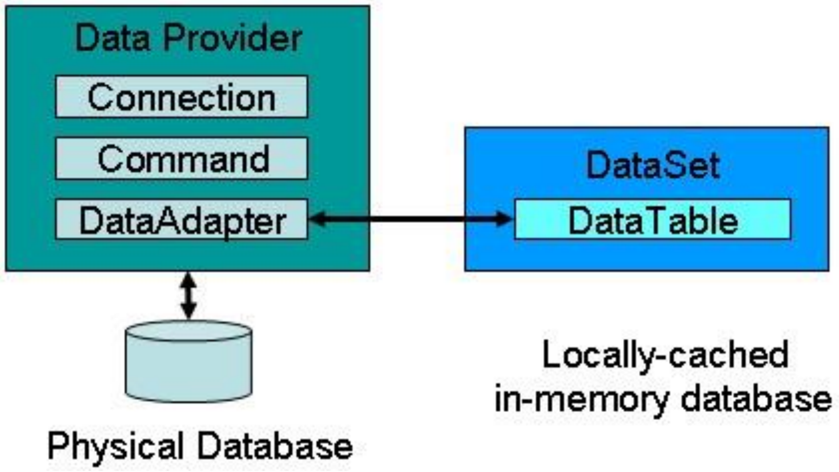
OBJECTIVES:

1. **Understanding and working with disconnected data access.**
2. **Understanding and working with installer package creation.**
3. **Understanding and working with LINQ**
4. **Understanding and working with MDI applications 5. Practice activities.**

OBJECTIVE 1: Understanding and working with disconnected data access.

Disconnected data access

1. Disconnected data access architecture supports access of data even when no connection is active with database server, this is achieved by creating in memory copy of data, this means creation of data store at local machine.
2. Data set is mechanism for creation of in memory data store.
3. The DataSet is a memory-resident representation of data that provides a consistent relational programming model regardless of the source of the data it contains.
4. One of the key characteristics of the DataSet is that it has no knowledge of the underlying Data Source that might have been used to populate it.
5. This is b/c of introduction of DataAdapter class which is facilitator for performing operations on a dataset and from dataset to database.
6. DataSet is a class for creation of Data set
7. DataTable is a class for creation of table/relation in data set and will be added in dataset by Add method Tables collection of dataset
8. DataColumn is a class for creation of column of a table and will be added in table by Add method of Columns collection of table
9. DataRow is a class for creation of row in a table and will be added in a table by Add method of Rows collection of a table



OBJECTIVE 2: Understanding and working with installer package creation.

⇨ Windows Installer deployment allows you to create installer packages to be distributed to users; the user runs the setup file and steps through a wizard to install the application.

⇨ This is done by adding a Setup project to your solution; when built, it creates a setup file that you distribute to users; the user runs the setup file and steps through a wizard to install the application.

⇨ There are many ways to do this and one of those ways is set up project creation.

⇨ **Setup** projects allow you to create installers in order to distribute an application.

⇨ The resulting Windows Installer (.msi (msi=Microsoft installer)) file contains the application, any dependent files, information about the application such as registry entries, and instructions for installation.

⇨ When the .msi file is distributed and runs on another computer, you can be assured that everything necessary for installation is included; if for any reason the installation fails (for example, the target computer does not have the required operating system version), the installation will be rolled back and the computer returned to its pre-installation state.

⇨ There are two types of setup projects in Visual Studio: **Setup** projects and **Web Setup** projects. The distinction between **Setup** and **Web Setup** projects is where the installer will be deployed:

- **Setup** projects will install files into the file system of a target computer.

  - **Web Setup** pro

Installer Package Reference:
2013 and onwards : https://www.advancedinstaller.com/user-guide/tutorial-ai-ext-vs.html

OBJECTIVE 3: Understanding and working with LINQ.

LINQ

⇨ A *query* is an expression that retrieves data from a data source. Queries are usually expressed in a specialized query language. Different languages have been developed over time for the various types of data sources, for example SQL for relational databases and XQuery for XML Therefore, developers have had to learn a new query language for each type of data source or data format that they must support.

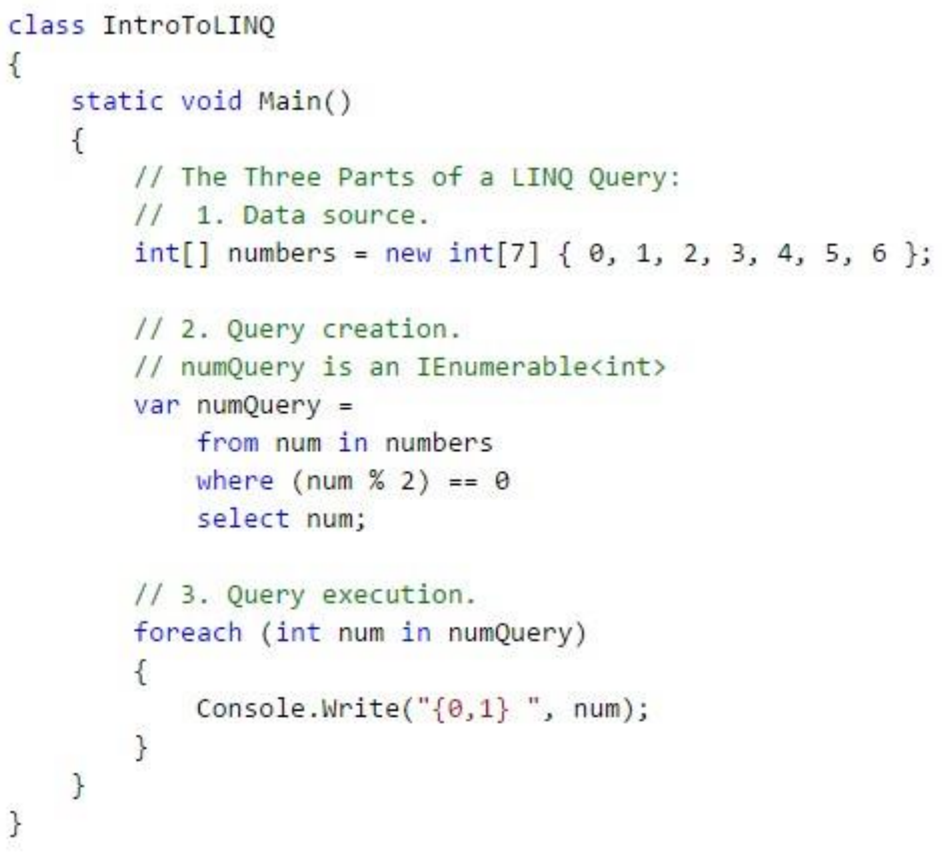
⇨ LINQ simplifies this situation by offering a consistent model for working with data across various kinds of data sources and formats.

⇨ In a LINQ query, you are always working with objects. You use the same basic coding patterns to query and transform data in XML documents, SQL databases, ADO.NET Datasets, .NET collections, and any other format for which a LINQ provider is available.

