MUHAMMAD ALI NASIK AWAN

023-20-0107

BSSE V

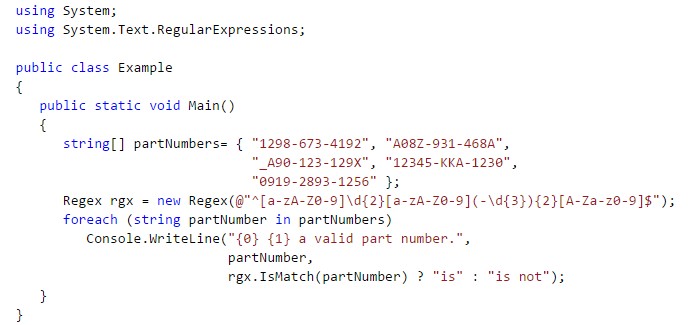
OBJECTIVES:

1. Understanding and working with regular expressions.
2. Understanding and working with File system.
3. Understanding and working with menus.
4. Understanding and working with RichTextBox.
5. Understanding and working with databases.
6. Practice activities.

**OBJECTIVE 1:** Understanding and working with regular expressions

# Regular Expressions

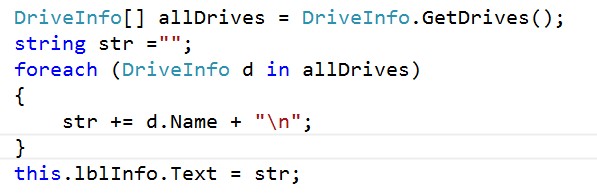
1. A regular expression is a pattern or set of rules that regular expression engine attempts to match in input text.
2. A pattern consists of one or more character literals, operators, or construct.
3. The Regex class is used for representing a regular expression.
4. Regular expressions provide a powerful, flexible, and efficient method for processing text.
5. [System.Text.RegularExpressions.Regex](https://msdn.microsoft.com/en-us/library/system.text.regularexpressions.regex(v=vs.110).aspx) is path for Regex existence.
6. At a minimum, processing text using regular expressions requires that the regular expression engine be provided with the following two items of information:
   1. The regular expression pattern to identify in the text.
   2. In the .NET Framework, regular expression patterns are defined by a special syntax or language, which is compatible with Perl 5 regular expressions and adds some additional features such as right-to-left matching
   3. The text to parse for the regular expression pattern.



**OBJECTIVE 2:** Understanding and working with File system.

# File System

* Accessing drives, directories and files on system.
* *System.IO.DriveInfo class* o This class models a drive and provides methods and properties to query for drive information.
  + Use DriveInfo to determine available drives
  + You can also query to determine the capacity and available free space on the drive.
  + The drive name must be either an uppercase or lowercase letter from 'a' to 'z'



* ***System.IO.DirectoryInfo class*** o Exposes instance methods for creating, moving, and enumerating through directories and subdirectories. o This class cannot be inherited.

* ***System.IO.FileInfo class*** o Provides properties and instance methods for the creation, copying, deletion, moving, and opening of files.
  + This class cannot be inherited.

**OBJECTIVE 3**: Understanding and working with Menus

# Menus

Menus provides a way to place multiple commands in less space in an application windows.

Menus can be fixed but can also be popped up at area where user right clicks the mouse.

Fixed menus are created by using MenuStrip class in windows forms applications.

MenuStrip is the top-level container that supersedes [MainMenu.](https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.mainmenu?redirectedfrom=MSDN&view=netframework-4.7.2)

[ToolStripDropDownItem](https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.toolstripdropdownitem?redirectedfrom=MSDN&view=netframework-4.7.2) and [ToolStripMenuItem](https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.toolstripmenuitem?redirectedfrom=MSDN&view=netframework-4.7.2) work along with MenuStrip for adding menu items in menu strip or bar.

Following classes can also be used with MenuStrip.

[ToolStripMenuItem](https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.toolstripmenuitem?view=netframework-4.7.2)

[ToolStripTextBox](https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.toolstriptextbox?view=netframework-4.7.2)

[ToolStripComboBox](https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.toolstripcombobox?view=netframework-4.7.2)

[ToolStripSeparator](https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.toolstripseparator?view=netframework-4.7.2)

Popup menus are created by using ContextMenuStrip class.

