IMAT2204 Project Management and Development

Let's take a moment to stop and think!

It is far too easy to simply rush from one assessment to the next without actually taking the time to stop and think about what you are doing.

If we don't think about things then we are more likely to end up doing the wrong thing!

In this lecture what I want to do is talk about the project process for this module.

We will look at some of the deliverables you will need to produce over the module also the reasons why we need to produce them.

What is the Module About?

If we look at the module template (the official description of the module) we see the following learning outcomes…

1. Create an advanced prototype with suitable database functionality
2. Create the beginnings of a professional portfolio of work
3. Demonstrate skills allowing you to act as a computing professional
4. Demonstrate problem solving skills allowing you to adapt to the challenges of changing technology
5. Application of skills from other modules on the course

Apart from taking you by the hand on the development process of a reasonably complex problem there are a few things I want you to take away from this module.

Some of these things are skills I hope you will acquire…

- An understanding of important documentation artefacts
- Some idea of how to approach systems development problems

Some other things are more tangible…

- A better sense of your skills reflected in your CV
- A class library that you may use in other projects, especially your final year project

The Need for Courage

Taking on a project of significant size is going to test you in quite difficult ways.

Many books talk about the need for courage and I will raise this point too.

When taking on a project the first thing you will need to deal with is the sense of fear and intimidation that comes with it.

This sense of fear never really disappears no matter how long you have been doing this.

Certainly with more experience you get to a stage where certain problems are familiar to you. However the fact of the matter the starting point for any development project is the question

“How do I do this?”

Fear is in fact the starting point for many projects and we need to work out ways to deal with this fear.

Fear makes you tentative.

If you are in a position of fear you are less likely to be adventurous. You hold back, play it safe and don’t step out of your comfort zone.

Fear makes you less communicative.

When we are afraid the temptation will be to try and solve problems in isolation. We need to accept that this is a bad thing to do.

In industry we seldom have the option of working on our own (if ever).

The big advantage of working in a team is that we have the opportunity of making use of the expertise of other team members.

You may not know the answer but you probably know somebody who does.

Fear makes you shy away from feedback.

OK, you may work up the courage to ask for help but what happens if the advice you are given isn’t what you want to hear?

When working on a system we are inevitably going to get errors in our code. These errors typically don’t make us feel good about ourselves.

Fear is probably the first stumbling block that we need to deal with when starting a project.

Fear will make us clam up, withdraw and inevitably scale the project down to something we can manage.

The down side of doing this is that the project then becomes so small that it fails to meet its requirements.

Fear makes you shy away from mistakes

Do not get hung up on getting things right on the first attempt.

In creating the system it will be a learning process for you and the client.

You are going to draw up documentation that may seem OK at first but later on you will realise the design is flawed.

The same is true of your code. Even in the case of code that works correctly you will almost always find a better way of doing the same thing.

So what is the answer?

The answer is to muster up as much courage as you can.

You need to say to yourself "I have this job to do and I don't know the answer but I will rise to the challenge".

Finding this mind set is an important first step in tackling any project.

In so doing we end up coming face to face with our own ignorance and then moving on.

The Ethical Review

This may seem like a strange place to start but actually it is really important.

In the case of authenticating a user there are a number of ethical issues we need to think about.

For example…

Is the system of authentication as secure as we think it is?

An authentication system with hidden security vulnerabilities is probably worse than no security at all. If we think we are secure but the system has been hacked without our knowing that places our data in a very dangerous situation.

If we plan to store credit card details, personal details anything that that should be private we need to be absolutely sure that the data is safe.

If a hacker has found a way into our system and we don't know it they will just sit there quietly harvesting data from the system and we will be oblivious.

On the subject of personal data we also need to ask the question "how are we storing the user's password?"

The odds are that the password used in this system is the same as the one they have on other systems.

If we are storing passwords as plain text then what is to stop the system administrator simply opening the database table and printing off all of the user names and passwords?

Not only do we need to think about the way the password data is stored we also need to think about more general security on the server the database is stored on along with the physical security of the machine.

What happens if somebody breaks into the server room and simply takes the entire server?

Also what happens in the case of catastrophic failure?

It might not be a big deal when it comes to our student assignments but what if rather than books the system dealt with replacement organs? How fast could we get the system running in a catastrophic failure?

If this takes too long, are we ready for the litigation that might ensue if a customer suffers loss of trade due to our system failure?

The ethical angle is a useful starting place as it gets us thinking out the people involved and the kind of issues that may face us.

Keep it Simple

Don't be afraid to keep things simple too. There will be times when you need complex solutions to tricky problems but simplicity implies clarity. If you can create a solution to a problem that is clear and simple do it.

Resist the temptation to try and impress by trying to appear clever.

Dr. Abraham Erskine – The Need for Documentation

It will be interesting to see how many of you know who Dr Abraham Erskine is.

More of you know who it is than you think.

What if I said Captain America?

OK let me put you out of your misery.

Dr Erskine is the scientist in the Captain America film that invented the super soldier serum that turns Steve Rogers into Captain America.

Where am I going with this?