⇨ All LINQ query operations consist of three distinct actions:

- Obtain the data source.
- Create the query. ○

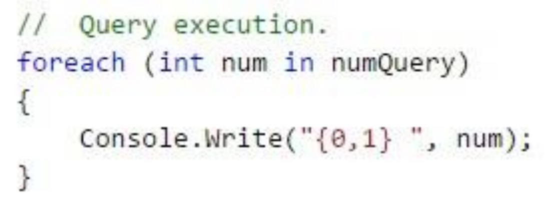
Execute the query



```
class IntroToLINQ
{
    static void Main()
    {
        // The Three Parts of a LINQ Query:
        //  1. Data source.
        int[] numbers = new int[7] { 0, 1, 2, 3, 4, 5, 6 };

        // 2. Query creation.
        // numQuery is an IEnumerable<int>
        var numQuery =
            from num in numbers
            where (num % 2) == 0
            select num;

        // 3. Query execution.
        foreach (int num in numQuery)
        {
            Console.Write("{0,1} ", num);
        }
    }
}
```

Deferred Execution

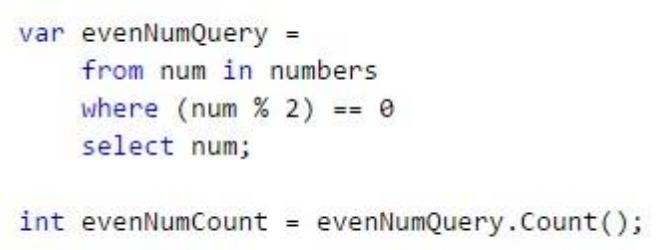
As stated previously, the query variable itself only stores the query commands. The actual execution of the query is deferred until you iterate over the query variable in a foreach statement. This concept is referred to as *deferred execution* and is demonstrated in the following example:



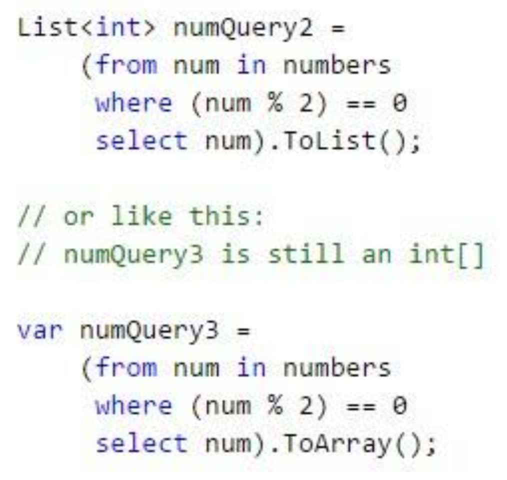
```
//  Query execution.
foreach (int num in numQuery)
{
    Console.Write("{0,1} ", num);
}
```

Forcing Immediate Execution

Queries that perform aggregation functions over a range of source elements must first iterate over those elements. Examples of such queries are Count, Max, Average, and First. These execute without an explicit foreach statement because the query itself must use foreach in order to return a result. Note also that these types of queries return a single value, not an IEnumerablecollection. The following query returns a count of the even numbers in the source array:



```
var evenNumQuery =
    from num in numbers
    where (num % 2) == 0
    select num;

int evenNumCount = evenNumQuery.Count();
```



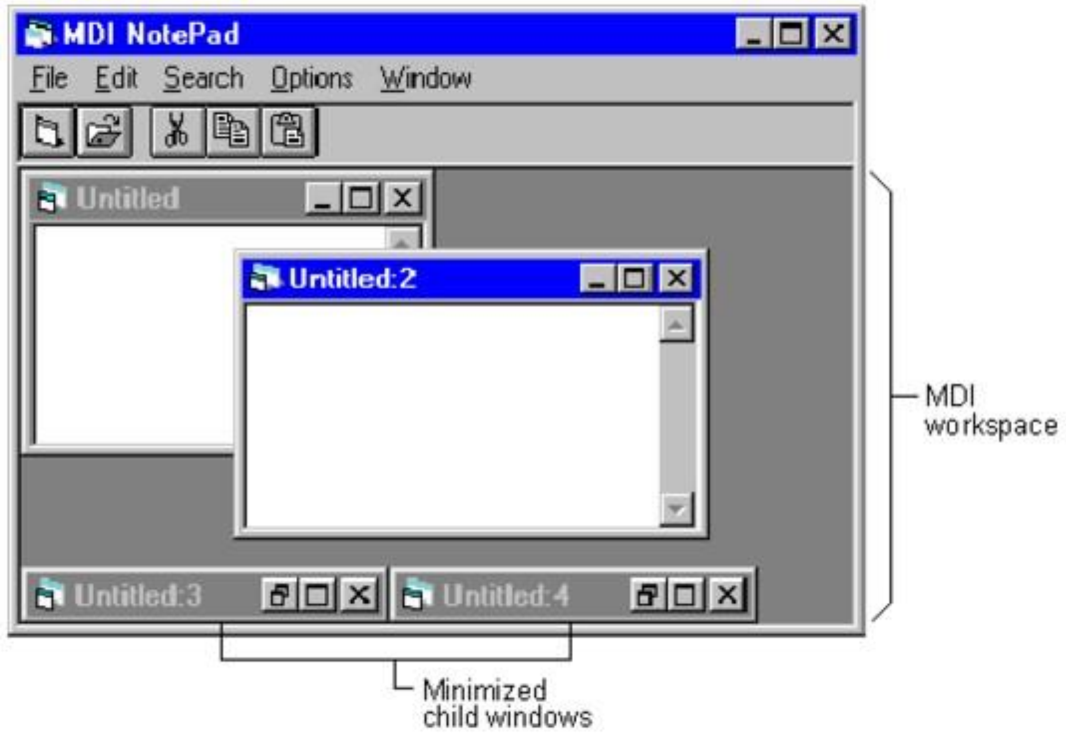
```
List<int> numQuery2 =
    (from num in numbers
     where (num % 2) == 0
     select num).ToList();

// or like this:
// numQuery3 is still an int[]

var numQuery3 =
    (from num in numbers
     where (num % 2) == 0
     select num).ToArray();
```

OBJECTIVE 4: Understanding and working MDI applications.

MDI application

1. The multiple-document interface (MDI) allows you to create an application that maintains multiple forms within a single container form.

2. Applications such as Microsoft Excel and Microsoft Word for Windows have multiple-document interfaces.
3. An MDI application allows the user to display multiple documents at the same time, with each document displayed in its own window.
4. Documents or *child windows* are contained in a *parent window*, which provides a workspace for all the child windows in the application.
5. For example, Microsoft Excel allows you to create and display multipledocument windows of different types. Each individual window is confined to the area of the Excel parent window. When you minimize Excel, all of the document windows are minimized as well; only the parent window's icon appears in the task bar.
6. A child form is an ordinary form that has its MDIChild property set to True.
7. Your application can include many MDI child forms of similar or different types.
8. At run time, child forms are displayed within the *workspace* of the MDI parent form.
9. When a child form is minimized, its icon appears within the workspace of the MDI form instead of on the taskbar.