It represents a short cut menu.

One can associate a ContextMenuStrip with any control, and a right mouse click automatically displays the shortcut menu.

ContextMenuStrip supports images, menu-item check state, text, access keys, shortcuts, and cascading menus.

The following items are specifically designed to work seamlessly with ContextMenuStrip.

[ToolStripMenuItem](https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.toolstripmenuitem?view=netframework-4.7.2)

[ToolStripTextBox](https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.toolstriptextbox?view=netframework-4.7.2)

[ToolStripComboBox](https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.toolstripcombobox?view=netframework-4.7.2)

[ToolStripSeparator](https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.toolstripseparator?view=netframework-4.7.2)

Shortcut menus are typically used to combine different menu items from a MenuStrip of a form that are useful for the user given the context of the application.

**OBJECTIVE 4:** Understanding and working with Rich Text Box.

# Rich Text Box



Represents a Windows rich text box control.

The control also provides more advanced formatting features than the standard TextBox control.

Text can be assigned directly to the control, or can be loaded from a rich text format (RTF) or plain text file.

The text within the control can be assigned character and paragraph formatting.

To change the formatting of text, it must first be selected.

Only selected text can be assigned character and paragraph formatting.

Once a setting has been made to a selected section of text, all text entered after the selection is also formatted with the same settings until a setting change is made or a different section of the control's document is selected.

The SelectionFont property enables you to make text bold or italic or to change typeface & size.

The SelectionColor property enables you to change the color of the text.

To create bulleted lists you can use the SelectionBullet property.

The LoadFile method enables you to load an existing RTF or ASCII text file into the control.

The SaveFile enables you to save a file to RTF or ASCII text.

**OBJECTIVE 5:** Understanding and working with databases.

Database is collection of tables, that are collection of records and each record is collection of attributes of that record.

Databases are used as persistent storage mechanisms.

# Database Access

ADO.NET supports two types of data access.

o Connected data access o Disconnected data access

**SQLClient** namespace is imported for stated purpose.

Classes used to work with SQL Server.

* SqlConnection: for creating connection

Open() is used for opening of connection and Close() for closing the connection

* SqlCommand: for wrapping SQL query in object

CommandText = “query”

ExecuteReader() // for query operation

|  |  |
| --- | --- |
| VISUAL PROGRAMMING: LAB 7  ExecuteNonQuery() // for DML operation o SqlDataReader: provides forward access only on tables. Read() // read a record from recordset | 5 |

**ACTIVITIES SECTION**

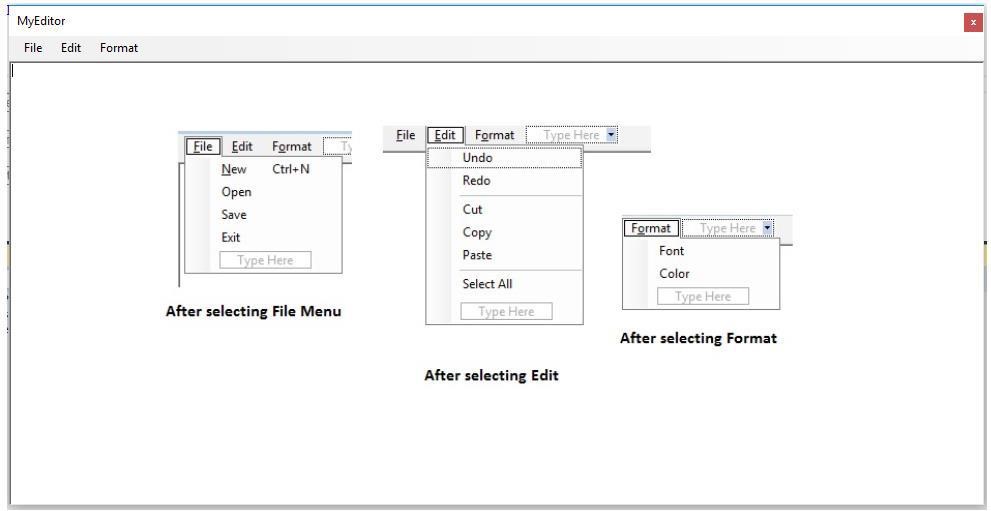
# ACITVITY 1: STEPS



Create a windows forms application named EditorApp.

