# Design

Add more class diagrams showing relations etc. not just inheritance.

## **Why I chose to do an Object-Oriented Project**

The greatest reason I had for choosing object oriented programming was due to the ability to create user defined objects to be used in my program. After I decided on my project idea I sat down and thought about some of the basic functionality it would need to do and the easiest way to implement it. I had already had some experience with OOP and after looking at the basics of an accounting system I noticed that there were lots of clear objects which could be created. The ability to create your own object set properties and give functionality to the objects really stood out as a necessity to me in designing my project.

A more trivial reason for my decision to go with OOP was just for organisation of the code in visual studio. OOP allows for your code to be split up over multiple segments which allows for easy navigation and organisation of your program.

Another big selling point to use polymorphism is inheritance, having the ability to create sub/super classes of your use defined objects is incredibly useful to reduce the repetition of code and functionality, which in turn reduces the likelihood of errors and usually makes them easier to fix if they effect one part of a class hierarchy rather than finding an error in a part of code in an object which is in other objects and then having to individually fix them all.

**System Design**

The base user interface will be created by Naked Objects Framework, this includes Action bars, menu buttons etc. However I will make a few changes to the base design such as changing object colours to help provide standardisation between objects, this will help the user distinguish between types when multiple objects are being displayed on the same page. Below I will explain the main components of my project and what I want them to achieve in terms of functionality, usability, display and restrictions. Actions on instances of objects can be accessed via the action tab at the top of the UI which is on all instances of objects. The parameters for specific objects are explained in the following part of the design section, here I will be discussing more about there layout and integration into the user interface and methods they may have.

**Login / Authentication**

The login system will be done using Auth0. Auth0 is a third-party login / identity authenticating software which can be integrated into a program which has access to the internet. Auth0 will allow for people to login into my project using a google account, this will then provide me with the data I need for auditing. I feel that using Auth0 is the right direction as it will reduce the time I need to spend on the login side of my system which isn’t fundamental to the overall functionality which I wish to achieve from this project, however it is needed for Auditing capabilities.

**Home Page**

The Home Page will be the first page displayed to the user after they have Logged in. It won’t have any specific functionality, it will just be used to separate the two sides of the project, that being the double entry bookkeeping side and the Auditing / Administration side.

**Accounts**

*Home Page => Main Menu => Accounts*

Accounts will be accessible from the Main Menu via an Accounts option which should contain an “All accounts” option or by using the “Find Account by Name” option. Along with transactions; Accounts are the base of a bookkeeping system as they are effectively the conduits for which money is spent and earned through. There are no methods / actions I need on accounts as they are primarily used by other aspects of the project as they are mainly used for holding data. The Accounts Menu will also need an action for creating an Account. Accounts should contain a name, a type of account (capital, assets, liabilities), balance (based on transactions) and a list of credit and debit transactions involving that account.

**Transactions**

*Home Page => Main Menu => Transactions*

The main menu should contain a Transactions option which will contain an “All Transactions” option which will display all transactions in date order, displaying the newest at the top. A create new transaction option should also be available, this should implement both a drag and drop and auto fill features for filling out the Accounts field. There should also be a find Transaction by date option. The Transactions themselves should contain an action called “Show History” which is part of the auditing system. This action should show a list of previous versions of that transaction (if it has any previous versions). This is useful because say a Transaction has been updated and you are looking at it in the Audit Record menu you will be able to click that transaction to go to and then perform the “show history” action to see the previous versions. Transactions should show the date of transaction the debit and credit accounts, the amount and the name given to it by the user.

**Balance Sheet**

*Home Page => Main Menu => Balance Sheets*

Balance Sheets will contain a method to view all balance sheets and a method to create new balance sheets. The balance sheet will evaluate the transactions and accounts to assert that all movements of money have been accounted for, this is done by checking that the sum of assets and capital minus liabilities equals zero. Instances of balance sheets should display all accounts of type asset followed by their balance total and then the same for Liability accounts and Capital accounts. Right at the end there should be the total for Assets Minus Liabilities and Capital and then the balance for the whole sheet which should always equal 0.

**Profit and Loss**

*Home Page => Main Menu => Profit and Loss*

Profit and Loss will contain a method to view all profit and loss statements and a method for creating a new profit and loss statement. The profit and loss statement will need calculate the profit made from sales between two dates (entered by the user when created). It will do this by adding up the price the stock was sold at and subtracting the price the stock was bought at by the company. The information for these values on individual items will be held by a sales object which will act similarly to transactions (explained further on in this section). No actions should be needed on Profit and Loss statements as they are basically reports. Profit and Loss statements should show the start and end date for the statement and the gross profit over that time.

