# Source Code

## Form.CS

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Configuration;

using System.Data.SqlClient;

namespace TTRCv1 // Name space of the project

{

public partial class Form1 : Form

{

SqlConnection connection; // The connection tells the rest of the ADO.NET code which database it is talking to

string connectionString;

public List<FullMember> clubList; // Takes the list of objects from the 'Full member' list for use in the database linking to the class diagram

public Form1()

{

InitializeComponent();

//connectionString = ConfigurationManager.ConnectionStrings["TrawdenTrampersRCLib.Properties.Settings.TTRCConnectionString"].ConnectionString; // String used to open a SQL server database

}

private void Form1\_Load(object sender, EventArgs e)

{

clubList = new List<FullMember>();

PopulateMembers(); // Populates 'Members' table in the database

PopulateAllMedicalIssues(); // Populates 'Medical issues' table in the database

PopulateAllAddresses(); // Populates 'Address' table in the database

}

private void PopulateMembers() // Void needed to populate 'Members' table

{

//string query =

// "SELECT a.Name FROM Members a " + // Query to join all tables to retrieve and display information in the membersLST list box

// "INNER JOIN AddressesMedicalIssuesMembers b ON a.Id = b.MembersId " +

// "Where b.MembersId = @MembersId";

//using (connection = new SqlConnection(connectionString)) // SQL database connection

//using (SqlCommand command = new SqlCommand(query, connection)) // connect and retrieve database information requested through queries

//using (SqlDataAdapter adapter = new SqlDataAdapter("SELECT \* FROM Members", connection)) // Sql query to take information from the members table when connected and requested

//{

// command.Parameters.AddWithValue("@MembersId", memberLST.SelectedValue); // Add memberId value to a new member in the member list

// DataTable MembersTable = new DataTable();

// adapter.Fill(MembersTable); // Fill 'Members' table with new mmeber information and Id value

// memberLST.DisplayMember = "Name"; // memberLST list to display member names

// memberLST.DisplayMember = "Id"; // memberLST to display member Id

// memberLST.DataSource = MembersTable; // memberLST to display all other member table information

//}

}

private void PopulateAllMedicalIssues() // Add data to the 'Medical Issues' table

{

//using (connection = new SqlConnection(connectionString))

//using (SqlDataAdapter adapter = new SqlDataAdapter("SELECT \* FROM MedicalIssues", connection)) // Sql query to take information from the medical issues table when connected and requested

//{

// DataTable MedicalIssuesTable = new DataTable();

// adapter.Fill(MedicalIssuesTable); // Fill table with medical conditions of members

// memberLST.DisplayMember = "Name"; // memberLST list to display member names

// memberLST.ValueMember = "Id"; // memberLST to display member Id

// memberLST.DataSource = MedicalIssuesTable; // memberLST to display all medical data about members

//}

}

private void PopulateAllAddresses() // Sql query to populate address table

{

//using (connection = new SqlConnection(connectionString))

//using (SqlDataAdapter adapter = new SqlDataAdapter("SELECT \* FROM Addresses", connection))

//{

// DataTable AddressesTable = new DataTable(); // link to address table

// adapter.Fill(AddressesTable); // Fill address table with member housing information

// memberLST.DisplayMember = "Name"; // memberLST list to display member names

// memberLST.ValueMember = "Id"; // memberLST to display member Id

// memberLST.DataSource = AddressesTable; // memberLST to display all member address information

//}

}

private void memberBTN\_Click(object sender, EventArgs e) // add member button

{

string query = "INSERT INTO Members VALUES (@newPerson)"; // query to add memeber information into the database

using (connection = new SqlConnection(connectionString))

using (SqlCommand command = new SqlCommand(query, connection))

{

connection.Open(); // Open the database connection

command.Parameters.AddWithValue("@MembersName", txtFname.Text); // Add members first name to the database

command.ExecuteScalar(); // execute sql command

}

PopulateMembers(); // Populate Members table

// In context with the 'Full member' class, all data must match to coincide with the database and the 'Full member' class for a successful application

