```
//-- ***************************************
//--     Project:         HND Data Project
//--     Module:          PersonDB
//--     Author:          Paul McKillop
//--     Date created:    19 January 2019
//-- ***************************************


using System;
using System.Collections.Generic;
using System.Linq;
using RentalWPF.Models;
using Dapper;
using System.Windows;
using System.Data.SqlClient;

namespace RentalWPF.Data
{

    /// <summary>
    /// This class must perform FULL CRUD on Person objects
    /// </summary>
    public static class PersonDB
    {
        //-- Create
        public static int ClientInsert(string myDb, Client myClient)
        {
            //-- DB Connection
            var conn = AppUtility.DbConnection(myDb);

            try
            {
                //-- open the connection to the database
                if (conn.State == System.Data.ConnectionState.Closed)
                {
                    conn.Open();
                }

                //-- Set up the parameters
                DynamicParameters param = new DynamicParameters();
                param.Add("@ClientType_id", myClient.ClientTypeID);
                param.Add("@ClientForename", myClient.Forename);
                param.Add("@ClientSurname", myClient.Surname);
                param.Add("@AccountNumber", myClient.AccountNumber);
                param.Add("MobilePhone", myClient.MobilePhone);

                //-- Execute the stored procedure to store the vehicle
                conn.Execute("spClient_Insert", param, commandType: System.Data.
                CommandType.StoredProcedure);
            }
            catch (Exception ex)
            {

                MessageBox.Show(ex.Message);
            }
            finally
            {
                conn.Close();
            }

            return 1;
        }

        //-- Read all
        public static List<Client> ClientsGetAll(string myDb)
        {
            var clients = new List<Client>();

            var conn = AppUtility.DbConnection(myDb);

            try
```

```
        {
            //-- open the connection to the database
            if (conn.State == System.Data.ConnectionState.Closed)
            {
                conn.Open();
            }

            clients = conn.Query<Client>("spClient_GetAll", commandType: System.
            Data.CommandType.StoredProcedure).ToList();
        }
        catch (Exception ex)
        {

            MessageBox.Show(ex.Message);
        }
        finally
        {
            conn.Close();
        }

        return clients;
    }

    //-- Retrieve limited fields from Clients
    public static List<ClientBySurname> GetClientsBySurname(string myDb)
    {
        var clientsBySurname = new List<ClientBySurname>();

        var conn = AppUtility.DbConnection(myDb);

        try
        {
            //-- open the connection to the database
            if (conn.State == System.Data.ConnectionState.Closed)
            {
                conn.Open();
            }

            //-- Get the list by Dapper
            clientsBySurname = conn.Query<ClientBySurname>(
            "spClient_NamesBySurname", commandType: System.Data.CommandType.
            StoredProcedure).ToList();
        }
        catch (Exception ex )
        {
            MessageBox.Show(ex.Message);
        }
        finally
        {
            conn.Close();
        }

        return clientsBySurname;

    }

    //-- Read by ID
    public static Client ClientGetByID(string myDb, int clientId)
    {
        var client = new Client();

        var conn = AppUtility.DbConnection(myDb);

        try
        {
            //-- open the connection to the database
            if (conn.State == System.Data.ConnectionState.Closed)
            {
                conn.Open();
            }
```

```
            //-- Get the Client
            DynamicParameters param = new DynamicParameters();
            param.Add("@ClientID", clientId);

            client = conn.Query<Client>("spClient_GetByID", param, commandType:
            System.Data.CommandType.StoredProcedure) as Client;

            MessageBox.Show($"Forename: { client.Forename }");
            MessageBox.Show($"Surname: { client.Surname }");
            MessageBox.Show($"ClientTypeID: { client.ClientTypeID }");
            MessageBox.Show($"AccountNumber: { client.AccountNumber }");
            MessageBox.Show($"MobilePhone: { client.MobilePhone }");
        }
        catch (Exception ex)
        {

            MessageBox.Show(ex.Message);
        }
        finally
        {
            conn.Close();
        }

        return client;
    }

    //-- Update
    public static int ClientUpdate(string myDb, Client myClient)
    {
        var client = new Client();

        var conn = AppUtility.DbConnection(myDb);

        try
        {
            //-- open the connection to the database
            if (conn.State == System.Data.ConnectionState.Closed)
            {
                conn.Open();
            }

            DynamicParameters param = new DynamicParameters();
            param.Add("@ClientID",myClient.Id);
            param.Add("@ClientType_id", myClient.ClientTypeID);
            param.Add("@ClientForename",myClient.Forename);
            param.Add("@ClientSurname", myClient.Surname);
            param.Add("@AccountNumber",myClient.AccountNumber);
            param.Add("@MobilePhone", myClient.MobilePhone);

            conn.Execute("spClient_Update", param, commandType: System.Data.