Well Dr Erskine is killed shortly after the transformation of Steve Rogers and the super soldier serum is lost for all time.

Move ahead a number of years and we have Dr Banner trying to re-create the serum using gamma rays. His experiment fails and turns into the Incredible Hulk.

In fact many of the Marvel film plots revolve around different people's attempts to recreate the super soldier serum in the absence of Dr Erskine.

So what is the problem?

There are two problems here which make me doubt Dr Erskine's credibility as a doctor.

1. He kept no notes on how he made his discovery
2. He clearly didn't keep a backup of his work

If Dr Erskine had kept detailed notes as to how then all S.H.I.E.L.D. would have needed to do to recreate the serum would be to read them and follow the instructions.

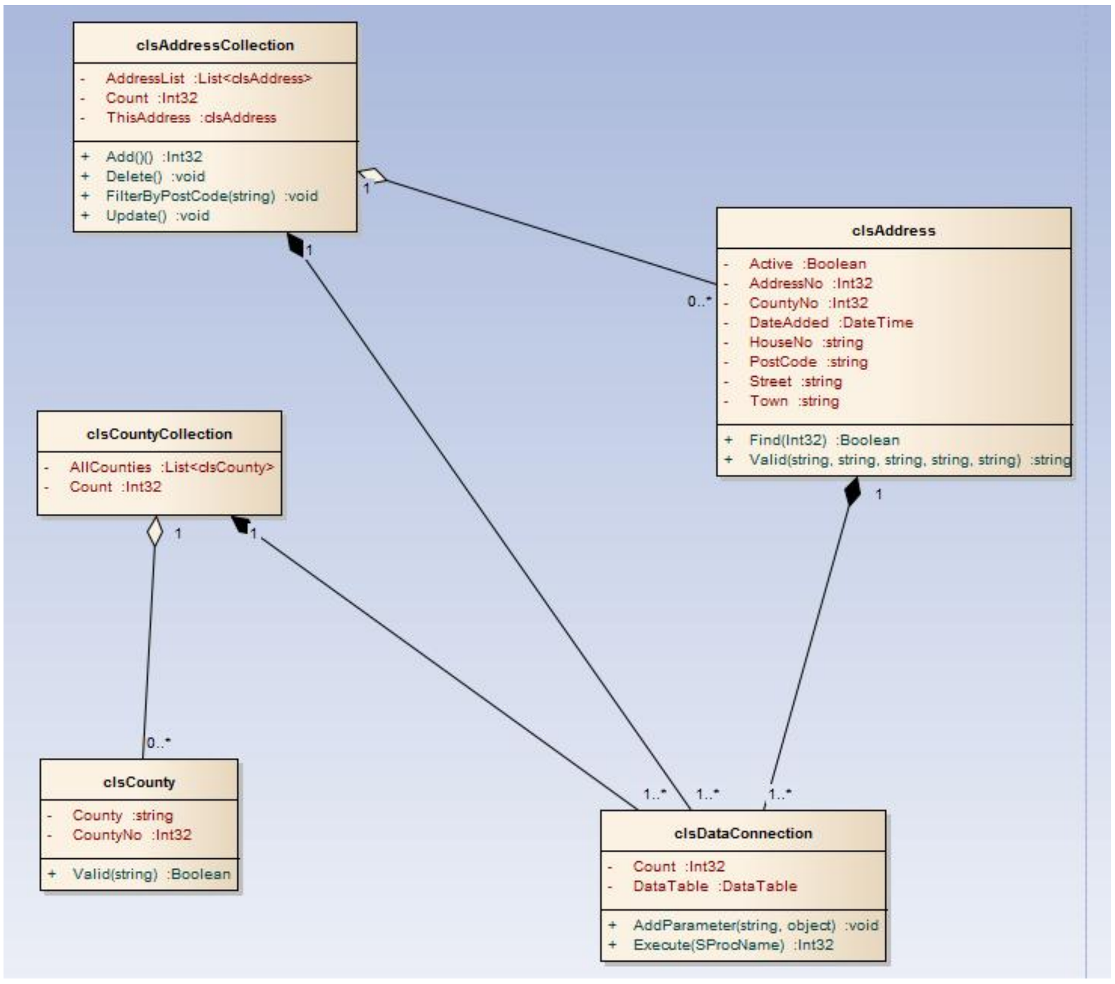
If there was a backup of the notes then no matter what happened the serum could still be recreated.

Admittedly if this was part of the story for the film it would have been a lot shorter but the point remains.

When working on a project of any type you need to keep suitable notes / documentation so that if anything happens to you somebody else can pick up where you left off.

Not only that but if you are new on the job and want to understand what is going on quickly having such documentation will make your life a lot easier.

For example take a look at the class diagram below for the address book application.