// public FullMember(string f, string l, string d, string e, string g, int m, Address a, int u, string j, string c)

clubList.Add(new FullMember(txtFname.Text, txtLname.Text, txtDOB.Text, txtEmail.Text, txtGender.Text, txtMob.Text, txtAdd1.Text, txtUK.Text, txtJoin.Text, txtMed.Text));

}

private void searchBTN\_Click(object sender, EventArgs e) // Search button

{

memberLST.Items.Clear(); // Clear list then add items to list from the submit button through the full member.cs file

foreach (var p in clubList)

{

memberLST.Items.Add(p.getDetails()); // displays information taken from temporary stroage using RAM

}

}

private void memberLST\_SelectedIndexChanged(object sender, EventArgs e)

{

PopulateMembers(); // Populate members list in memberLST from the database tables

}

private void editBTN\_Click(object sender, EventArgs e) // Edit button

{

string query = "UPDATE Members SET Name = @MembersFname WHERE Id = @MembersId"; //Sql query for calling and editing member details

// string query = "UPDATE Members SET Name = @MembersLname WHERE Id = @MembersId"; possibly all required? \*\*

using (connection = new SqlConnection(connectionString))

using (SqlCommand command = new SqlCommand(query, connection))

{

connection.Open();

command.Parameters.AddWithValue("@MembersName", txtFname.Text); // Command to edit members details through searching for the member with their first name

command.Parameters.AddWithValue("MembersId", memberLST.SelectedValue); // Command to edit members details through searching for a member using their Id

command.ExecuteScalar(); // Execute command

}

PopulateMembers();

}

private void dataBTN\_Click(object sender, EventArgs e)

{

string query = "INSERT INTO AddressesMedicalIssuesMembers VALUES (@MembersId, @AddressesId, @MedicalIssuesId)";

using (connection = new SqlConnection(connectionString))

using (SqlCommand command = new SqlCommand(query, connection))

{

connection.Open();

command.Parameters.AddWithValue("@MembersId", memberLST.SelectedValue); // Command to add members details through searching for the member with their first name

command.Parameters.AddWithValue("@AddressesId", memberLST.SelectedValue); // Command to add members details through searching for a member using their Id

command.Parameters.AddWithValue("@MedicalIssuesId", memberLST.SelectedValue); // Command to add members medical details to the medical Issues table and display this in the member list

command.ExecuteScalar(); // Execute command

}

PopulateMembers();

}

private void resignBTN\_Click(object sender, EventArgs e)

{

{

string query = "DELETE FROM Members WHERE Id = (@idSelected)"; // delete selected members through ID number using a query command

if (MessageBox.Show(

"You are about to Resign " + memberLST.Text,

"Are you sure?",

MessageBoxButtons.YesNo,

MessageBoxIcon.Warning)

== DialogResult.Yes); // Message box upon selection to ask if they are sure if they wish to resign a member. If so then the end result is 'Yes' folowed by the member staus of 'resigned' in the database

memberLST.Items.Clear();

//{

// using (connection = new SqlConnection(connectionString))

// using (SqlCommand command = new SqlCommand(query, connection))

// {

// connection.Open();

// command.Parameters.AddWithValue("@idSelected", memberLST.SelectedValue); // Identify through member ID to retrieve and resign the seleced member.

// command.ExecuteScalar();

// }

// PopulateMembers(); // Confirm and refresh

//}

}

}

private void memberLST\_SelectedIndexChanged\_1(object sender, EventArgs e)

{

}

private void searchBTN\_Click\_1(object sender, EventArgs e) // Search button

{

int x = clubList.FindIndex(f => f.lName() == txtLast.Text); // Searches for member by thier last name

if (x == -1)

MessageBox.Show("cannot Find " + txtLast.Text); // Message box to display member doesnt exist

else

txtFname.Text = clubList[x].getDetails(); // if member is found then display member details in the text boxes listed in the code, corrasponding with their correct fields.

txtLname.Text = clubList[x].getDetails();

txtDOB.Text = clubList[x].getDetails();

txtGender.Text = clubList[x].getDetails();

txtAdd1.Text = clubList[x].getDetails();

txtAdd2.Text = clubList[x].getDetails();

txtTown.Text = clubList[x].getDetails();

txtPost.Text = clubList[x].getDetails();

txtMob.Text = clubList[x].getDetails();

txtland.Text = clubList[x].getDetails();

txtEmail.Text = clubList[x].getDetails();

txtJoin.Text = clubList[x].getDetails();

txtNOK.Text = clubList[x].getDetails();

txtMed.Text = clubList[x].getDetails();

txtUK.Text = clubList[x].getDetails();

}

private void totalLST\_SelectedIndexChanged(object sender, EventArgs e) // Error. Accidental click. Do NOT touch!