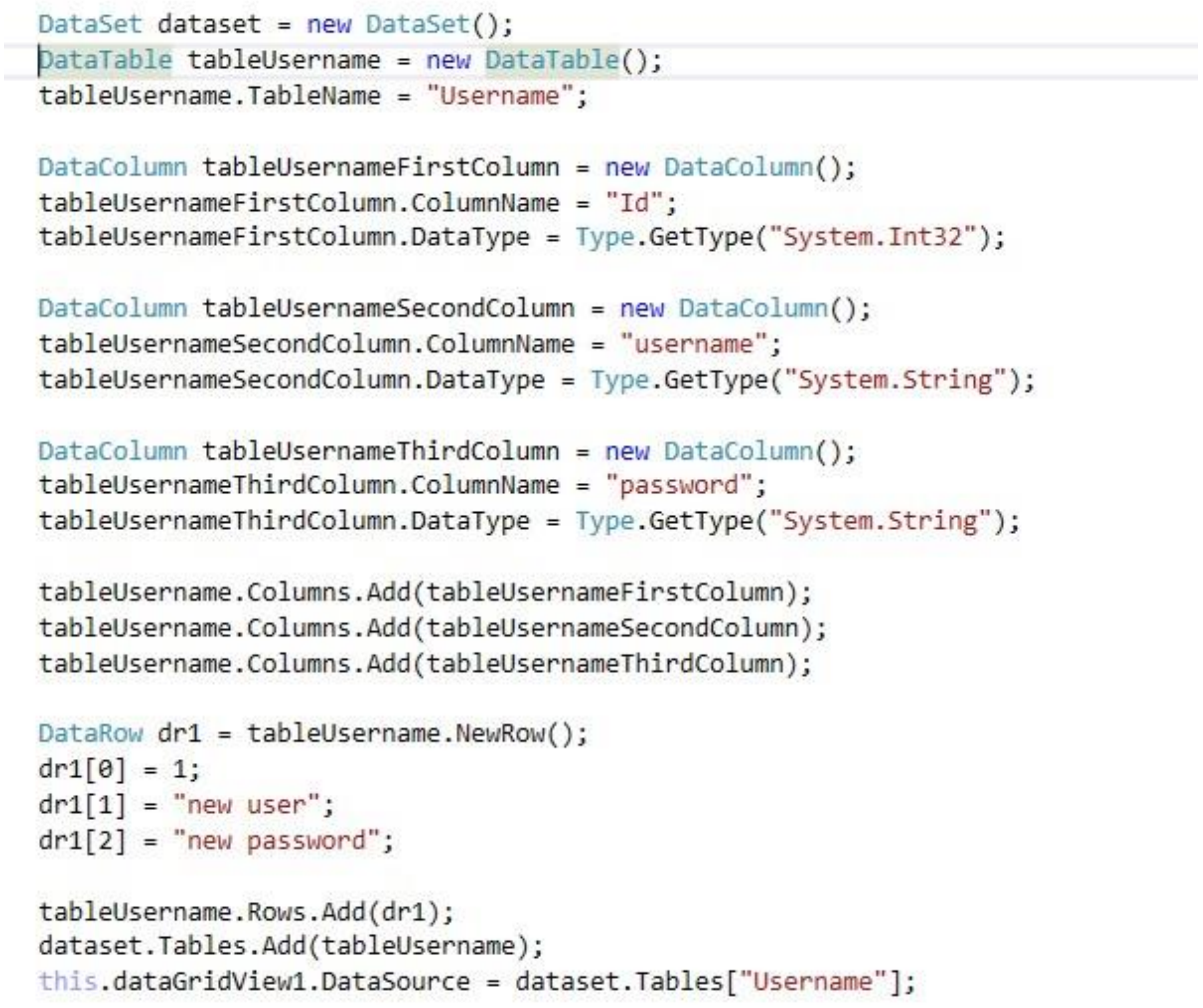


ACTIVITIES SECTION

Installer Package Reference:
2013 and onwards : https://www.advancedinstaller.com/user-guide/tutorial-ai-ext-vs.html

ACTIVITY 1: STEPS

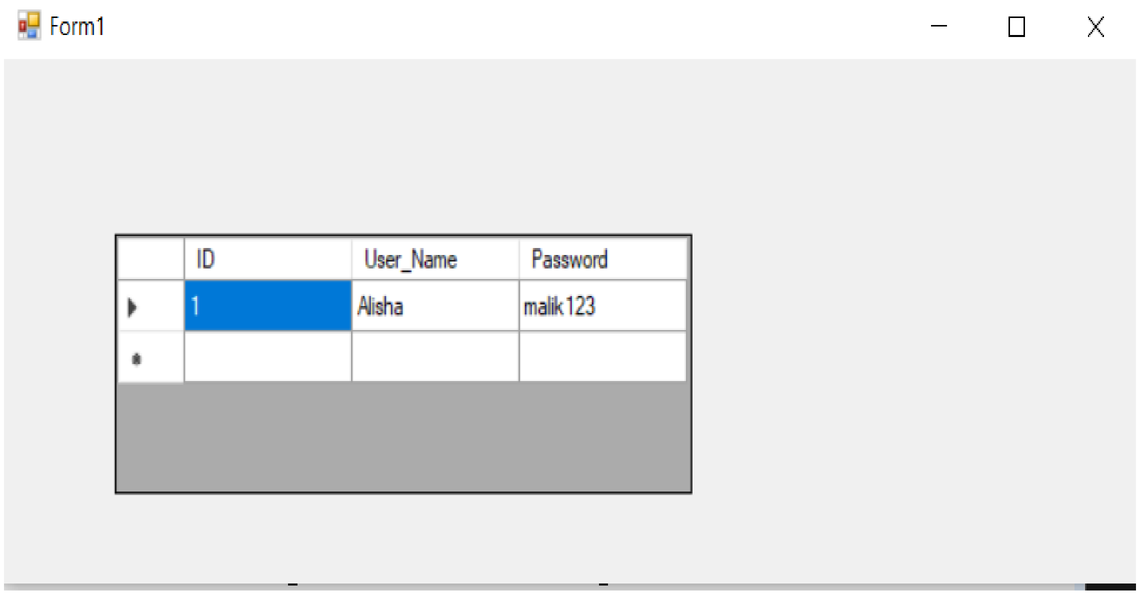
- ⇨ Create windows forms application named dsApp.
- ⇨ Create a form in it.
- ⇨ Place DataGridView on form.
- ⇨ On Form_Load event write following code in the form.



```
DataSet dataset = new DataSet();
DataTable tableUsername = new DataTable();
tableUsername.TableName = "Username";

DataColumn tableUsernameFirstColumn = new DataColumn();
tableUsernameFirstColumn.ColumnName = "Id";
tableUsernameFirstColumn.DataType = Type.GetType("System.Int32");

DataColumn tableUsernameSecondColumn = new DataColumn();
tableUsernameSecondColumn.ColumnName = "username";
tableUsernameSecondColumn.DataType = Type.GetType("System.String");

DataColumn tableUsernameThirdColumn = new DataColumn();
tableUsernameThirdColumn.ColumnName = "password";
tableUsernameThirdColumn.DataType = Type.GetType("System.String");

tableUsername.Columns.Add(tableUsernameFirstColumn);
tableUsername.Columns.Add(tableUsernameSecondColumn);
tableUsername.Columns.Add(tableUsernameThirdColumn);

DataRow dr1 = tableUsername.NewRow();
dr1[0] = 1;
dr1[1] = "new user";
dr1[2] = "new password";

tableUsername.Rows.Add(dr1);
dataset.Tables.Add(tableUsername);
this.dataGridView1.DataSource = dataset.Tables["Username"];
```

