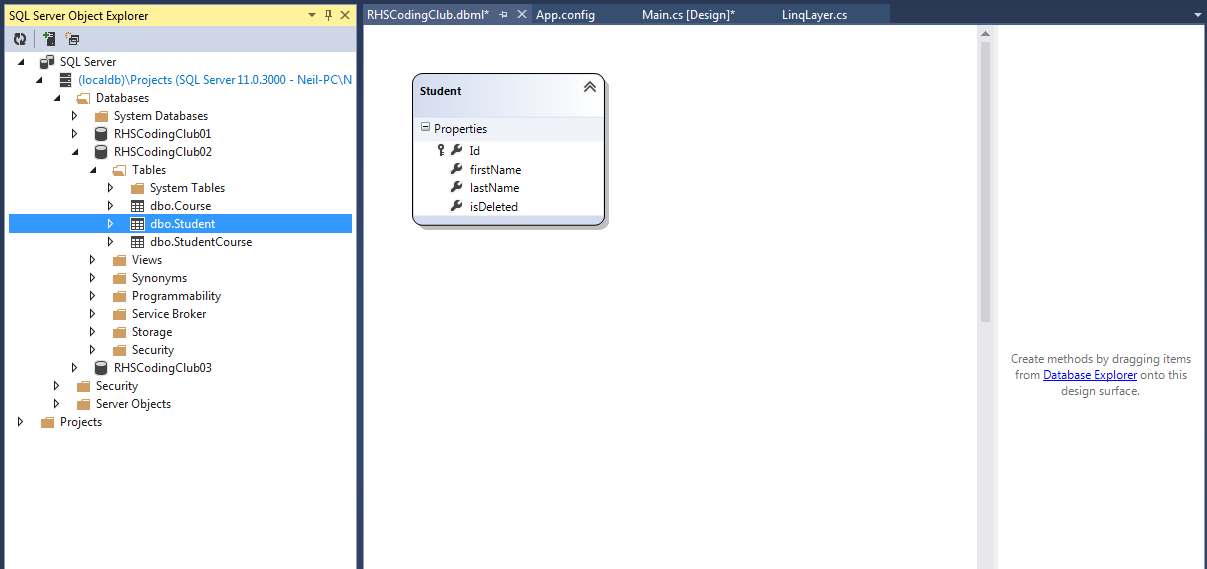


Right-click on the Linq folder. Select Add / New Item / Data / Linq to SQL Classes.

Name the file RHSCodingClub.dbml.



With the SQL Server Object Explorer, drag the Student table onto the open Linq file. This operation will make a connection between the application and the database.