{

//memberLST.count(count + 1);

}

private void editBTN\_Click\_1(object sender, EventArgs e) // Edit Button

{

int x = clubList.FindIndex(f => f.lName() == txtLast.Text);

if (x == -1)

MessageBox.Show("cannot Find " + txtLast.Text); // Message box to show it cannot find requested member

else

{

clubList[x].setfName(txtFname.Text); // If member is found, then allow user to change thier details in the corrasponding text boxes

clubList[x].setlName(txtLname.Text);

clubList[x].setdob(txtDOB.Text);

clubList[x].setgender(txtGender.Text);

clubList[x].setaddress(txtAdd1.Text);

clubList[x].setaddress(txtAdd2.Text);

clubList[x].setmobileNumber(txtMob.Text);

clubList[x].setdateJoined(txtJoin.Text);

clubList[x].setmedicalCondition(txtMed.Text);

clubList[x].setukAthleticsNumber(txtUK.Text);

}

}

}

}

## FullMember.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace TTRCv1 // Name of project

{

public class FullMember // class name from class diagram

{

private string id; // private strings used to define class items

private string DOB;

private string date;

private string Em;

private string Fn;

private string Ln;

private string Gen;

private string med;

private string mobile;

private string yesno;

private string num;

public string addressId() // string for class item address

{

return (id);

}

public string dob() // string for class item date of birth for members date of birth

{

return (DOB);

}

public string dateJoined () //string for class item to identify when the member joined

{

return (date);

}

public string email () // string for class item email to identify mmebers email address

{

return (Em);

}

public string fName () // string for class item first name for members first name

{

return (Fn);

}

public string lName() // string for class item last name for members last name

{

return (Ln);

}

public string gender () // string for class item gender for members gender

{

return (Gen);

}

public string medicalCondition () // string for class item medical condition to identify any conditions a member may have

{

return (med);

}

public string mobileNumber() // string for class item mobile number to save the members mobile numbers

{

return (mobile);

}

public string resignMember() // string for class item resign member to resign a mmeber at any given date

{

return (yesno);

}

public string ukAthleticsNumber() // string for class item athletics number to store members athletics numbers

{

return (num);

}

public virtual Address Address // address type for class item address to link fullmember.cs to address.cs

{

get;

set;

}

public FullMember() // list of private string types to display in a list

{

Fn = "";

Ln = "";

Gen = "";

id = "";

DOB = "";

date = "";

Em = "";

med = "";

mobile = "";

}

public FullMember(string f, string l, string d, string e, string g, string m, string a, string u, string j, string c)

{

Fn = f; // string types to link FullMember.cs with the submit button code to ensure that the string types match to display in the List box

Ln = l;

DOB = d;

Em = e;

Gen = g;

med = c;

id = a;

num = u;

date = j;

mobile = m;

}

public string getAddress()

{

throw new System.NotImplementedException(); // get members address

}

public string getDetails() // Enables the Full member types to be correctly displayed in the list box, retrieving the information held in the RAM until the sessions ends

{

return (Fn + " " + Ln + " " + DOB + " " + Gen + " " + med + " " + id + " " + num + " " + date + " " + mobile + " ");

}

public void setfName (string f) // Set methods to replace name in correspondence with 'Edit' button.

{ // This allows the user to edit any information that needs to be updated for the member.