OUTPUT:

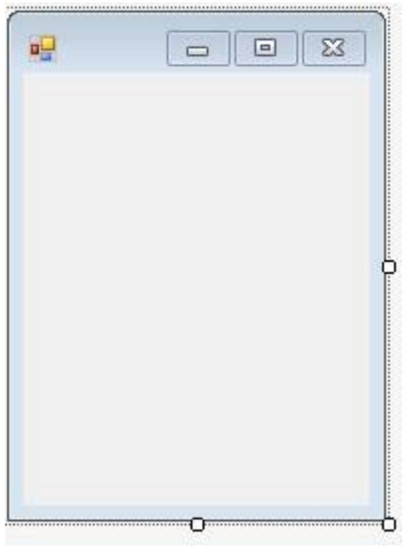
Installer Package Reference:
2013 and onwards : https://www.advancedinstaller.com/user-guide/tutorial-ai-ext-vs.html



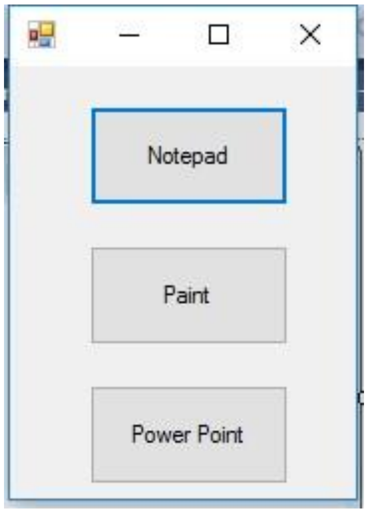
	ID	User_Name	Password
▸	1	Alisha	malik123
*			

ACTIVITY 2: STEPS

⇨ Create a windows forms application named ShortCutsApp. ⇨ Create a form in this application named frmShortCut



⇨ When form loads following interface should be displayed to user.



⇨ When user clicks button that has text "Notepad", open a notepad editor for user or simply launch notepad editor.
⇨ When user clicks button with text "Paint", open MS paint for user.
⇨ When user clicks button with text "Power Point", open power point for user.

[Note]
⇨ All buttons should be created and displayed at runtime.
⇨ Event handling should also be done at runtime.
⇨ You can use following command for opening relevant application

```
System.Diagnostics.Process.Start("notepad.exe");
```

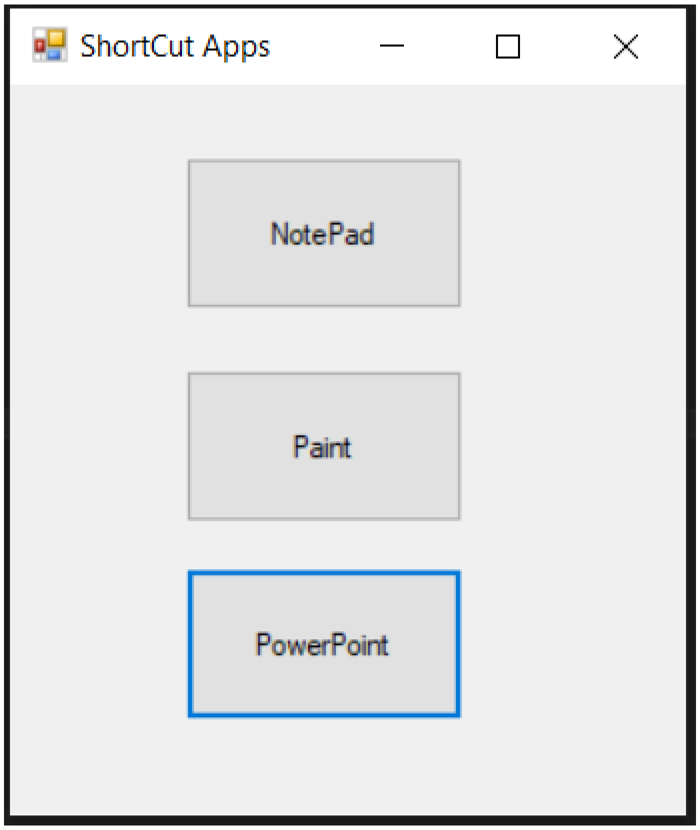
⇨ **System** is namespace
⇨ **Diagnostics** is also namespace
⇨ **Process** is a class: it provides way to start and stop local system processes.
⇨ **Start** is a method that requires name of exe file of application.
Here are names of exe files of applications that are required for example
1. Notepad: notepad.exe
2. Paint: mspaint.exe
3. Power Point: powerpnt.exe

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
```

Installer Package Reference:
2013 and onwards : https://www.advancedinstaller.com/user-guide/tutorial-ai-ext-vs.html

```
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Lab08_Task01
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            System.Diagnostics.Process.Start("notepad.exe");
        }

        private void button2_Click(object sender, EventArgs e)
        {
            System.Diagnostics.Process.Start("mspaint.exe");
        }