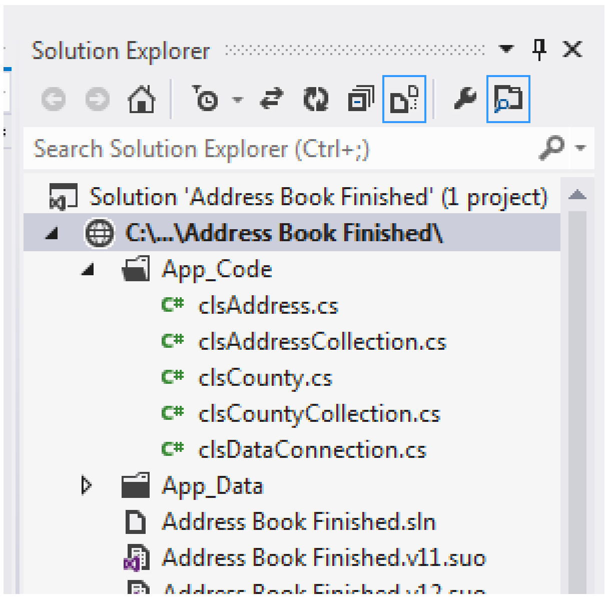
Many of the documents we are going to use act partly as a road map for the code and also as a tool to help us write our code.

In this case we are looking at the class diagram as a road map.

Like any road map we need to understand the different symbols if we are to fully appreciate what they are telling us about the code.

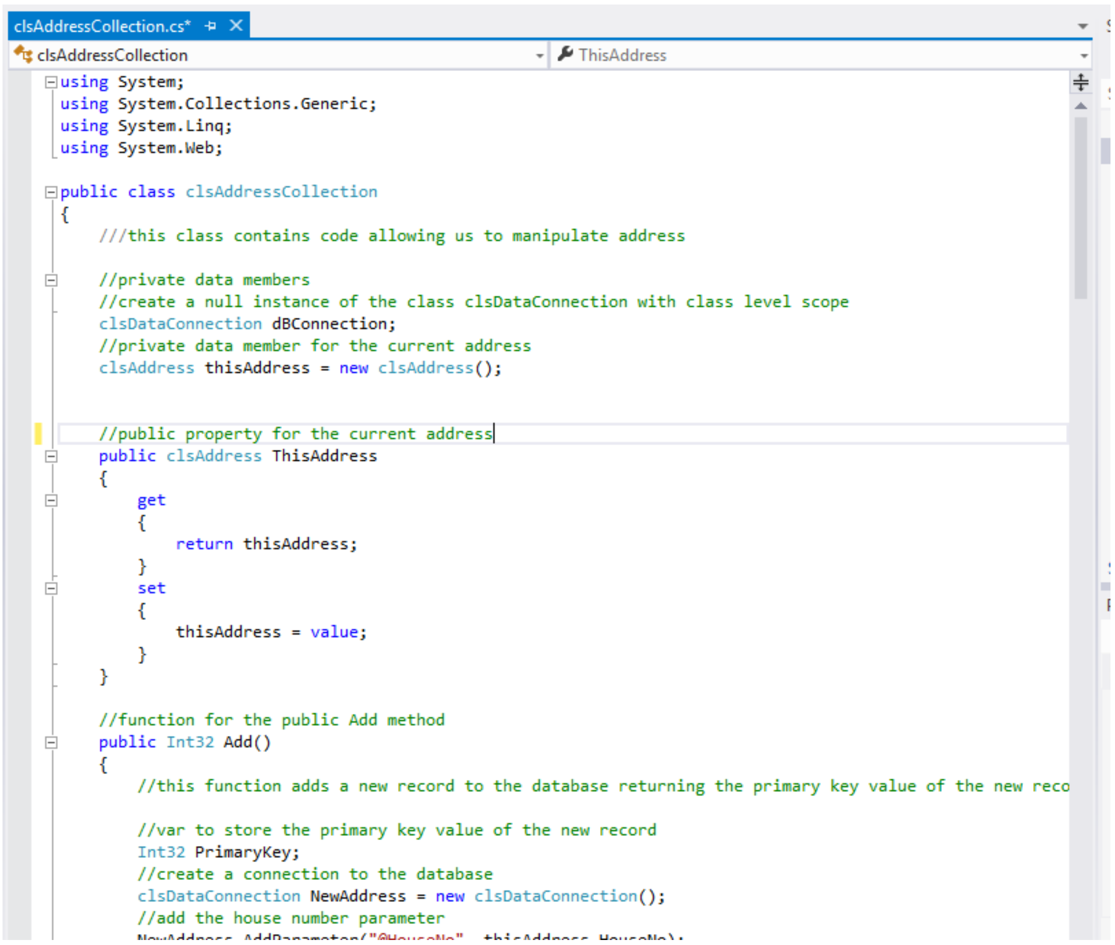
If we know what we are looking at this is very helpful as an overview of how the program is structured.

