C# Practice Programs

Demo 1:

You will be creating a Banking program. The bank program will have an account, and the account will be able to withdraw and deposit money. It will also be able to print its balance and other details to the terminal.

When the program runs, an `Account` object is created, and money is deposited and withdrawn into the account.

To get this started we will build the code for the Account class, and give it the knowledge and methods it needs to be able to withdraw, deposit and print its balance.

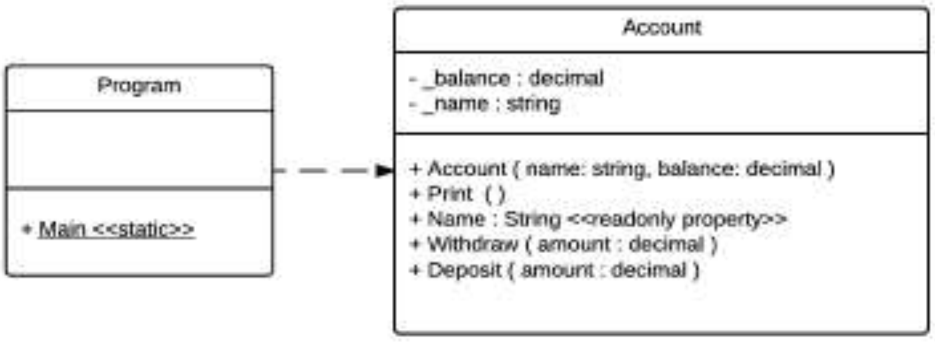
Before we can start writing the code we need to understand what we are trying to build (analyse the requirements) and then come up with a plan to get something working quickly.

For this program we will be modeling a standard **Bank Account**. Think about the kinds of things you would associate with a Bank Account, and these can be used to design an `Account` class. What things will an Account need to **know** and what will it need to be able to **do**?

Ok, hopefully you had some good ideas. The way I see this, we are going to need a **Program** which creates **Account** objects. The Program will then perform some operations on the Account (such as withdraw and deposit), and also ask the account to print its balance.

When building software, you want to work in small steps. So rather than trying to get *all* of this working in one go let's focus on getting part of the account working.

Here is the UML Class diagram that shows what we need to build in this iteration (another name for step):



Program
+ Main <<static>>
Account
- _balance : decimal
- _name : string
+ Account (name: string, balance: decimal)
+ Print ( )
+ Name : String <<readonly property>>
+ Withdraw (amount : decimal)
+ Deposit (amount : decimal)

Figure: Iteration 1, showing the Program and Account classes

1. In the project, create a new file called **Account.cs**.

   We will use this file to contain all of the code for the Account class. For anything beyond a small program, it is best to separate the code into multiple files.

2. Create a new `Account` class.

```
using System;

public class Account
{

}
```

3. Within the account class block, add the one `private decimal` field for the `_balance`.

```
  private decimal _balance;
```

   Reminder: a code block is the lines of code between a pair of braces: `{}`.

4. Have a go at adding the `private string` field `_name` yourself.

   Let's have a go at compiling and running this program, it won't do much right now, but it's a good time to make sure we have no syntax errors.

Key TakeAways

- How classes are used to define objects.
- How methods, fields, and properties all work together when you create a class
- How fields give knowledge to each object created from the class
- How methods give capabilities to each object created from the class