        private void button3_Click(object sender, EventArgs e)
        {

                System.Diagnostics.Process.Start("POWERPNT.EXE");
        }
    }
}
```

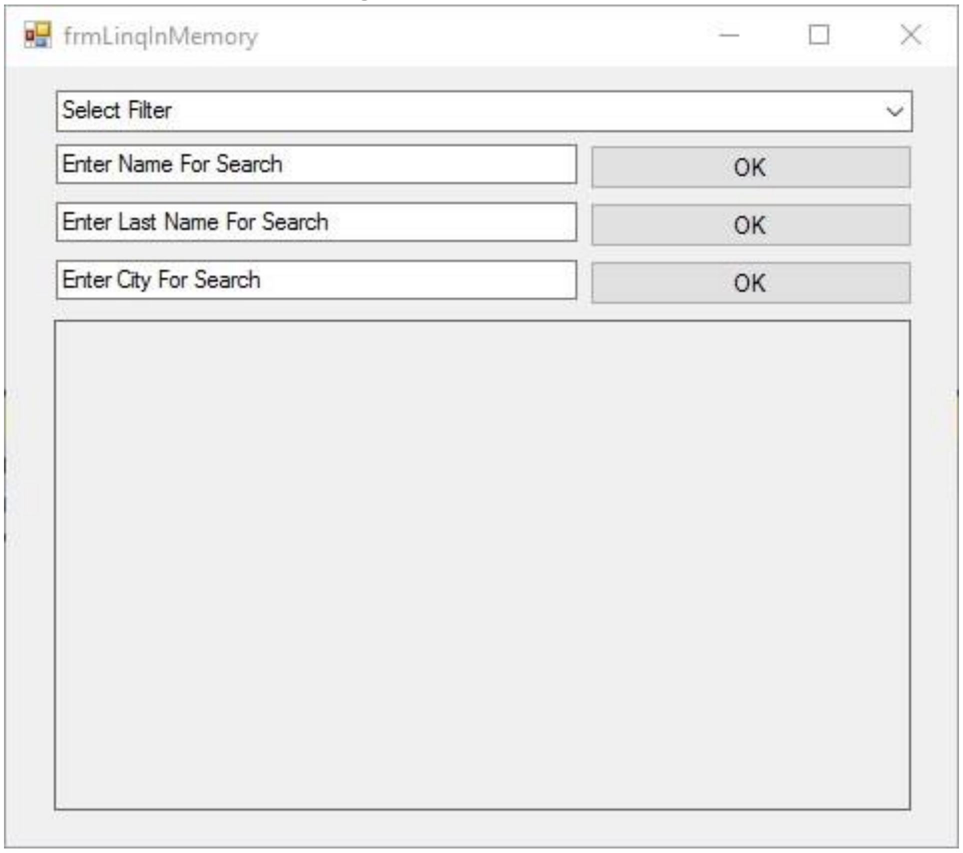
Installer Package Reference:
2013 and onwards : https://www.advancedinstaller.com/user-guide/tutorial-ai-ext-vs.html

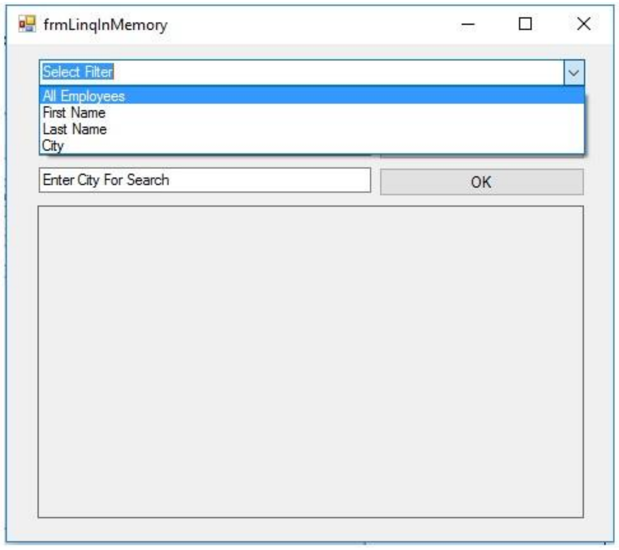


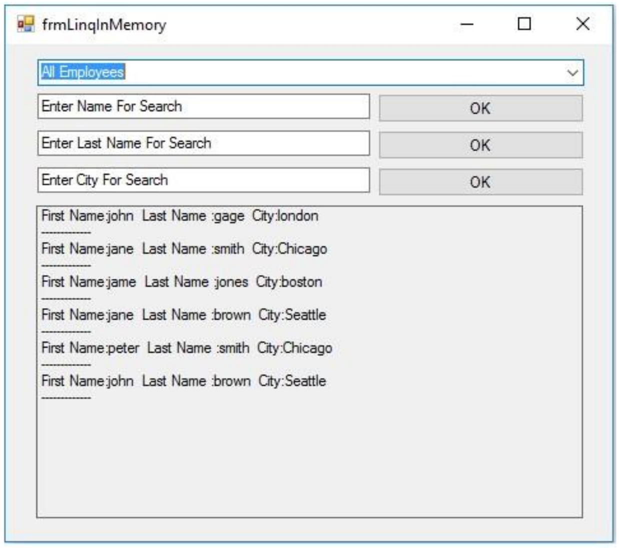
Installer Package Reference:
2013 and onwards : https://www.advancedinstaller.com/user-guide/tutorial-ai-ext-vs.html

ACITVITY 3: STEPS

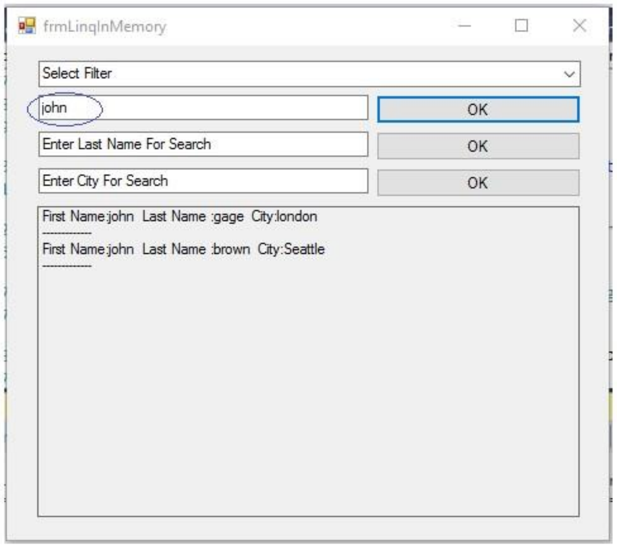
- ⇨ Create an application named LinqExamples.
- ⇨ Create a form named frmLinq.
- ⇨ Create interface of form as given



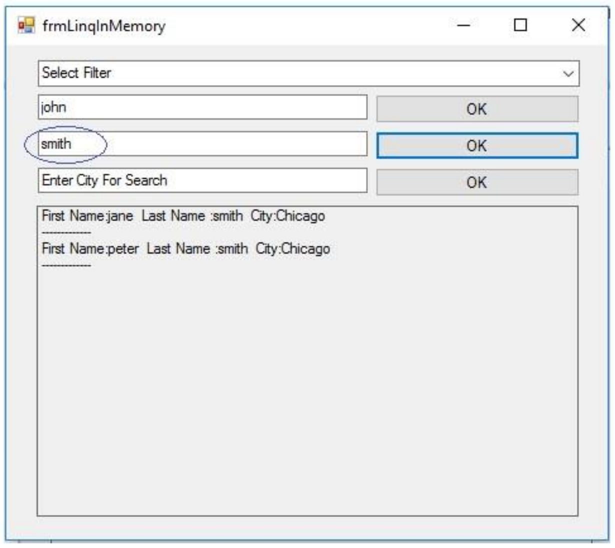




```
First Name:john  Last Name :gage  City:london
-----------
First Name:jane  Last Name :smith  City:Chicago
-----------
First Name:jame  Last Name :jones  City:boston
-----------
First Name:jane  Last Name :brown  City:Seattle
-----------
First Name:peter  Last Name :smith  City:Chicago
-----------
First Name:john  Last Name :brown  City:Seattle
-----------
```



```
First Name:john  Last Name :gage  City:london
-----------
First Name:john  Last Name :brown  City:Seattle
-----------
```