Create windows form in it named frmEditor.

Create Graphical interface of frmEditor as given in picture below.



Use following instance methods of rich text box for relevant functionality.

Undo(); // for undo the last action

Redo(); // for again doing the last action

Cut(); // for copying data by removing selected data from source

Copy(); // for copying data by leaving source data as it is

Paste(); // for pasting copied or cut data

SelectAll(); // for selecting complete text of rich text box.

For Font use **FontDialog** and set **SelectionFont** property for selected text of rich text box.

For Color use ColorDialog and set SelectionColor property for selected text of rich text box.

For opening file use OpenDialog to get file name and after it use LoadFile(name\_of\_file) method of rich text box to load a file in it.

For saving file use same path of current opened file and use SaveFile(name\_of\_file) method of rich text box.

After performing required operations assign access keys and shortcut keys for every menu item in menu.

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 NotePad

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void newToolStripMenuItem1\_Click(object sender, EventArgs e)

{

richTextBox1.Clear();

}

private void openToolStripMenuItem1\_Click(object sender, EventArgs e)

{

OpenFileDialog dialogbox = new OpenFileDialog();

dialogbox.Title = "Open";

dialogbox.Filter = "Text Document(\*.txt)|\*.txt| All Files (\*.\*)|\*.\* ";

if(dialogbox.ShowDialog()== DialogResult.OK)

{

richTextBox1.LoadFile(dialogbox.FileName, RichTextBoxStreamType.PlainText);

this.Text = dialogbox.FileName;

}

}

private void saveToolStripMenuItem1\_Click(object sender, EventArgs e)

{

SaveFileDialog dialogbox = new SaveFileDialog();

dialogbox.Title = "Save";

dialogbox.Filter = "Text Document(\*.txt)|\*.txt| All Files (\*.\*)|\*.\* ";

if (dialogbox.ShowDialog() == DialogResult.OK)

{

richTextBox1.SaveFile(dialogbox.FileName, RichTextBoxStreamType.PlainText);

this.Text = dialogbox.FileName;

}

}

private void exitToolStripMenuItem1\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void undoToolStripMenuItem1\_Click(object sender, EventArgs e)

{

richTextBox1.Undo();

}

private void redoToolStripMenuItem1\_Click(object sender, EventArgs e)

{

richTextBox1.Redo();

}

private void copyToolStripMenuItem1\_Click(object sender, EventArgs e)

{

richTextBox1.Copy();

}

private void cutToolStripMenuItem1\_Click(object sender, EventArgs e)

{

richTextBox1.Cut();

}

private void pasteToolStripMenuItem1\_Click(object sender, EventArgs e)

{

richTextBox1.Paste();

}

private void selectAllToolStripMenuItem1\_Click(object sender, EventArgs e)

{

richTextBox1.SelectAll();

}

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

{

FontDialog test = new FontDialog();

if (test.ShowDialog() == DialogResult.OK)

{

richTextBox1.Font = test.Font;

}

}

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

{

ColorDialog test = new ColorDialog();

if (test.ShowDialog() == DialogResult.OK)

{

richTextBox1.ForeColor = test.Color;

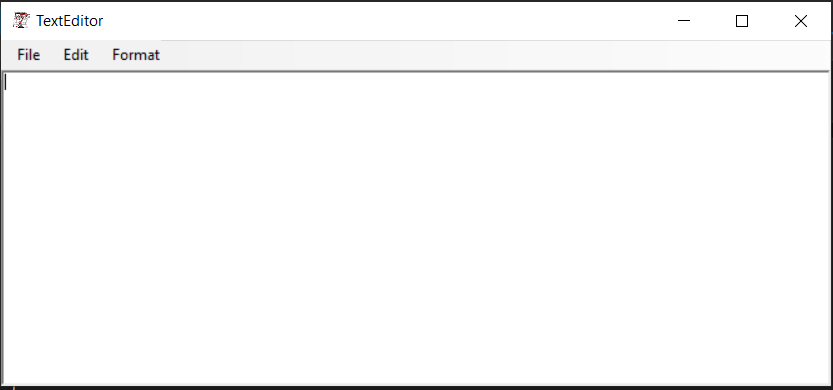
}

}

}

}

OUPUT:



# ACITVITY 2: STEPS



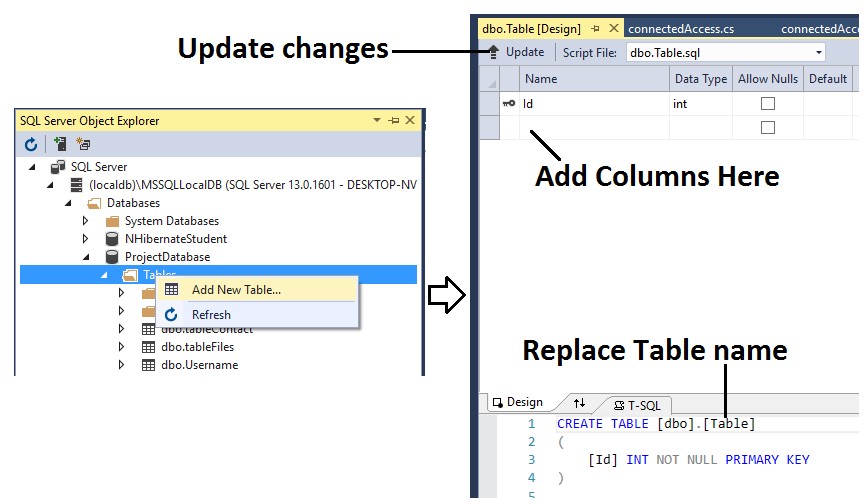
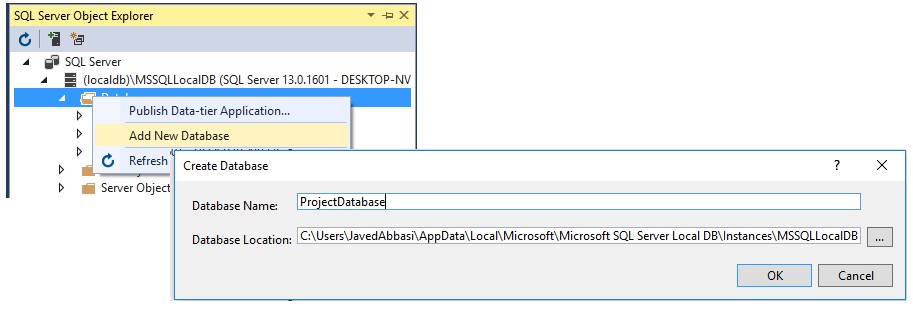
Create a windows forms application named **DatabaseApplication**.

Create a form named **frmConnectedAccess**.

Place label on this form named **lblInfo**.

**Setup resources for this example first**.

Open Sql Server Object Explorer from View Menu or by short cut CTRL+\ and CTRL + S

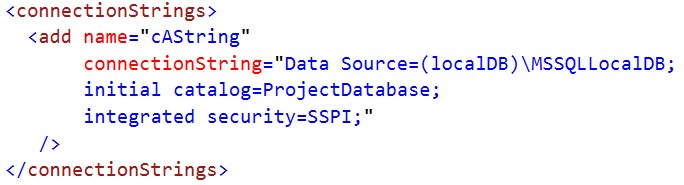




**Managing connection string now.**

Open app.config file.

Add following lines of code as child element of configuration element in xml file.