            CommandType.StoredProcedure);

            MessageBox.Show($"{ myClient.Forename } { myClient.Surname } updated"
            );

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
        finally
        {
            conn.Close();
        }

        return 1;
    }

    //-- Delete
    public static int ClientDelete(string myDb, int myClientId)
    {
```

```
        var conn = AppUtility.DbConnection(myDb);

        try
        {
            //-- open the connection to the database
            if (conn.State == System.Data.ConnectionState.Closed)
            {
                conn.Open();
            }

            DynamicParameters param = new DynamicParameters();
            param.Add("@ClientID", myClientId);

            conn.Execute("spClient_Delete", param, commandType: System.Data.
            CommandType.StoredProcedure);
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
        finally
        {
            conn.Close();
        }

        return 1;
    }


    //-- Demo of data retrieval using traditional SQl
    //-- rather than Dapper
    public static Client GetClientByIDSQL(string myDb,int clientID)
    {
        var myClient = new Client();

        var conn = AppUtility.DbConnection(myDb);

        var selectStatement = "SELECT * FROM Client WHERE id = @clientId";

        var selectCommand = new SqlCommand(selectStatement, conn);

        selectCommand.Parameters.AddWithValue("@clientId", clientID);


        try
        {
            //-- open the connection to the database
            if (conn.State == System.Data.ConnectionState.Closed)
            {
                conn.Open();
            }

            var reader = selectCommand.ExecuteReader();

            if (!reader.Read())
            {
                throw new InvalidOperationException("No record found");
            }

            myClient.Id = Convert.ToInt32(reader["id"]);
            myClient.ClientTypeID = Convert.ToInt32(reader["ClientType_id"]);
            myClient.Forename = reader["ClientForename"].ToString();
            myClient.Surname = reader["ClientSurname"].ToString();
            myClient.AccountNumber = reader["AccountNumber"].ToString();
            myClient.MobilePhone = reader["MobilePhone"].ToString();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
        finally
```

```
            {
                conn.Close();
            }

            return myClient;
        }


        //-- Method to enter a new Client Address
        public static int EnterClientAddress(string myDb, ClientAddress myAddress)
        {
            var conn = AppUtility.DbConnection(myDb);

            try
            {
                //-- open the connection to the database
                if (conn.State == System.Data.ConnectionState.Closed)
                {
                    conn.Open();
                }

                DynamicParameters param = new DynamicParameters();
                param.Add("@Client_Id", myAddress.Client_Id);
                param.Add("@Address1",myAddress.Address1);
                param.Add("@Address2", myAddress.Address2);
                param.Add("@Town", myAddress.Town);
                param.Add("@County", myAddress.County);
                param.Add("@Postcode", myAddress.Postcode);
                param.Add("@Landline", myAddress.Landline);

                conn.Execute("spClient_InsertAddress", param, commandType: System.
                Data.CommandType.StoredProcedure);
            }
            catch (Exception ex)
            {

                MessageBox.Show(ex.Message);
            }
            finally
            {
                conn.Close();
            }

            return 1;
        }
    }
}
```