```
First Name:jane  Last Name :smith  City:Chicago
-----------
First Name:peter  Last Name :smith  City:Chicago
-----------
```

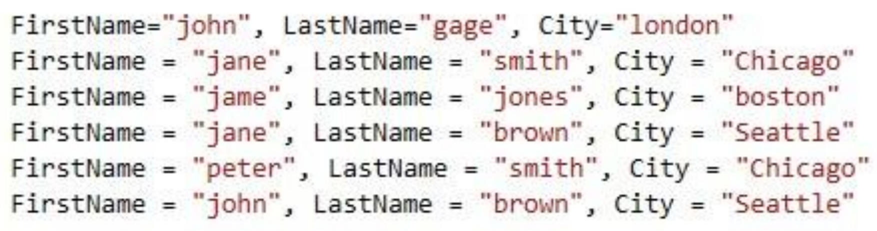
PROGRAM SETUP

⇨ Create a class named Employee with three fields and three properties.
- **firstName** field and **FirstName** property for it.
- **lastName** field and **LastName** property for it.
- **city** field and **City** property for it.

⇨ Create an instance of **List** class named **employees** which should be accessible in all methods of a form, this **employees** will be used as a data holder for different employee objects of **Employee** class.

⇨ Populate the **employees** object of List class using its Add method.
- Note: Here we are adding objects of Employee class in List.

⇨ Before adding employee objects to List setup their values as given here:

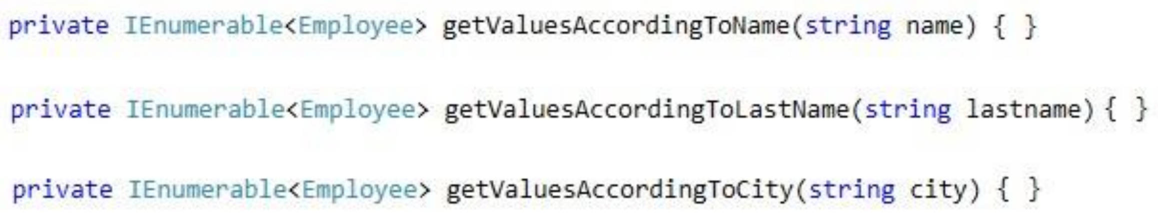


```
FirstName="john", LastName="gage", City="london"
FirstName = "jane", LastName = "smith", City = "Chicago"
FirstName = "jame", LastName = "jones", City = "boston"
FirstName = "jane", LastName = "brown", City = "Seattle"
FirstName = "peter", LastName = "smith", City = "Chicago"
FirstName = "john", LastName = "brown", City = "Seattle"
```

⇨ **Program Requirements**

1. Display records from data holder on basis of filter applied using combo box.

2. After that implement following methods for searching records relevant to values of text boxes.



```
private IEnumerable<Employee> getValuesAccordingToName(string name) { }

private IEnumerable<Employee> getValuesAccordingToLastName(string lastname) { }

private IEnumerable<Employee> getValuesAccordingToCity(string city) { }
```

3. Place combo box besides of each text box that shows two values ascending or descending and now display records according to order selected.
[Note]: ○ You can use orderby clause for this purpose.

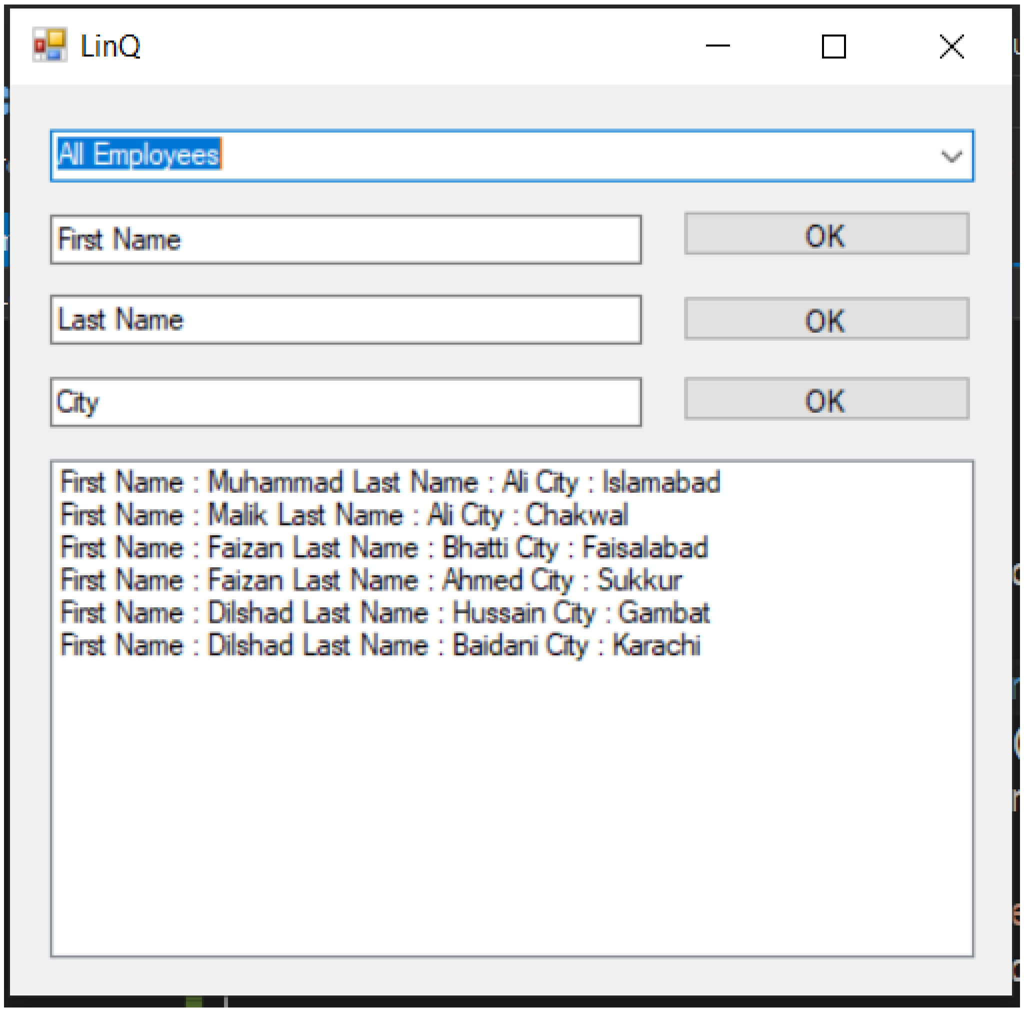
○ For example: orderby columnName
○ After getting filtered records you can write other LINQ query which will sort the records.

Code :

```
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;

namespace Lab08_Task01
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        public class Employee
        {
            public string name{get; set;}
            public string last_name { get; set; }
            public string city { get; set; }
        }
        Employee[] arr = new Employee[6];
        string add = "";

        private void Form1_Load(object sender, EventArgs e)
        {
            arr[0] = new Employee { name = "Muhammad", last_name = "Ali", city = "Islamabad" };
            arr[1] = new Employee { name = "Malik", last_name = "Ali", city = "Chakwal" };
            arr[2] = new Employee { name = "Faizan", last_name = "Bhatti", city = "Faisalabad" };
            arr[3] = new Employee { name = "Faizan", last_name = "Ahmed", city = "Sukkur" };
            arr[4] = new Employee { name = "Dilshad", last_name = "Hussain", city = "Gambat" };
            arr[5] = new Employee { name = "Dilshad", last_name = "Baidani", city = "Karachi" };