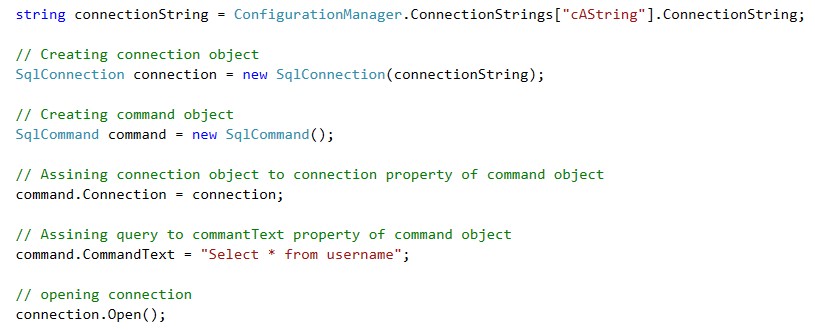
**Setup for classes used in this program.**

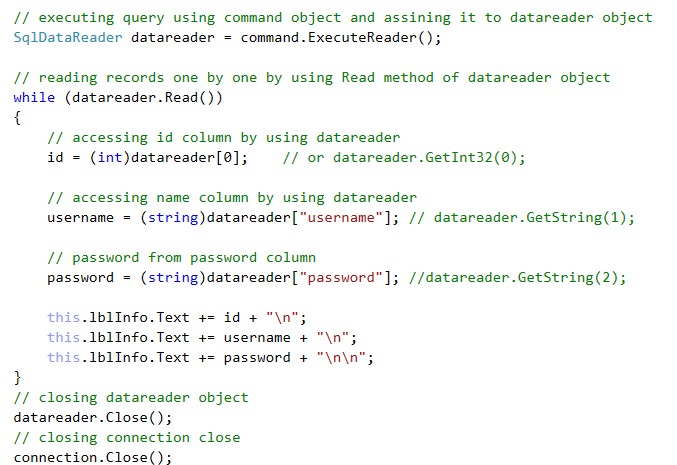
Import System.Data.SqlClient namespace in program by use of **using** keyword.

Import System.Configuration namespace by use of **using** keyword.

Now add a reference of System.Configuration namespace by right clicking on references option of project in solution explorer.

Create method named getUserData() and in its implementation place following code.







Execute the program and observe the results.

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.Data.SqlClient;

namespace DataBase

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

string connectionstring = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=master;Integrated Security=True;";

private void button1\_Click(object sender, EventArgs e)

{

//step 1 Create SQL Connection

SqlConnection connection = new SqlConnection();

connection.ConnectionString = connectionstring;

connection.Open();

//step 2 Create sql command

SqlCommand commands = new SqlCommand();

commands.Connection = connection;

// step 3 run quesries

commands.CommandText = @"SELECT \*FROM malik";

SqlDataReader datareader = commands.ExecuteReader();

textBox1.Text += "ID" + " " + "Name" + " " + "Password" + "\r\n";

while (datareader.Read()) {

String data = datareader.GetValue(0) + " " + datareader.GetValue(1) + " " + datareader.GetValue(2);

textBox1.Text += data + "\r\n";

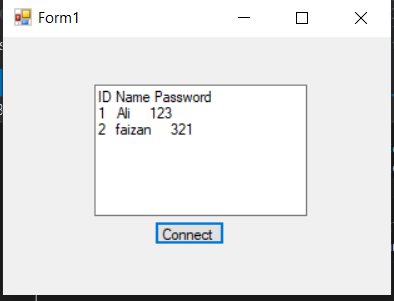
}

}

}

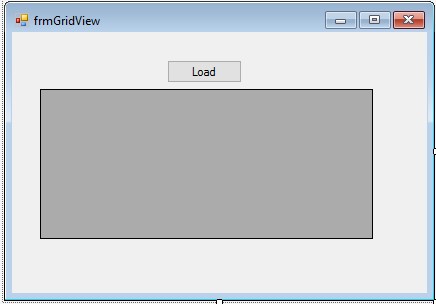
}

OUTPUT:



# ACITVITY 3: STEPS

* Create a form in application named frmGridView.
* Place button on this form.
* Place GridView on this form.



* Create a class named Person with following auto implemented properties.

o FirstNameo LastNameo City

* In Load event of form create an array of Person class with length 5.
* Populate array with objects of Person class and objects of Person class should also contain data of every property.
* Now in Click event of button assign array to DataSource property of GridView control.

