CSharp Syntax

```
using System;
using System.Security.Cryptography;

namespace ConsoleApp1{
    internal class Program{
        static void Main(string[] args){
            List<Client> clients = new List<Client>();

            Console.WriteLine("Name:");
            var name = Console.ReadLine();

            Console.WriteLine("Code:");
            var code = Console.ReadLine();

            Console.WriteLine("Credit Limit:");
            var cLimit = Console.ReadLine();
            var cLimitObj = double.Parse(cLimit ?? "0");

            var myClient = new Client();

            myClient.Name = name;
            myClient.Code = code;
            myClient.CreditLimit = cLimitObj;

            clients.Add(myClient);

        }
// Null coelesant operator ??
// if the left of the operator is not null return that value;
    }
}
```

You can skip over getter and setter declaration like this!

```
using System.Dynamic;
using System.Xml.Linq;

public class Client{
    public string? Name;
    public string? Code{get; set;} // automatically creates getters and setters

    public double CreditLimit{get; set;}

    // Default
    public Client(){

    }

    // Theres a better way then creating a parameterized constructor
    public Client(string? name, string? code, double creditLimit){
        Name = name;
        Code = code;
        CreditLimit = creditLimit;
    }
```

```
}
```

String interpolation is much like python within c++

```
public override string ToString()
{
return $"({Name} {Code} - ${CreditLimit})";
 // if you would like it to remain a literal replace $ with @
}
```

For Each Loop

```
foreach(var client in clients){
        Console.WriteLine(client.Name);
        Console.WriteLine(client.Code);
        Console.WriteLine(client.CreditLimit);

        Console.WriteLine(client.ToString());
}
```

Custom Get/Set

```
    get{
        return "";

    }
    set{
        Code = "";
    }
```