        }

        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {


            if (comboBox1.SelectedIndex == 0)
            {

                var query = from p in arr select p;

                foreach (var r in query)
                {

                    add = "First Name : " + r.name + " Last Name : " + r.last_name + " City : " +
r.city;
                    listBox1.Items.Add(add);
                }
            }

            else if (comboBox1.SelectedIndex == 1)
            {

                var query = from p in arr select p;
                listBox1.Items.Clear();
                foreach (var r in query)
                {


                    add = "First Name : " + r.name ;
                    listBox1.Items.Add(add);
                }
            }

            else if (comboBox1.SelectedIndex == 2)
            {

                var query = from p in arr select p;
                listBox1.Items.Clear();
                foreach (var r in query)
                {


                    add = "Last Name : " + r.last_name;
                    listBox1.Items.Add(add);
                }
            }

            else if (comboBox1.SelectedIndex == 3)
            {
                var query = from p in arr select p;

                listBox1.Items.Clear();
                foreach (var r in query)
                {


                    add = "City : " + r.city;
                    listBox1.Items.Add(add);
                }
            }
        }

        private void button1_Click(object sender, EventArgs e)
        {
            var query = from p in arr where p.name == textBox1.Text select p;
            listBox1.Items.Clear();
            foreach(var r in query)
            {
                add = "First Name : " + r.name + " Last Name : " + r.last_name + " City : " + r.city;
                listBox1.Items.Add(add);
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            var query = from p in arr where p.last_name == textBox2.Text select p;
            listBox1.Items.Clear();
            foreach (var r in query)
            {
                add = "First Name : " + r.name + " Last Name : " + r.last_name + " City : " + r.city;
                listBox1.Items.Add(add);
            }
        }

        private void button3_Click(object sender, EventArgs e)
        {
            var query = from p in arr where p.city == textBox3.Text select p;
            listBox1.Items.Clear();
            foreach (var r in query)
            {
                add = "First Name : " + r.name + " Last Name : " + r.last_name + " City : " + r.city;
                listBox1.Items.Add(add);
            }
        }
    }
}
```

Output:

Installer Package Reference:
2013 and onwards : https://www.advancedinstaller.com/user-guide/tutorial-ai-ext-vs.html



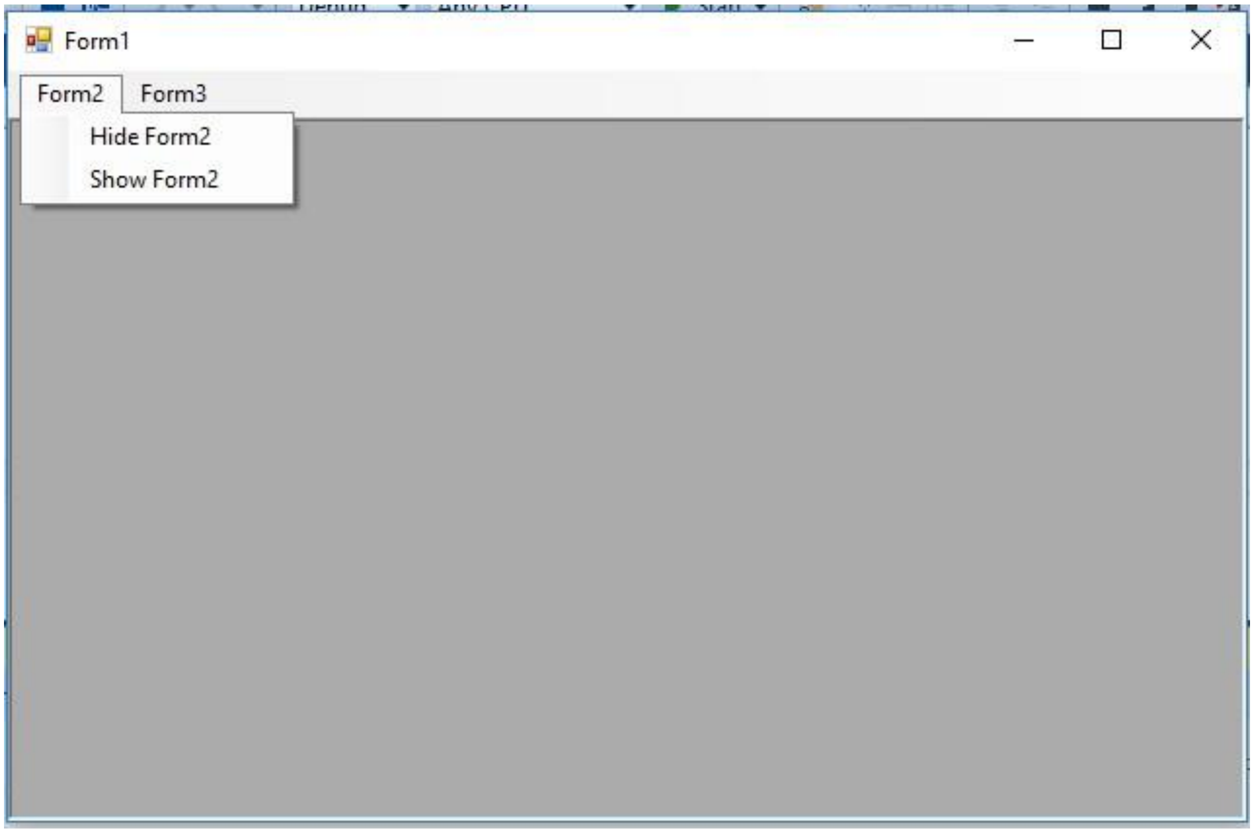
```
First Name : Muhammad Last Name : Ali City : Islamabad
First Name : Malik Last Name : Ali City : Chakwal
First Name : Faizan Last Name : Bhatti City : Faisalabad
First Name : Faizan Last Name : Ahmed City : Sukkur
First Name : Dilshad Last Name : Hussain City : Gambat
First Name : Dilshad Last Name : Baidani City : Karachi
```

Activity 4

- ⇨ Create an application named MDIApp.
- ⇨ Create three forms in this application.

- ⇨ Create interface of form1 as given in image.
- ⇨ Set **isMdiContainer** property of form1 to **true**.

Installer Package Reference:
2013 and onwards : https://www.advancedinstaller.com/user-guide/tutorial-ai-ext-vs.html



⇨ Show form2 using Show() method
⇨ Hide form2 using Hide() method

⇨ Implement FormClosing event of Form2 and Form3 and write following code there.
  - Cancel the event by using this line of code: e.Cancel = true; ○
  Now hide the form by using this line of code: this.Hide();

⇨ Do the same with Form3.

⇨ Execute the code and validate the results.

Code:

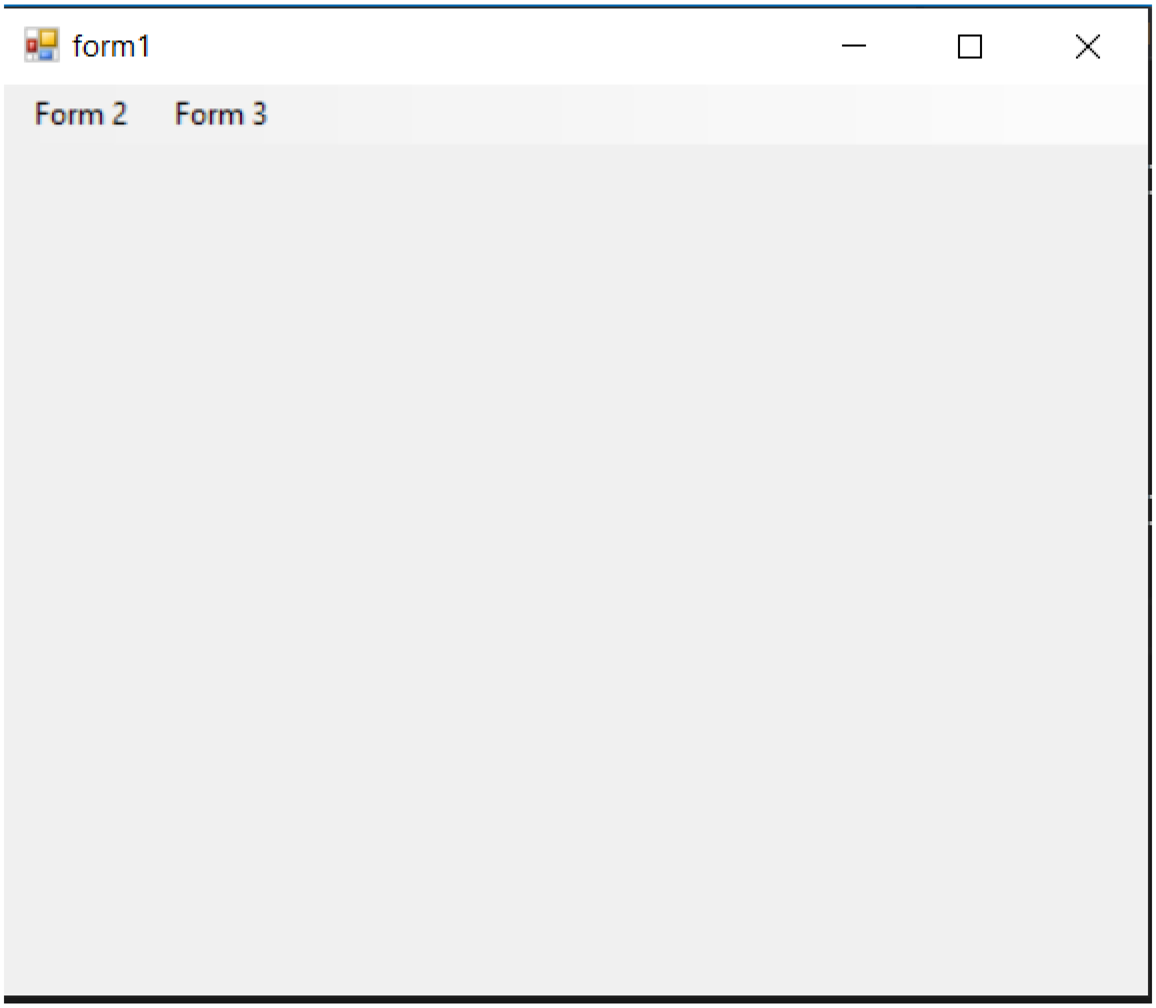
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Lab08_Task01
{
```