Without documents such as this we are left with this view of the system.



```
Solution 'Address Book Finished' (1 project)
◢ C:\...\Address Book Finished\
    ◢ App_Code
        clsAddress.cs
        clsAddressCollection.cs
        clsCounty.cs
        clsCountyCollection.cs
        clsDataConnection.cs
    ▷ App_Data
      Address Book Finished.sln
      Address Book Finished.v11.suo
      Address Book Finished.v12.suo
```

Or we are forced to read the code



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

public class clsAddressCollection
{
    ///this class contains code allowing us to manipulate address

    //private data members
    //create a null instance of the class clsDataConnection with class level scope
    clsDataConnection dBConnection;
    //private data member for the current address
    clsAddress thisAddress = new clsAddress();


    //public property for the current address
    public clsAddress ThisAddress
    {
        get
        {
            return thisAddress;
        }
        set
        {
            thisAddress = value;
        }
    }

    //function for the public Add method
    public Int32 Add()
    {
        //this function adds a new record to the database returning the primary key value of the new reco

        //var to store the primary key value of the new record
        Int32 PrimaryKey;
        //create a connection to the database
        clsDataConnection NewAddress = new clsDataConnection();
        //add the house number parameter
        NewAddress.AddParameter("@HouseNo", thisAddress.HouseNo);
```

We would probably figure out what is going on eventually but without the big picture documentation it is much harder to understand how the different parts of the system are composed and also how they relate to each other.

This module should help you acquire a set of skills and strategies for taking an initial specification and turning it into a professional system with a full set of supporting documentation.

The first half of this term the will look at approaches to tackling the analysis and design of a system. At this stage you will create the following artefacts…

- System specification
- Ethical review
- Event tables
- Use case diagrams
- Use case descriptions
- Test plans
- Smoke and mirrors prototype
- Class diagrams

After the Christmas break you will concentrate on the actual implementation of the system.