Fn = f;

}

public void setlName (string l)

{

Ln = l;

}

public void setaddress (string a)

{

id = a;

}

public void setdob (string d)

{

DOB = d;

}

public void setdateJoined (string j)

{

date = j;

}

public void setemail (string E)

{

Em = E;

}

public void setgender (string g)

{

Gen = g;

}

public void setmedicalCondition (string c)

{

med = c;

}

public void setmobileNumber (string m)

{

mobile = m;

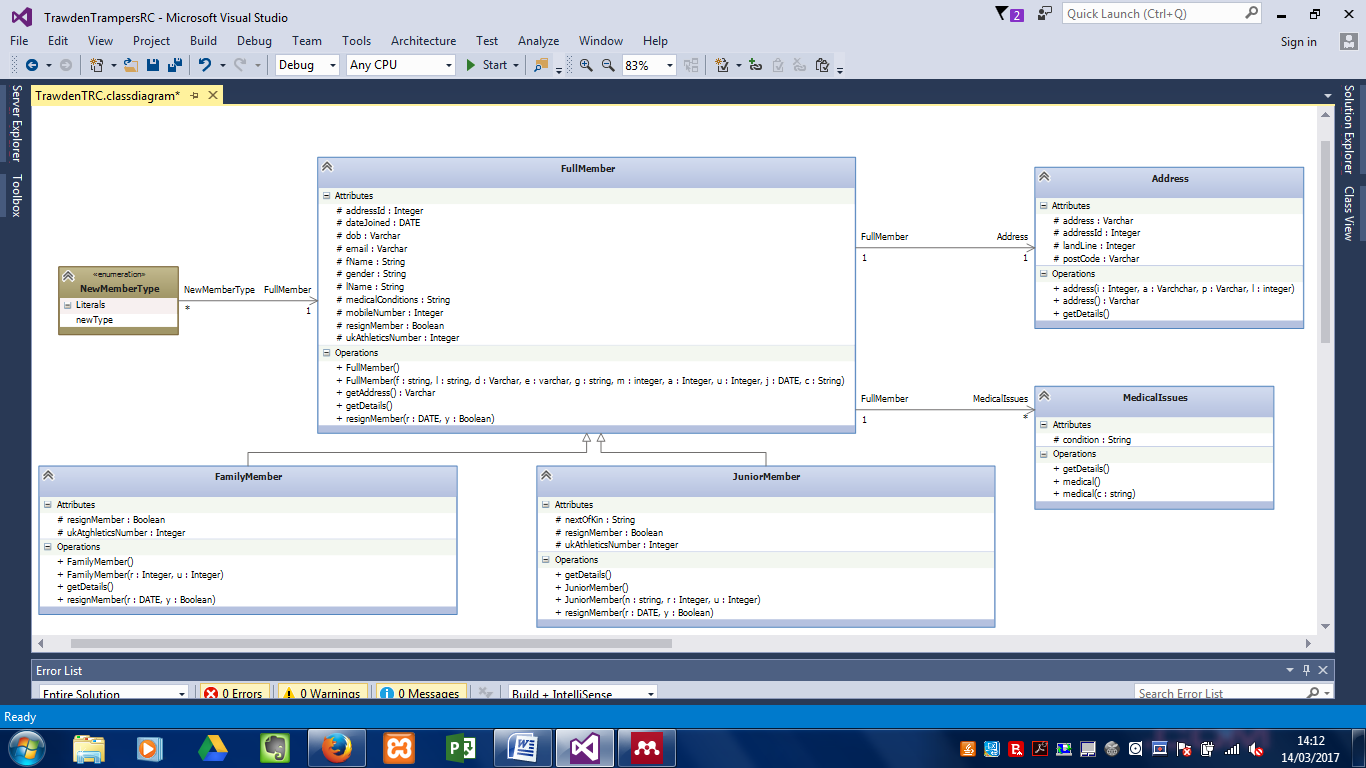
}

public void setukAthleticsNumber (string u)

{

num = u;

}

 }

}

Class Diagram : Full member Table

## Family Member.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace TTRCv1

{

public class FamilyMember : FullMember // Family member class is extention of full member class

{

private bool yesno;

private int num;

public new bool resignMember() // Used to address whether a family member has resigned from the club

{

return (yesno); // Option is boolean (Yes or No)

}

public int ukAthleticsNumber () // Unique ID

{

return (num); // Return unique number to member

}

public FamilyMember() // Listed class type

{

}

public FamilyMember(bool r, int u) // Class definition for above types

{

yesno = r;

num = u;