**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 Lab\_07

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private person[] arr = new person[5];

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

{

dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

}

private void button1\_Click(object sender, EventArgs e)

{

arr[0] = new person("Malik", "Ali", "Talagang");

arr[1] = new person("Muhammad", "Faizan", "Sukkur");

arr[2] = new person("Dilshad", "Hussain", "Sukkur");

arr[3] = new person("Saud", "Ahmad", "Obaro");

arr[4] = new person("Muhammad", "Hussain", "Mir Pur Mathelo");

dataGridView1.DataSource = arr;

}

}

public class person

{

String First\_name;

String last\_name;

String city;

public person() { }

public person(String First\_name,

String last\_name,

String city)

{

this.first\_name = First\_name;

this.Last\_name = last\_name;

this.City = city;

}

public string first\_name

{

get

{

return First\_name;

}

set

{

First\_name = value;

}

}

public string Last\_name

{

get { return last\_name; }

set { last\_name = value; }

}

public string City

{

get { return city; }

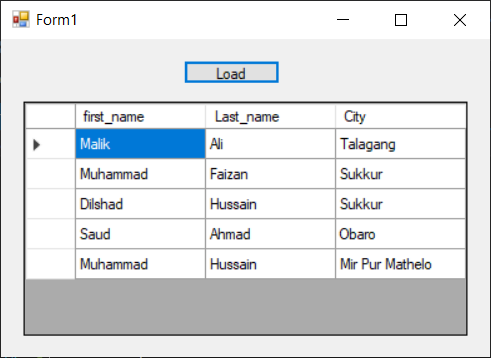
set { city = value; }

}

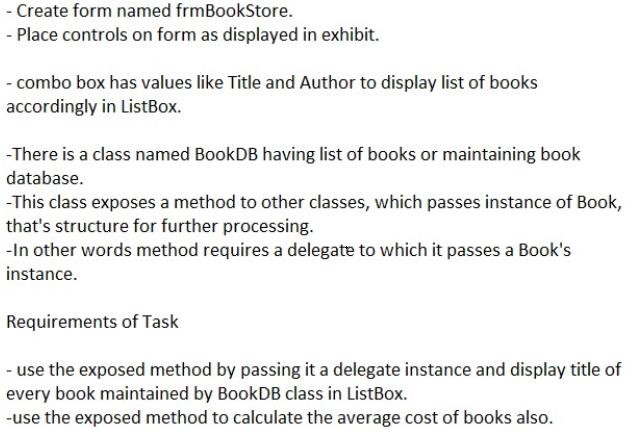
}

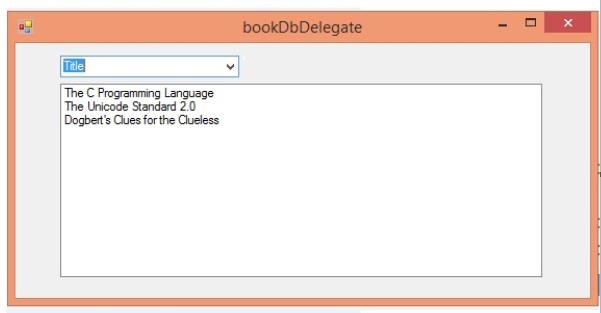
}

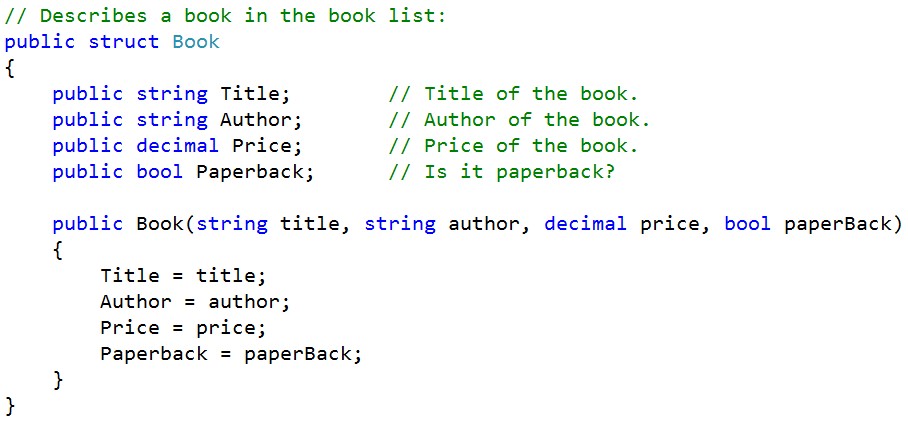
OUTPUT:

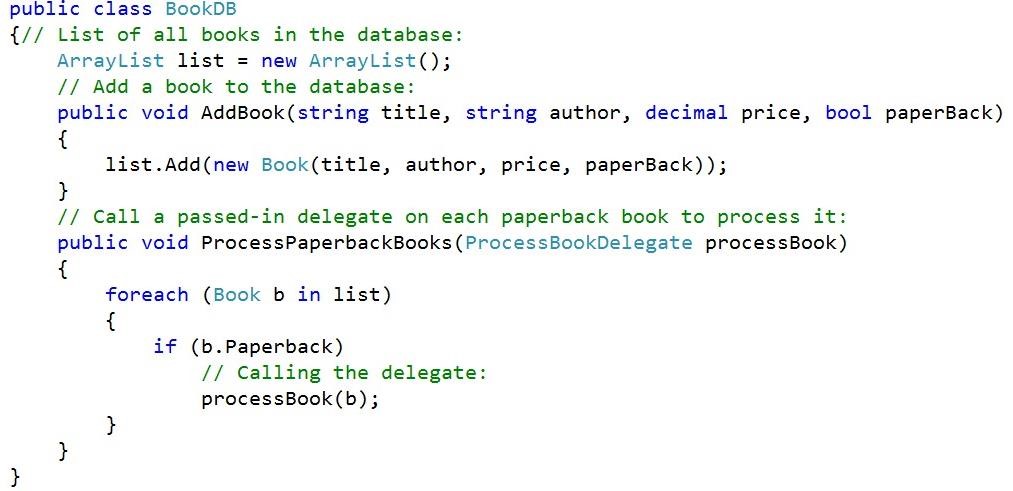


# ACITVITY 4: STEPS

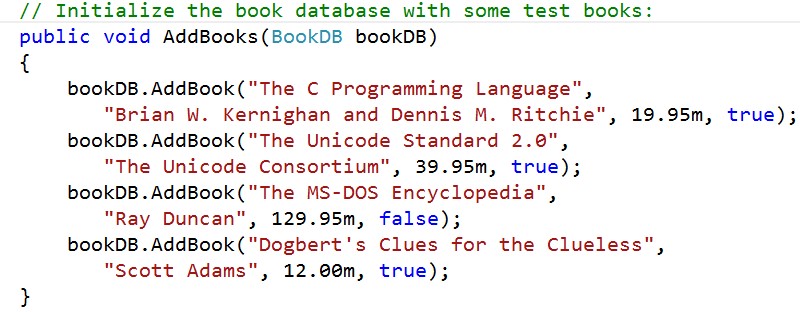








**S**



# 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;

using System.Collections;

namespace DB

{

public partial class Form1 : Form

{

public delegate void ProcessBookDelegate(book b);

ArrayList list;

public struct book

{

public string tittle;

public string author;

public decimal price;

public bool paperbook;

public book(string Tittle, string Author, decimal Price, bool Paperbook)

{

tittle = Tittle;

author = Author;

price = Price;

paperbook = Paperbook;

}

}

public Form1()

{

InitializeComponent();

BookDB book = new BookDB();

add(book);

list = book.Getter();

}

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

{

}

public void add(BookDB book)

{

book.addBook("The C programming Language", "Brian W.Kernighan and Dennis M.Ritchie", 19.95m, true);

book.addBook("The Unicode Standard 2.0","The Unicode Consortium",39.95m,true);

book.addBook("The MS-DOS encyclopedia","Ray Duncan",129.95m,true);

book.addBook("Dogberts Clues for the Clueless","Scott Adams",12.00m,true);

}

private void comboBox2\_SelectedIndexChanged(object sender, EventArgs e)

{

string str = comboBox2.SelectedItem.ToString();

listBox2.Items.Clear();

if (str == "Tittle")

for (int i = 0; i < list.Count; i++)

{

book book = (book)list[i];

listBox2.Items.Add(book.tittle);

}

else if (str == "Author")

for (int i = 0; i < list.Count; i++)

{

book book = (book)list[i];

listBox2.Items.Add(book.author);