This will involve you writing code using a development technique called test driven development.

Remember Systems Development is Difficult

The truth is that even after years of experience there are still many things that need to be addressed that will be challenging to say the least.

For example

- Getting inside the head of the client (so that they get the product they want)
- Coming to terms with the various tools you need to use (e.g. Visual Studio, Enterprise Architect etc.)
- Working with team members (They will drive you mad!)
- Creating the documentation and understanding the notation
- Worst of all … writing the code!

Simply diving into a system is a really bad idea. Doing this means that we end up having to process too much information and lack a sufficiently thorough understanding of the detail involved in the problem at hand.

Question – How do you eat an elephant?

(Change the colour of the text below to reveal the answer.)

Answer - < >

A silly point but true nonetheless.

If we break a problem down into its individual components then we are more likely to "digest" a small part of it rather than attempting to consume the large whole.

The Project Bank

The default setting for the assessment for this module is called the Project Bank. What I mean is that unless you tell me otherwise we will all be working on the same system.

The Project Bank is intended to be a Customer Relationship Management (CRM) system.

> *Customer relationship management is a system for managing a company's interactions with current and future customers. It often involves using technology to organize, automate and synchronize sales, marketing, customer service, and technical support.*
>
> *(Wikipedea)*

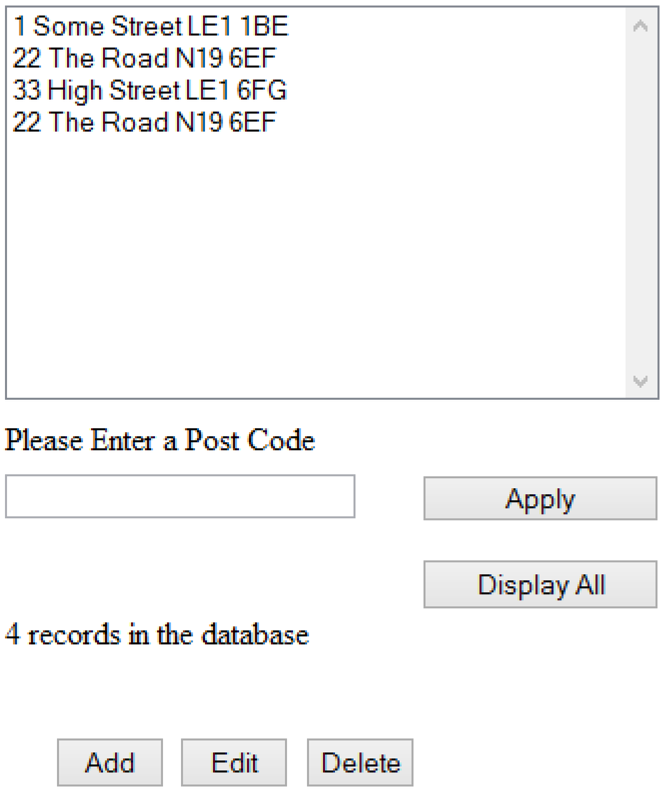
This module is very much going to continue the work that you did on IMAT1604 Visual Web Development.

The main thrust of this module is not specifically programming it is about developing a project and utilising a range of tools to create that project.

Programming skills will be one of those tools.

During the first year module we have spent a great deal of time looking at the address book application and looking at the code concepts that make it work.

The application on the surface looks like a fairly simple system.



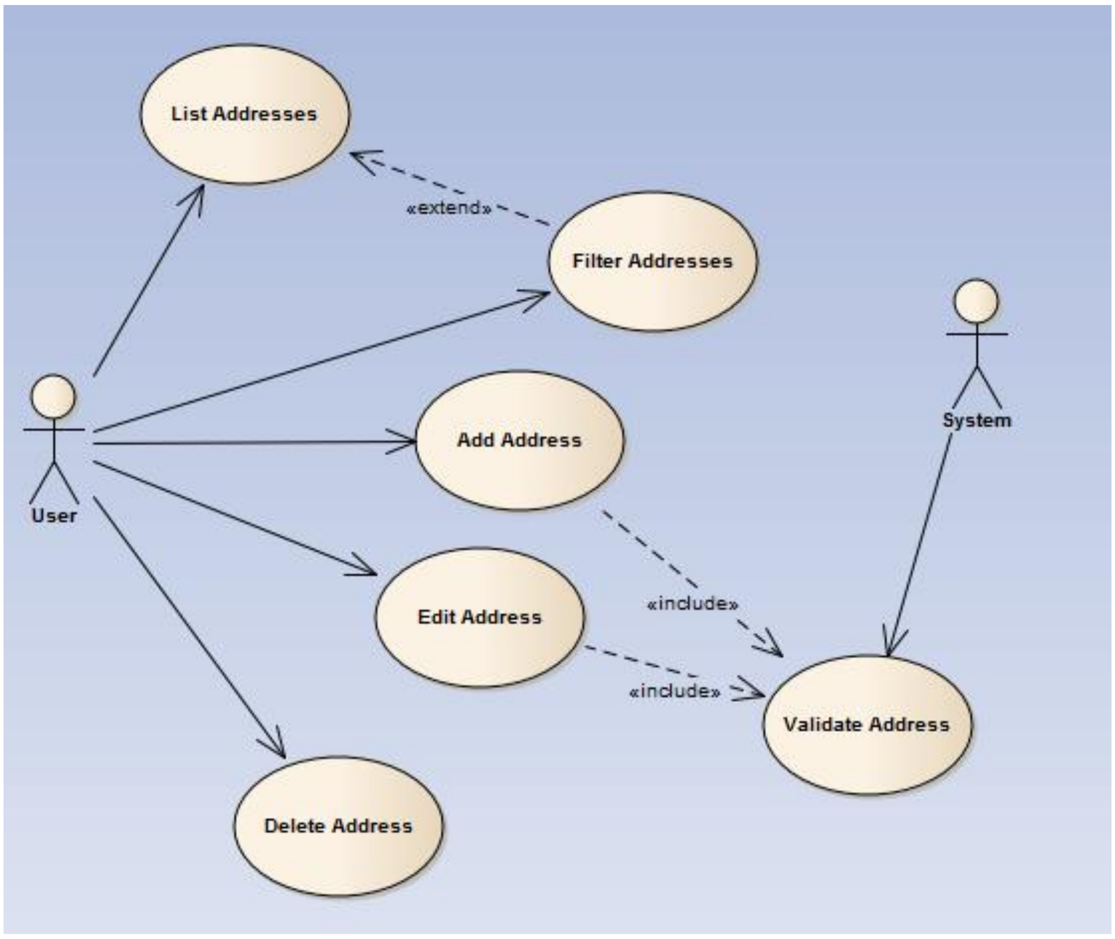
1 Some Street LE1 1BE
22 The Road N19 6EF
33 High Street LE1 6FG
22 The Road N19 6EF

Please Enter a Post Code

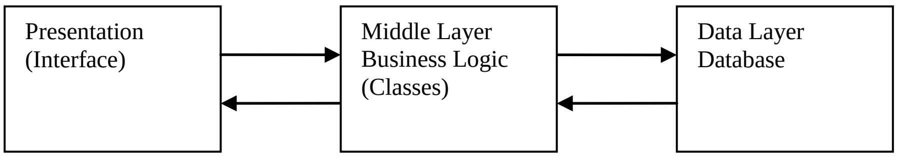