}

public new string getDetails() // Used for addressing and displaying details of family member

{

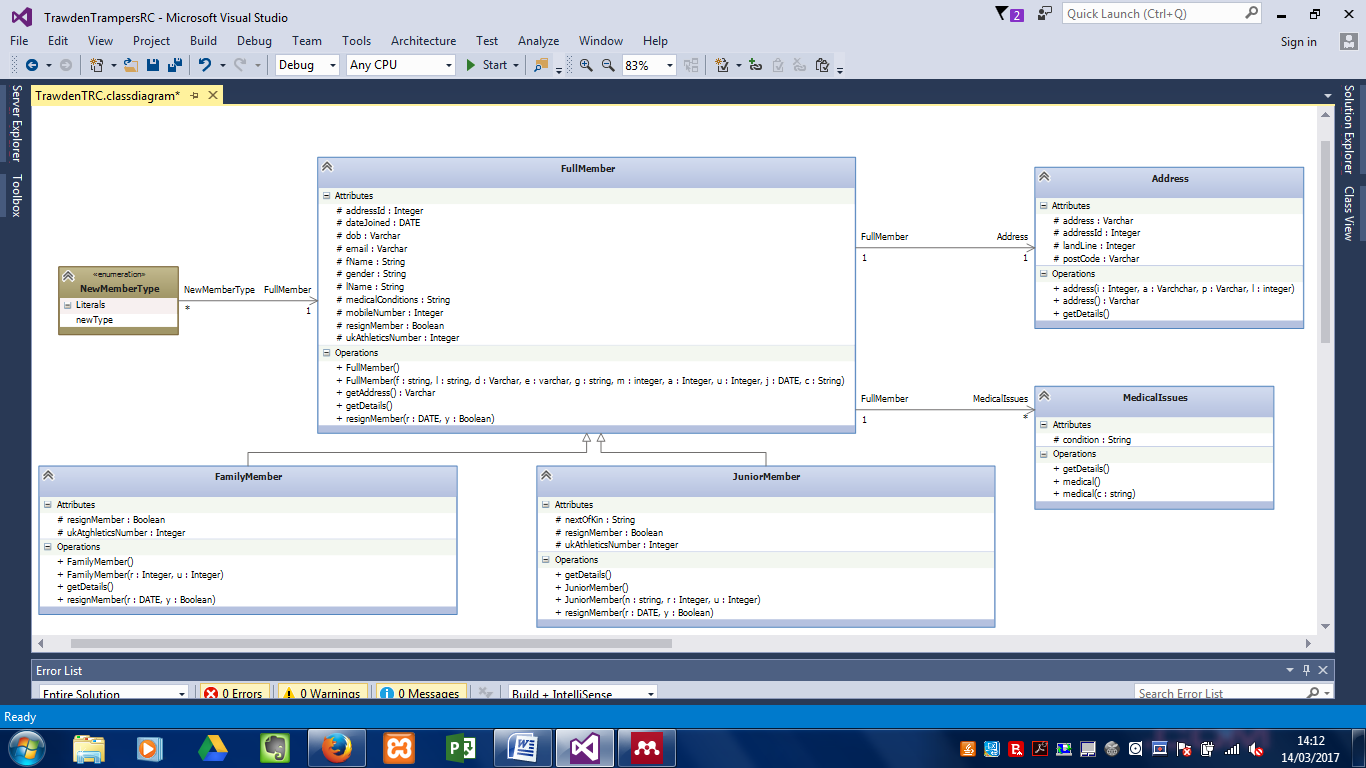
return (yesno + "" + num + "");

}

}

}

Class Diagram : Family Member Table



## Junior member.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace TTRCv1

{

public class JuniorMember : FullMember // Junior member is an extention of the full memebr class

{ // strings used to define unique identifiers

private string nok;

private bool yesno;

private int uk;

public virtual string nextOfKin () // Unique identifiers in class adaption.

{

return (nok);

}

public new bool resignMember()

{

return (yesno);

}

public new int ukAthleticsNumber()

{

return (uk);

}

public JuniorMember()

{

//yesno =

uk = 0;

}

public JuniorMember(string n, bool r, int u) // Identifiers used to returning code in 'memberLST' list object.

