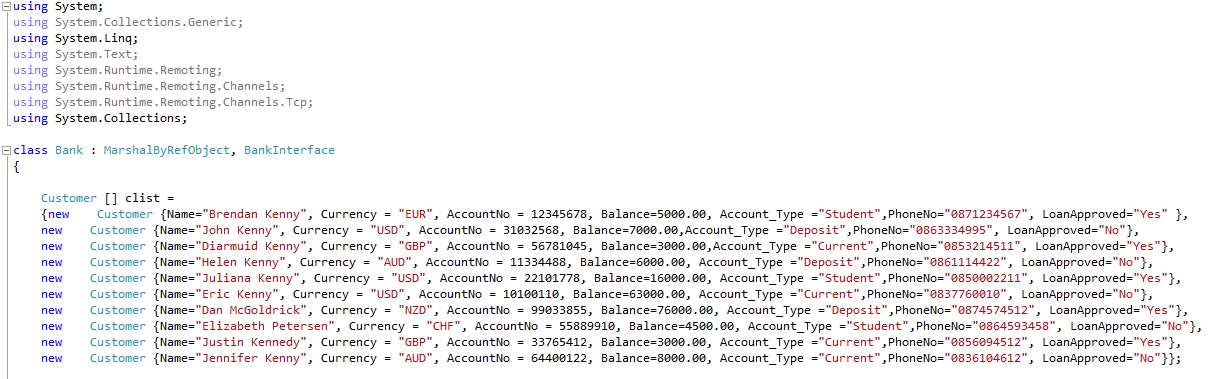
***Outline of C# Windows Application***

This C# application uses a graphical user interface to display details of customer accounts which can be Current, Deposit or Student. It will display details such as the customer name, account number, currency, account balance, account type, loan approval status and phone number.

The user will be able to search for accounts details for a specific customer, scroll through the records via Previous and Next Record buttons, display the First Record and go back to the start of the array via the First Record button and Clear all fields via the Clear button. The Exit button will allow the user to exit the application.

The GUI features GUI components such as radio buttons to display the Currency type of the account being displayed and a Checkbox to display whether or not the customer in the record being displayed has been approved for a loan.

This collection was created and stored on a Server and the GUI acts as a client.



The above Customer objects in the ***clist array*** are instances of the ***Customer*** class that inherits from the ***Person*** Class. This demonstrates ***inheritance***. The class Customer extends the Person class, in other words the Customer ***is a*** Person and therefore ***inherits*** the properties of the Person class. The Bank is linked to a collection of Customers ***(aggregation).***

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

class Person

**{**

**public** string Name **{** get**;** set**;** **}**

**public** string PhoneNo **{** get**;** set**;** **}**

**}**

class Customer **:** Person

**{**

**public** int AccountNo **{** get**;** set**;}**

**public** double Balance **{** get**;** set**;** **}**

**public** string Currency **{** get**;** set**;** **}**

**public** string Account\_Type **{** get**;** set**;** **}**

**public** string LoanApproved **{** get**;** set**;** **}**

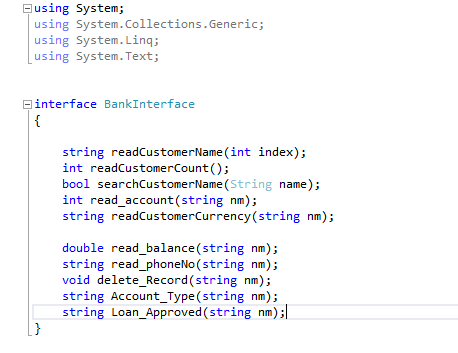
**}**

Please see the below screenshots for the finished graphical user interface design.