4 records in the database

We have the options to perform the following tasks

- Add an address
- Edit an address
- Validate an address
- Delete an address
- List addresses
- Filter the list of address



These tasks may have been quite simple if we had not been trying to strictly follow a three layered architecture.



By splitting the system like this it creates a huge amount of complexity under the surface of this seemingly simple interface.

To be honest creating an application as simple as this rigorously following a three layered architecture could be seen as overkill.

It did however allow us to explore within a simple set of business rules the essential structure of the three layered architecture.

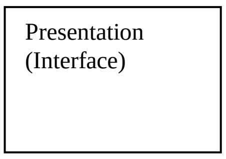
Lie: Applications must follow the three layered architecture

This is simply not true.

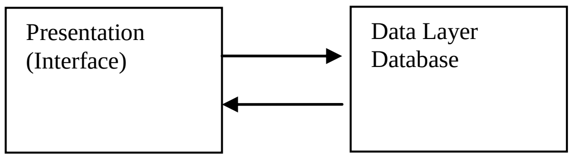
The three layered architecture is only one way of slicing the application cake.

If you had a small business or a sole trader wanting a web site building a full blown three layer application would certainly be too much.

You might simply create a static web page with no middle layer or data layer.



Or if the site was in need of persistent data we might have no middle layer and place all of the application code inside the presentation layer.



These two architectures would remove all of the complexity in the middle layer.

The advantage being that an application like this would be relatively quick and easy to put together.

However there are two problems with this design

1. With all of the code in the presentation layer it becomes difficult to share that code around the system or other systems we may create at a later date
2. With the presentation layer “knowing” so much about the database any changes to the database will inevitably mean making changes to the presentation layer

Not such a big deal on a small system but certainly a problem when the system gets bigger or we want to recycle any code across this or other applications.

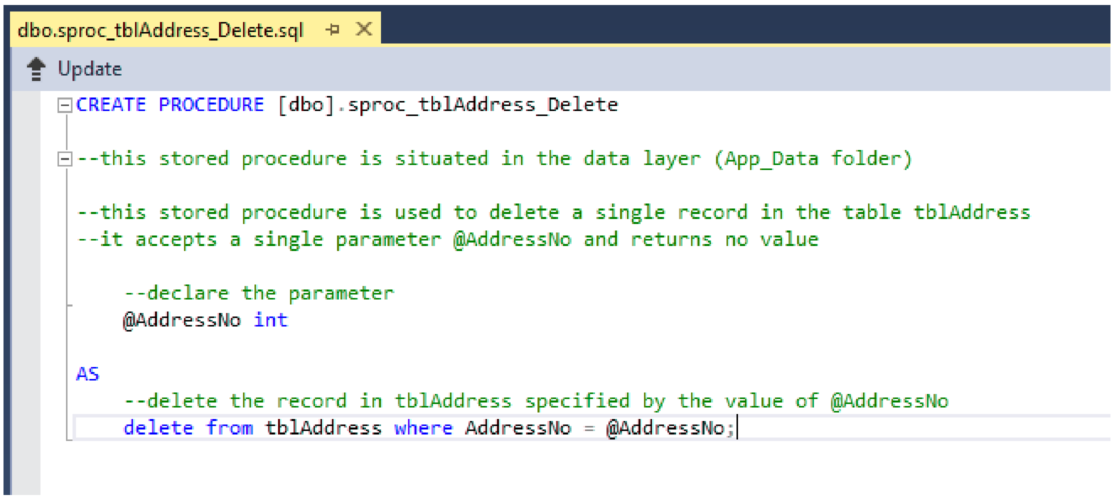
Thick and Thin Layers

When talking about the layers in a system you may hear the terms “thick and thin”.

What this means is that a layer might be present but it doesn’t do very much.

For IMAT1604 we used the Data Connection class to pass parameters and trigger stored procedures in the data layer.

For example we might have a stored procedure called sproc_tblAddress_Delete