Installer Package Reference:
2013 and onwards : https://www.advancedinstaller.com/user-guide/tutorial-ai-ext-vs.html

```
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        Form2 test   = new Form2();
        Form3 test1 = new Form3();
        private void Form1_Load(object sender, EventArgs e)
        {



        }

        private void hideForm2ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            test.Hide();
        }

        private void showForm2ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            test.Show();
        }

        private void hideForm3ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            test1.Hide();
        }

        private void showForm3ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            test1.Show();
        }
    }
}
```

Output:

Installer Package Reference:
2013 and onwards : https://www.advancedinstaller.com/user-guide/tutorial-ai-ext-vs.html

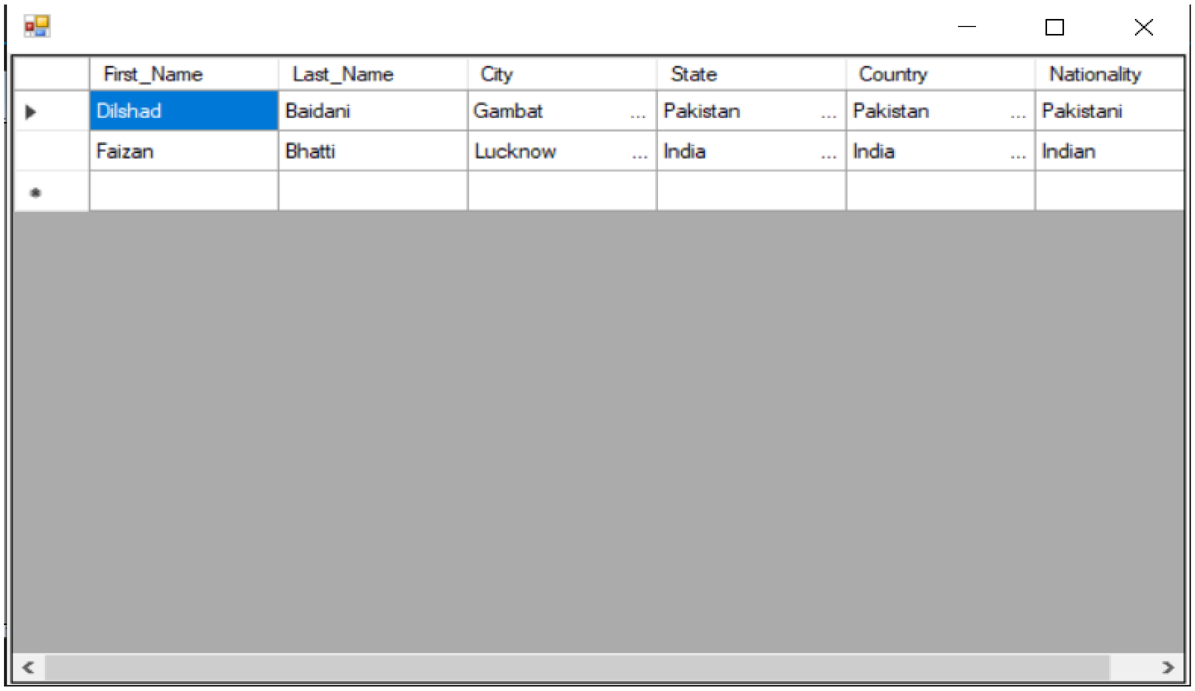


Activity 5

⇨ Create windows forms application named disconnectedAccess.
⇨ Create a form in it.
⇨ Place DataGridView on form.
⇨ In load event of form place following code.

```
// step 1 -- connection
string connectionString = ConfigurationManager.ConnectionStrings["cAString"].ConnectionString;

SqlConnection connection = new SqlConnection(connectionString);
connection.Open();

// step 2 -- command (SQL)
SqlCommand command = new SqlCommand("Select * from Username", connection);

// step 3 -- data adapter
SqlDataAdapter sda = new SqlDataAdapter(command);
DataSet ds = new DataSet();
sda.Fill(ds); // filling dataset using adapter

connection.Close();

this.dataGridView1.DataSource = ds.Tables[0];
```

Execute the program and validate the results.



First_Name	Last_Name	City	State	Country	Nationality
Dilshad	Baidani	Gambat ...	Pakistan ...	Pakistan ...	Pakistani
Faizan	Bhatti	Lucknow ...	India ...	India ...	Indian

Installer Package Reference:
2013 and onwards : https://www.advancedinstaller.com/user-guide/tutorial-ai-ext-vs.html