# ***Bank Interface***



**The above interface is common to both the client and server. The methods above refer to methods in the Bank.cs class.**

***Client Side***

In the above code in the BankGui class (the Client), a connection is established to the Server (using the Bank Interface) via dot.net remotting.

class Program

**{**

static void Main**(**string**[]** args**)**

**{**

BankServer**();**

**}**

static void BankServer**()**

**{**

Console**.**WriteLine**(**" Server started..."**);**

TcpChannel tcpChannel **=** **new** TcpChannel**(**9998**);**

ChannelServices**.**RegisterChannel**(**tcpChannel**);**

Type commonInterfaceType **=** Type**.**GetType**(**"Bank"**);**

RemotingConfiguration**.**RegisterWellKnownServiceType**(**commonInterfaceType**,**

"MyBank"**,** WellKnownObjectMode**.**Singleton**);**

System**.**Console**.**WriteLine**(**"Press ENTER to Exit"**);**

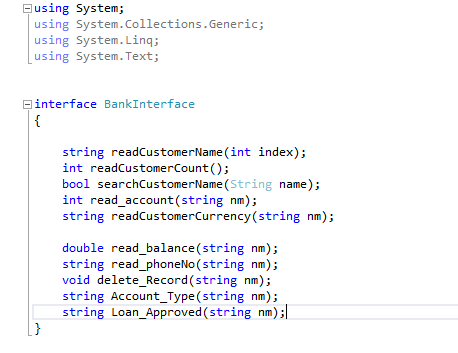
System**.**Console**.**ReadLine**();**

**}**

**}**

***Server Side Connection***

# ***Bank Interface***



public partial class BANKGUI **:** Form

**{**

private BankInterface b**;**

static int count**,** index**;**

The ***Search*** button in the BankGui class when clicked calls the ***searchCustomerName*** method in the ***Bank.cs*** class and referenced by the ***BankInterface*** class

private void Search\_Click**(**object sender**,** EventArgs e**)**

**{**

String name **=** t6**.**Text**;**

bool res **=** b**.**searchCustomerName**(**name**);**

**if** **(**res **==** **false)**

**{** MessageBox**.**Show**(**"Not Found"**);**

t6**.**Text **=** ""**;**

**}**

**else{**

t1**.**Text **=** "" **+** name**;**

t2**.**Text **=** "" **+** b**.**read\_account**(**name**);**

t3**.**Text **=** "" **+** b**.**readCustomerCurrency**(**name**);**

t4**.**Text **=** "" **+** b**.**read\_balance**(**name**);**

t5**.**Text **=** "" **+** b**.**read\_phoneNo**(**name**);**

t6**.**Text **=** ""**;**

**if** **(**b**.**Account\_Type**(**name**).**Equals**(**"Deposit"**)** **&&** **(**Deposit**.**Checked **==** **false))**

**{**

Deposit**.**PerformClick**();**

**}**

**else** **if** **(**b**.**Account\_Type**(**name**).**Equals**(**"Current"**))**

**{**

Current**.**PerformClick**();**

**}**

**else** **if** **(**b**.**Account\_Type**(**name**).**Equals**(**"Student"**))**

**{**

Student**.**PerformClick**();**

**}**

**if** **(**b**.**Loan\_Approved**(**name**).**Equals**(**"Yes"**))**

**{**

checkBox1**.**Checked **=** **!**checkBox1**.**Checked**;**

**}** **else** **{** checkBox1**.**Checked **=** **false;** **}**

**}**

**}**

# ***The searchCustomerName Method in Bank.cs***

public bool searchCustomerName**(**String nm**)**

**{**

var c\_list **=** from a in clist

where a**.**Name **==** nm

select a**;**

**if** **(**c\_list**.**Count**()** **>** 0**)** **return** **true;**

**else** **return** **false;**

**}**

The above query is a clear demonstration of ***LINQ***. Basically what is going on here is that the query is saying select the record from the array ***clist*** where a.Name is equal to the parameter being passed in String nm. So the query finds the record where the Name value of that array is equal to the String parameter being passed in (String nm).

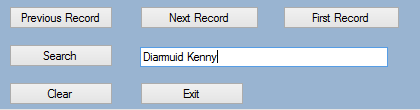
The second part states if the array in question (clist) is greater than zero, (in other words not empty) return true, else return false. The varible name var is an implicit polymorphic variable type and is possible to use here as C# is not a strongly typed language like Java for example. What that means is that variables that are denoted as var can be of any variable type, unlike the variable type String, which has to be a string for example.