```
CREATE PROCEDURE [dbo].sproc_tblAddress_Delete

--this stored procedure is situated in the data layer (App_Data folder)

--this stored procedure is used to delete a single record in the table tblAddress
--it accepts a single parameter @AddressNo and returns no value

    --declare the parameter
    @AddressNo int

AS
    --delete the record in tblAddress specified by the value of @AddressNo
    delete from tblAddress where AddressNo = @AddressNo;
```

The stored procedure accepts a single parameter @AddressNo of integer (Int) data type.

The SQL in uses this parameter to delete any records in tblAddress where the AddressNo column matches the value of this parameter.

To make use of this parameter in the C# code we would do the following.

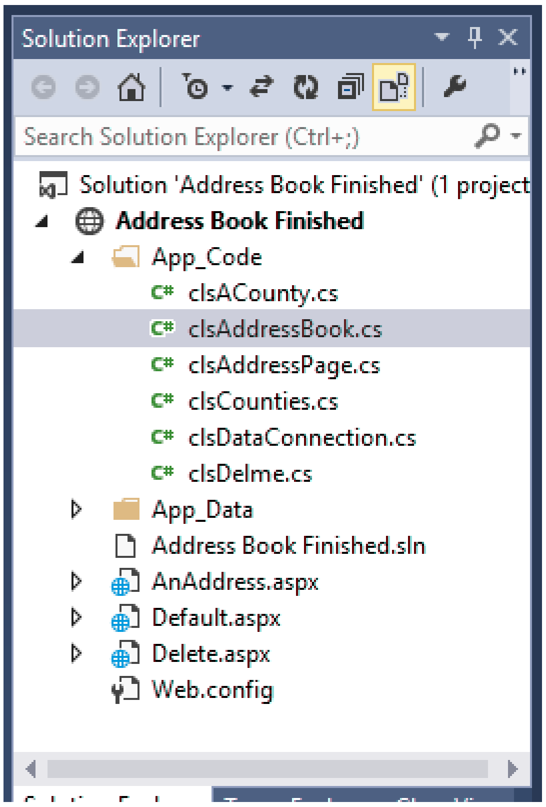
```
//initialise the DBConnection
dBConnection = new clsDataConnection();
//add the parameter data used by the stored procedure
dBConnection.AddParameter("@AddressNo", AddressNo);
//execute the stored procedure to delete the address
dBConnection.Execute("sproc_tblAddress_Delete");
```

Where the variable AddressNo would be assigned some suitable data identifying which row we wanted to delete.

So what do we mean by thick or thin?

In the address book application we have a fairly thick middle layer.

In the middle layer we have the following class definitions…

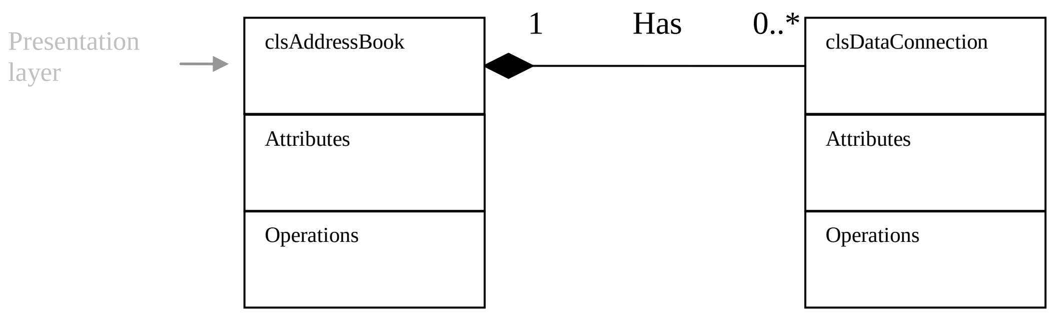


Solution 'Address Book Finished' (1 project
Address Book Finished
App_Code
clsACounty.cs
clsAddressBook.cs
clsAddressPage.cs
clsCounties.cs
clsDataConnection.cs
clsDelme.cs
App_Data
Address Book Finished.sln
AnAddress.aspx
Default.aspx
Delete.aspx
Web.config

The thing with this design is that clsDataConnection is never linked directly to the presentation layer.

If the presentation layer is to delete a record it must create an instance of clsAddressBook which in turn creates an instance of clsDataConnection.

In a class diagram this would be documented like so (Ignore the greyed out section as this is just to indicate at what point the presentation layer fits into the picture!)…



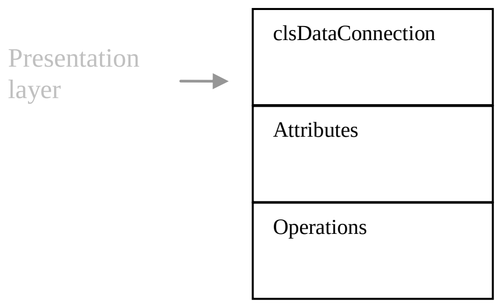
We will re-visit class diagrams in more detail later but this will serve as an introduction.

The black diamond ◆ indicates that the relationship is a "composition". This means that if we want to use clsDataConnection we must do so via the class clsAddressBook. We cannot create an instance of clsDataConnection on its own.

We could however ditch all of this complexity and not use other classes to manage the data connection for us.

In this case we could link the data connection straight to the presentation layer.

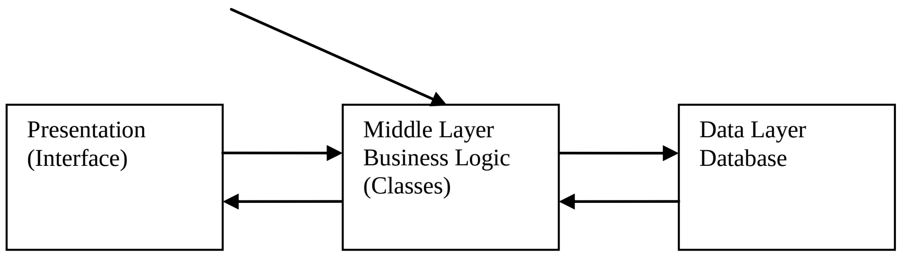
Like so…



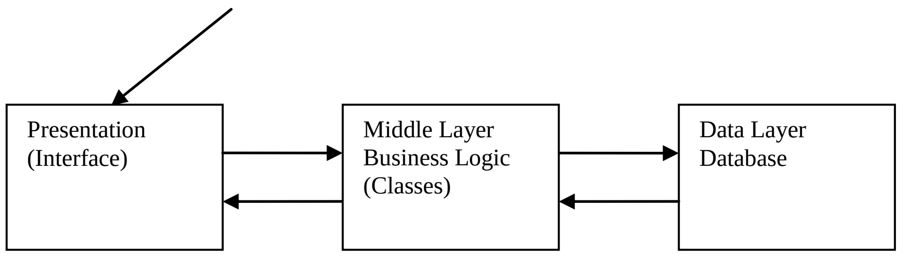
From the point of view of the three layered architecture the code…