{

nok = n;

yesno = r;

uk = u;

}

public new string getDetails() // return details of junior member in 'memberLST' list box

{

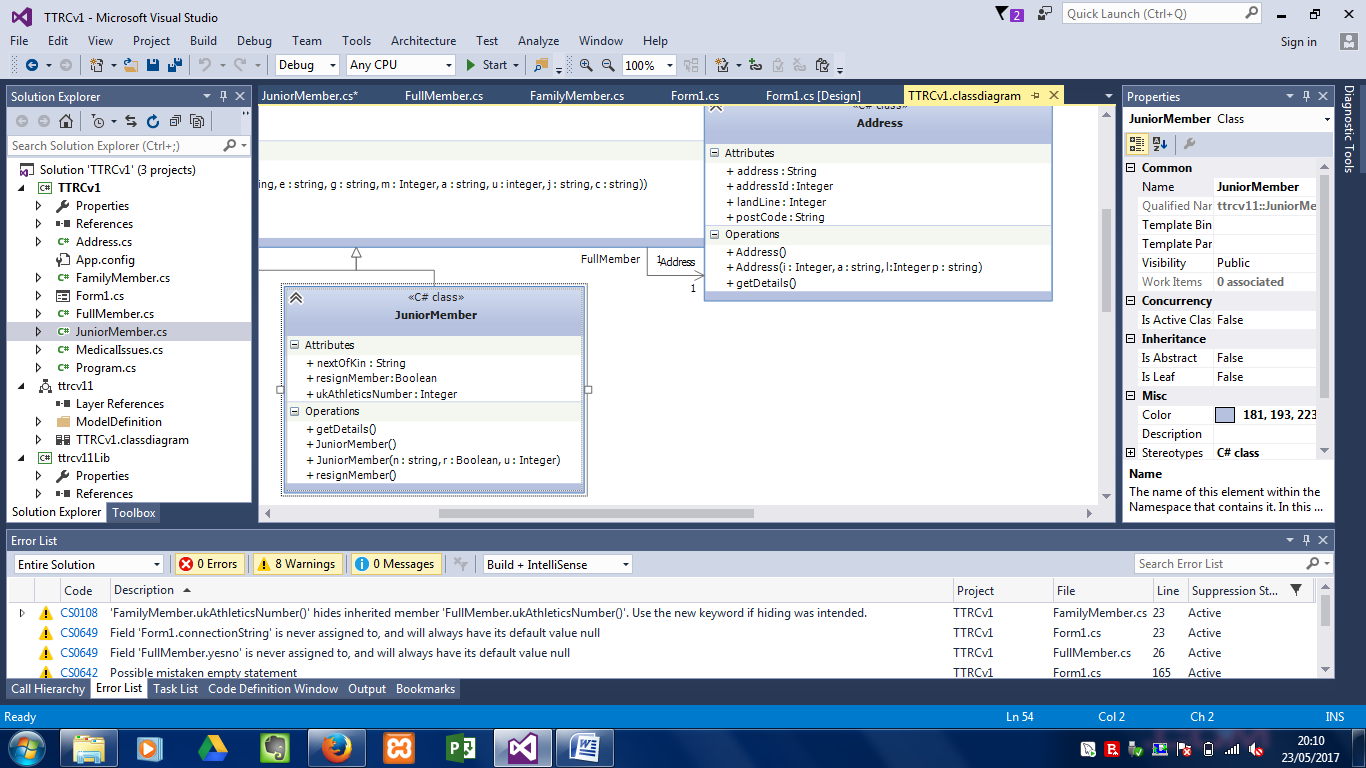
return (nok + " " + yesno + " " + uk + " ");

}

}

}

Class Diagram : Junior Member Table



## Address.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace TTRCv1 // Name space

{

public class Address // Class name

{

private string add; //class definitions

private int la;

private int id;

private string PC;

public string address() // used to create and return members details into a list object and the database

{

return (add);

}

public int landLine()

{

return (la);

}

public int addressId()

{

return(id);

}

public virtual string postCode()

{

return (PC);

}

public Address()