Customer **[]** clist **=**

**{new** Customer **{**Name**=**"Brendan Kenny"**,** Currency **=** "EUR"**,** AccountNo **=** 12345678**,** Balance**=**5000.00**,** Account\_Type **=**"Student"**,**PhoneNo**=**"0871234567"**,** LoanApproved**=**"Yes" **},**

**new** Customer **{**Name**=**"John Kenny"**,** Currency **=** "USD"**,** AccountNo **=** 31032568**,** Balance**=**7000.00**,**Account\_Type **=**"Deposit"**,**PhoneNo**=**"0863334995"**,** LoanApproved**=**"No"**},**

# **The above is a sample of some records in the clist array.**



The user can now search for a customer by name using the Search button which implements the Search\_Click method that uses the searchCustomerName method in the Bank class on the server side by means of the BankInterface object and the dot.net remotting connection.

# ***Display Method***

**public** void display**(**int index**)** //Display Method, displays all relevant info to the GUI components - textboxs, radio buttons etc

**{**

String name **=** b**.**readCustomerName**(**index**);**

t1**.**Text **=** name**;**

t2**.**Text **=** "" **+** b**.**read\_account**(**name**);**

t3**.**Text **=** "" **+** b**.**readCustomerCurrency**(**name**);**

t4**.**Text **=** "" **+** b**.**read\_balance**(**name**);**

t5**.**Text **=** "" **+** b**.**read\_phoneNo**(**name**);**

**if** **(**b**.**Account\_Type**(**name**).**Equals**(**"Deposit"**))**

**{**

Deposit**.**PerformClick**();**

**}**

**else** **if** **(**b**.**Account\_Type**(**name**).**Equals**(**"Current"**))**

**{**

Current**.**PerformClick**();**

**}**

**else**

**{**

Student**.**PerformClick**();** Console**.**WriteLine**(**"Clicked={0}"**,** name**);**

Student**.**Checked **=** **true;**

**}**

**if** **(**b**.**Loan\_Approved**(**name**).**Equals**(**"Yes"**))**

**{**

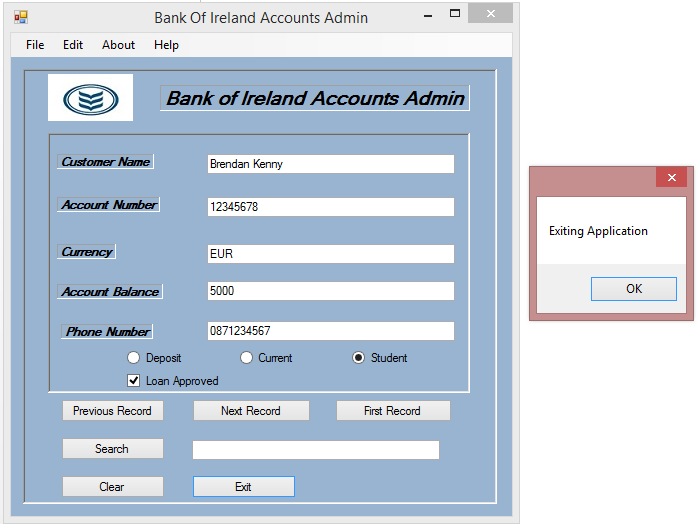
checkBox1**.**Checked **=** **!**checkBox1**.**Checked**;**

**}**

**else** **{** checkBox1**.**Checked **=** **false;** **}**

**}**

The above Display method in the BankGui class relevant methods on the server side using the Bank Interface, in the same way the Search\_Click method does.



# ***Exit Screen***

# ***Conclusion***

In the project as outlined above, my goal was to demonstrate the use of .dot net remotting, querying using LINQ and the use of GUI interface and GUI components to demonstrate this. The methods that use LINQ also demonstrate that C# is not a strongly typed language in the sense that Java and other languages are through the use of the word var to denote a variable, which is an implicit polymorphic type. I had intended to use a combo box but I did not figure out the correct way to implement this in the time given, so it was left out, I had some trouble using the radio buttons correctly but this issue was eventually addressed. I had intended to use a delete button also but did not figure out the correct way to implement this in the time given. However, I am glad that I was able to demonstrate and implement LINQ, .net remotting, aggregation and inheritance.