```
//initialise the DBConnection
dBConnection = new clsDataConnection();
//add the parameter data used by the stored procedure
dBConnection.AddParameter("@AddressNo", AddressNo);
//execute the stored procedure to delete the address
dBConnection.Execute("sproc_tblAddress_Delete");
```

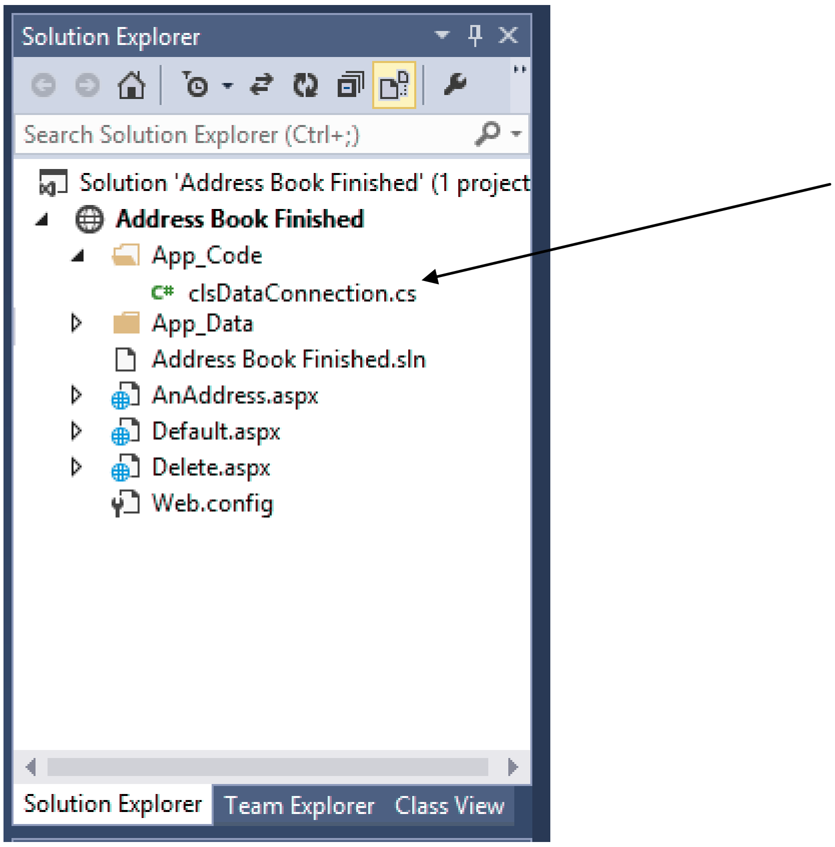
Which would normally go here…



Would go here…



It would be called a “thin” middle layer because the middle layer would contain a single class clsDataConnection



Thick and thin may also be terms applied to the other layers to indicate their complexity.

Which architecture should you use?

Well that is up to you to decide based on the nature of the project.

Facebook constructed as a simple static web page really wouldn’t be a good idea.

A static web site for a small business constructed as a sophisticated three layered architecture would be a sledge hammer to crack a walnut.

Although we need to make intelligent choices about the architecture we apply to our business problems there are some points that need to be considered when building our systems.

Design for Code Re-use

So far in our C sharp applications we have an address book.

The address book implements the features listed above.

Now is the point in time when we want to create other applications more interesting than an address book.

Let’s say we want now to create a book shop.

What overlap of functions will there be between the two applications?

Both applications will require the use of some sort of database connectivity.

Both applications will need to store details of people and possibly addresses.

Both applications my require search functionality.

Validation will be required for the data across the two applications.

If there is shared functionality between the two applications then how do we approach this?

Rebuild everything from scratch?

This is the standard option used by most students. The down side to this approach is that it is a massive waste of time and effort.

If you have created a wonderful bit of functionality then it makes no sense to throw it away and start from scratch.

Copy and Paste the Code?

This is another standard approach for students. When you create some code (or you use somebody else’s code) you copy and paste the code from the original application into the new application.

On the up side at least you are not creating all of the code from scratch!

There are a number of down sides to this approach.

Maintenance

How many copies will you need to make before modifying the code becomes a nightmare?

Code changes over time. Bugs are located and fixed. Enhancements and new features are added.

If you have two copies of the same code then any changes to the code must be made twice. If you have more copies of the same code then this problem multiplies.

Version control

In the real world not everybody has the same versions of particular software.

For example are we all using the same version of MS Windows on our PC?

If there is a problem with version 2 of some software we need to locate the state of the code when that version was released.

If we are to do that then we need to be pretty systematic about how we store our code so that we know we are correcting the right code along with understanding the impact it may have on other versions of the same software.

Create a Class Library?

In the end this is the smart option.

You create a library of code in a central location. The code is then shared across all of the applications we are creating. This code is designed in such a way that a specific bit of functionality exist only once in the library.

Designing Code for Ease of Testing

One important feature that we will touch on here is that of testing.

Testing is an essentially but deadly dull aspect of system's development.

Wouldn't it be nice if there were a way of running our tests automatically?

Well the answer is that there is!

There is another type of project that we may create using what is called a test framework.

Test frameworks allow us to create code that tests our work as we go.

We shall look at an approach to testing called Test Driven Development (TDD).

Traditionally testing of a system has been something of an afterthought.

You write a feature then write tests to test that feature.

In this case we will look to identify a feature, write the test code for that feature, see the feature fail the test then implement the code that makes the feature work.

This will be handled by a test framework.