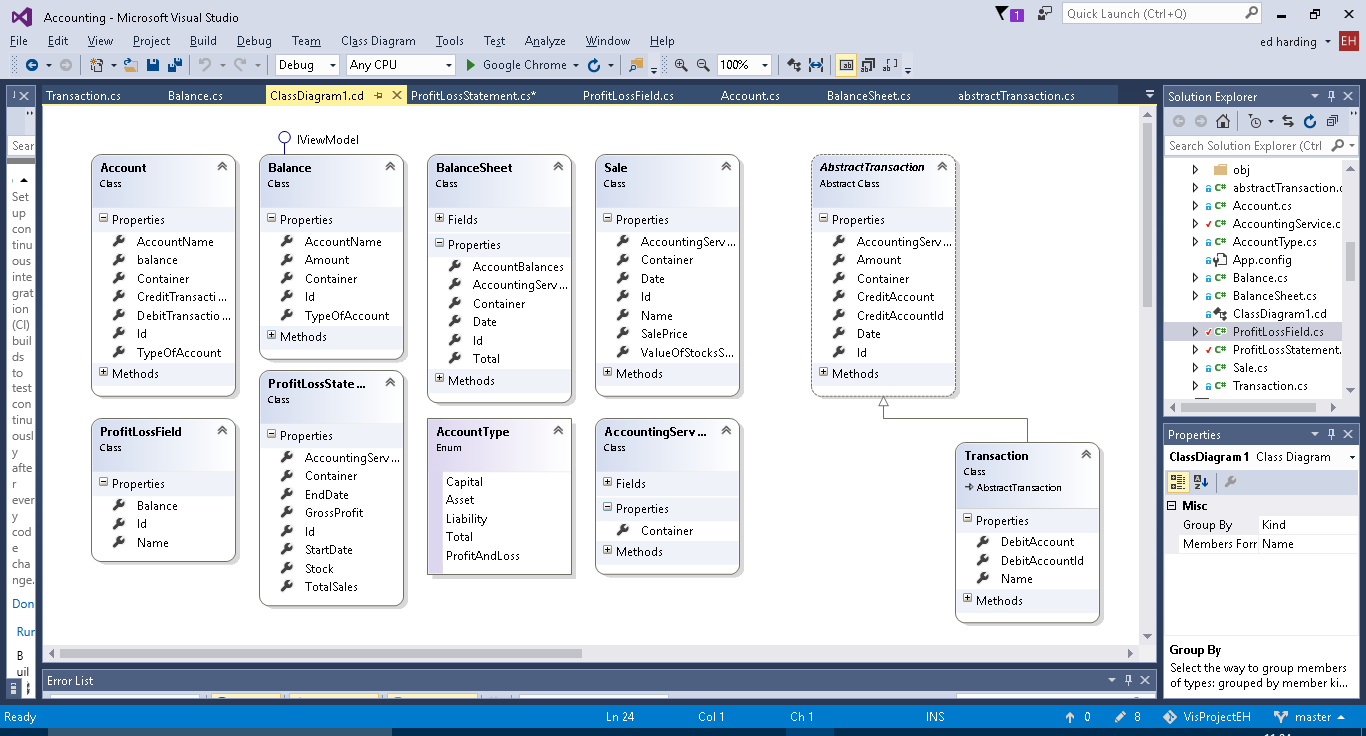
**Auditing**

*Home Page => Audit Menu*

As mentioned before the Auditing system will need to have user login details to work as the point in auditing is that you know the what happened, who did it and when it happened (the action, the user and the date time). Unlike the other parts of the project I want Auditing to have its own menu accessible from the Home Page and not be a part of the Main Menu. This is because Auditing is more of an administration section of my project rather than actual functionality to do with double entry book keeping. Inside the Audit Menu I will create Audit ‘reports’ such as “List all Audit Records” and then some specified ones such as “List all Audit Records for Persisted Objects”, and more. This is just for usability purposes so that the user can access specific Audit Records easier than having to search through all of them.

## **Class Diagram and Objects Used**

Overall Class Diagram.

Account

The account class is – as it is named – meant to represent an account in a running business. Every business has a list of different accounts which are used to record the movement of money and value of objects owned by the company. For example, a small retail company may have an account for the current cash held by them in store.

The account contains a collection of transactions and a calculated balance from these transactions. As this balance is used in the balance sheets and balance sheets can be made for past dates a function to calculate the balance at a specific date is needed. For both of the above a function to calculate a total balance is needed therefore a function can be made to do this to reduce repetition.

Balance

The Balance class doesn’t really have a business application behind it, it was just necessary to have for functionality purposes to be used as to retrieve and format the account data into the balance sheet.

Explain why I view model used.

Balance Sheet

As it is named the balance sheet object is made to create a balance sheet and calculate the balances of all the accounts and make sure that the capital equals the company assets minus liabilities.

A title builder method is implemented this is so that an apt title can be created automatically for the user using the date for which the balance sheet is for. This allows for them to be ordered in the user interface by date for user convenience.

Abstract transaction

An Abstract super class for transactions. Which will have more sub-classes.

Contains an auto complete method for the account to be credited for user convenience.

Transaction

The most common type of transaction made in a double entry bookkeeping system. It debits and credits individual accounts as it should. As you can see from the class diagram it is a sub class of Abstract Transaction.