{

}

public Address(int i, string a,int l, string p) // list of return types

{

add = a;

la = l;

id = i;

PC = p;

}

public string getDetails() // Display member address details impleneted in a list

{

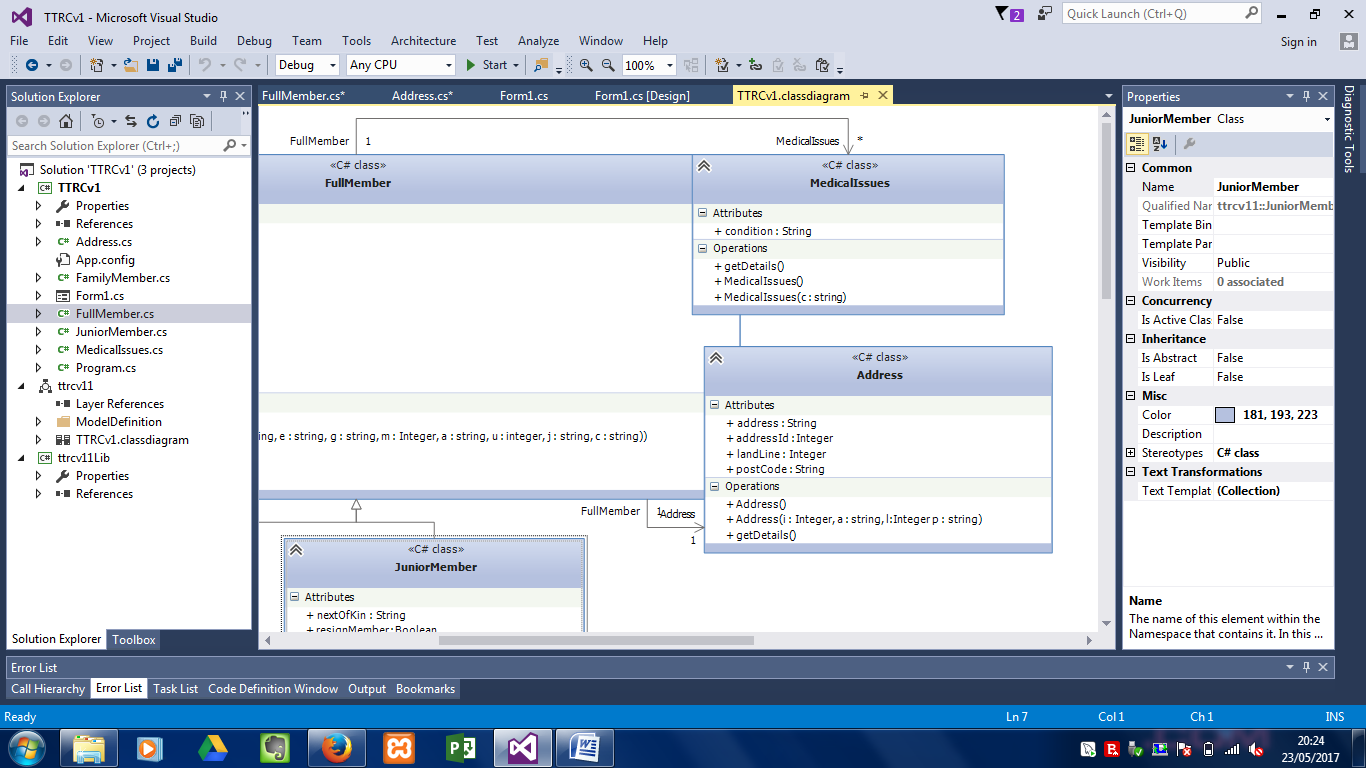
return (add + "" + la + "" + id + "" + PC + "");

}

}

}

Class diagram : Address Table



## Medical.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace TTRCv1 //name space of the project

{

public class MedicalIssues // medical issues table

{

private string con;

public string condition()

{

return (con); //return type

}

public MedicalIssues()

{

}

public MedicalIssues(string c)

{

con = c;

}

public string getDetails()

{

return (con + ""); // Input a members medical condition in the text box 'txtMed' and display in the list box 'memberLST'

}

}

}

Class diagram : Medical Table