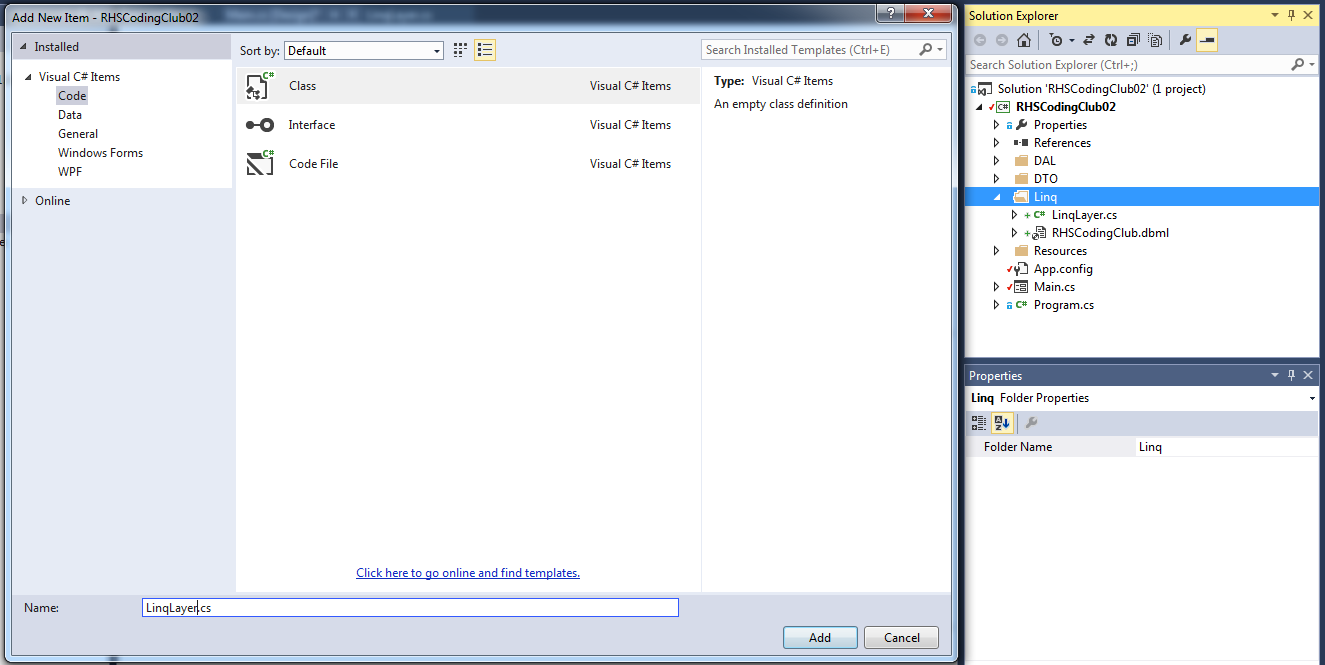


You can save and close the dbml file.

Now we must add LinqLayer code file.

Right-click on the Linq folder. Select Add / New Item / Code / Class.

Name the file LinqLayer.cs



Open the file LinqLayer.cs. Within this file we’ll write code to access the Student table (functionality like selecting data, adding data, deleting data and updating data). The following links are a great tutorial on Microsoft Linq. We’ll be using those concepts as we develop Linq queries for accessing data.

<http://weblogs.asp.net/scottgu/using-linq-to-sql-part-1>

<http://weblogs.asp.net/scottgu/linq-to-sql-part-2-defining-our-data-model-classes>

<http://weblogs.asp.net/scottgu/linq-to-sql-part-3-querying-our-database>

<http://weblogs.asp.net/scottgu/linq-to-sql-part-4-updating-our-database>

<http://weblogs.asp.net/scottgu/linq-to-sql-part-5-binding-ui-using-the-asp-linqdatasource-control>

<http://weblogs.asp.net/scottgu/linq-to-sql-part-6-retrieving-data-using-stored-procedures>

<http://weblogs.asp.net/scottgu/linq-to-sql-part-7-updating-our-database-using-stored-procedures>

<http://weblogs.asp.net/scottgu/linq-to-sql-part-8-executing-custom-sql-expressions>

<http://weblogs.asp.net/scottgu/linq-to-sql-part-9-using-a-custom-linq-expression-with-the-lt-asp-linqdatasource-gt-control>

With the LinqLayer.cs file open we’ll add code to start performing some database interactions. Here is some sample code to help you start.

namespace RHSCodingClub02.Linq

{

public class LinqLayer

{

private static RHSCodingClubDataContext db = new RHSCodingClubDataContext();

public static DataTable GetAllStudents()

{

//get all student records where that are not flagged as deleted (0)

var students = from s in db.Students where s.isDeleted.Equals(0) select new { s.Id, s.firstName, s.lastName, s.isDeleted };

DataTable dtStudents = new DataTable("Students");

dtStudents.Columns.Add("Id");

dtStudents.Columns.Add("First Name");

dtStudents.Columns.Add("Last Name");

dtStudents.Columns.Add("isDeleted");

foreach (var student in students)

{

dtStudents.Rows.Add(student.Id, student.firstName, student.lastName, student.isDeleted);

}

return dtStudents;

}

public static void AddStudent(string firstName, string lastName)

{

//add a new student record by suppling the firstname and lastname

Student student = new Student();

student.firstName = firstName;

student.lastName = lastName;

db.Students.InsertOnSubmit(student);

db.SubmitChanges();

}

public static void DeleteStudent(int id)

{

//delete a student by setting its isDeleted flag to 1

Student student = db.Students.Single(s => s.Id.Equals(id));

student.isDeleted = 1;

db.SubmitChanges();

}

public static void UpdateStudentName(int id, string firstName, string lastName)

{

//update a student record by updating its first and last name

Student student = db.Students.Single(s => s.Id.Equals(id));

student.firstName = firstName;

student.lastName = lastName;

db.SubmitChanges();

}

}

Testing the LingLayer.cs code.

In order to test the code, we’ll use the buttons that we added to the Student tab.

Double-click the Add button. This will open the code page and create the shell of a button click event.

Add your test code to this event.

private void btnStudentAdd\_Click(object sender, EventArgs e)

{

LinqLayer.AddStudent("Fred", "Flintstone");

}

Now run the app, go to the Student tab, click the Add button to run the test, then check the database to see if the new student was added.

Test the UpdateStudentName and DeleteStudent code.