Transaction also contains a title builder which creates a title consisting of the inputted transaction name and the date of which the transaction occurred, again this is just for user convenience and for data recording purposes. As Transaction is a sub class of abstract transaction it auto inherits the auto complete method for the account to be credited. However, it also contains an additional auto complete for the account to be debited (works the same as the credit one).

Account Type

The accounts in a double entry book keeping system fall into three categories capital, assets and liabilities. I need a way to represent them, so that total balances of each category can be calculated so that the balance can be checked as already mentioned in the Balance Sheet. The easiest way to do this was to create an enum.

Accounting Services

Accounting services doesn’t really represent anything specific in business terms it just adds functions which are useful to multiple different objects and to the user in the user interface, such as being able to add new instances of objects like Account and transactions. It also contains methods for the user such as listing all the accounts, transactions, balance sheets etc.

**Sale**

Is used in the profit and loss side of the program. Represents a sale of goods to a customer, and records the profit or loss made from that sale.

Sale also has a Title builder method which like transaction creates a title from the date of the sale and the name given to the sale.

**Sales Account**

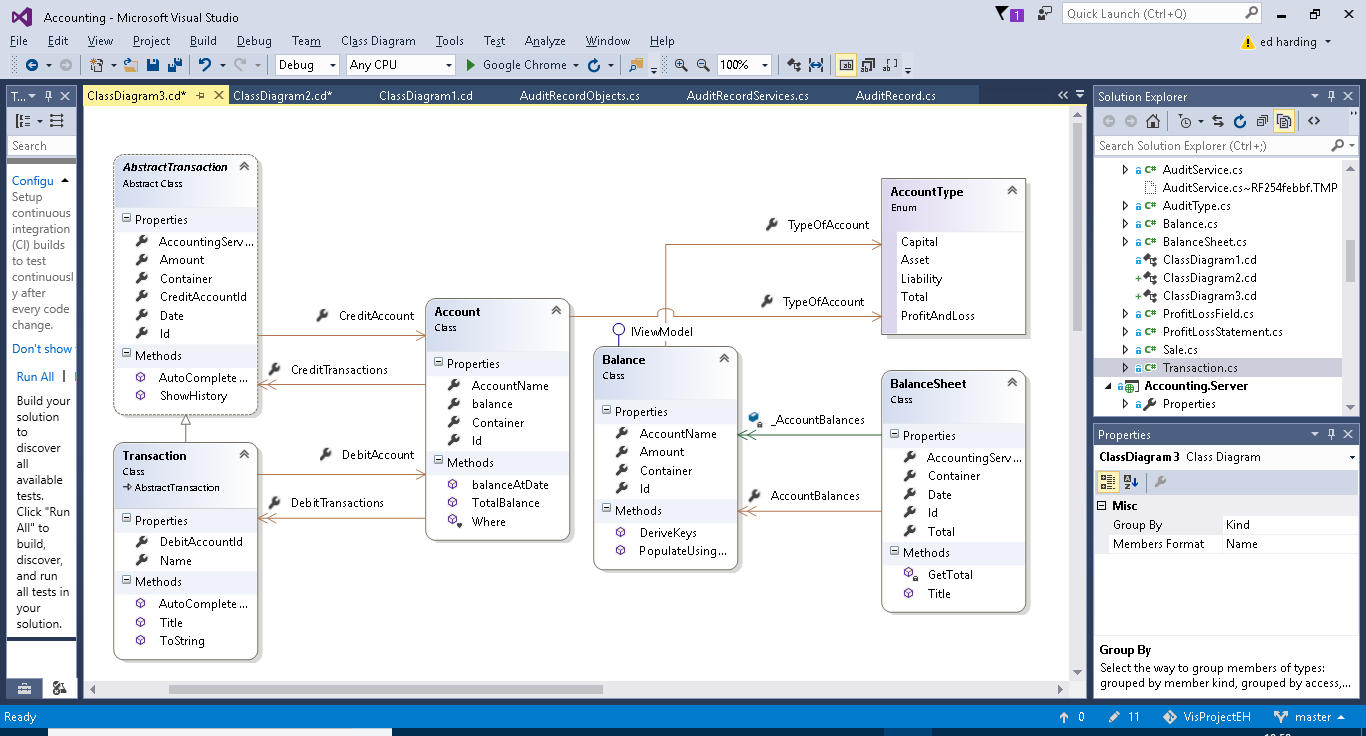
Is used to create the fields used in the profit and loss statement, those being the total value of the stock and the total value for sales made.

Needs renaming to ProfitLossField.

**Profit Loss Statement**

As the name suggests this class creates a profit and loss statement for a given time frame. It does this by getting the Sales (from Sale) made in that time frame and the value of the stock (at the latest point of the time frame) and then calculating the amount of stock sold and for what total price it was all sold at.

**Additional Class Diagrams**

Below is the class diagram which displays the object associations for the main part of the prgram(that being the transaction and balance sheet functionality) the objects are described in the previous section.

Below is the class diagram for the classes related to the profit and loss statement functionality of the program.