}

}

}

public class BookDB

{

public string Title;

public string Author;

public decimal Price;

public bool Paperbook;

ArrayList list = new ArrayList();

public void addBook(string Tittle, string Author, decimal Price, bool Paperbook)

{

list.Add(new Form1.book(Tittle, Author, Price, Paperbook));

}

public void processpaperbook(Form1.ProcessBookDelegate processBook)

{

foreach(Form1.book b in list)

{

if (b.paperbook)

{

processBook(b);

}

}

}

public ArrayList Getter()

{

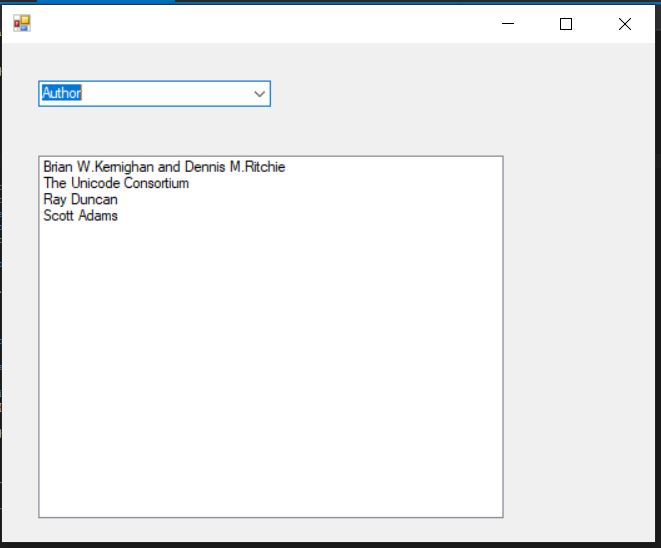
return list;

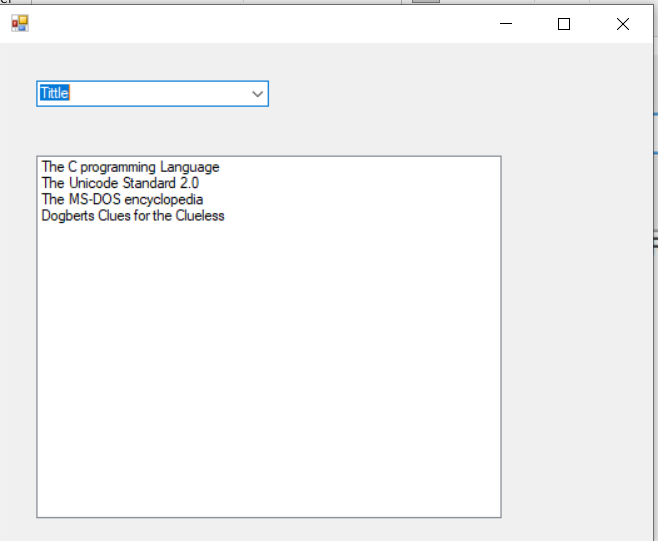
}

}

}

OUTPUT:



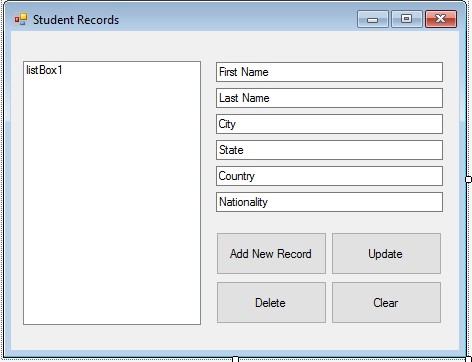


# ACITVITY 5: STEPS

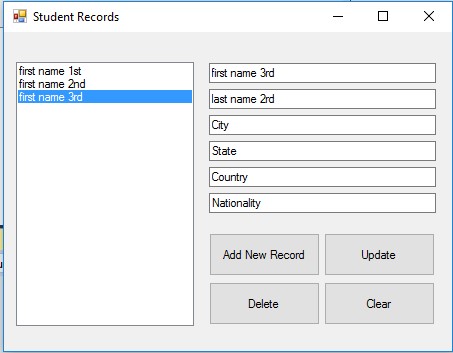
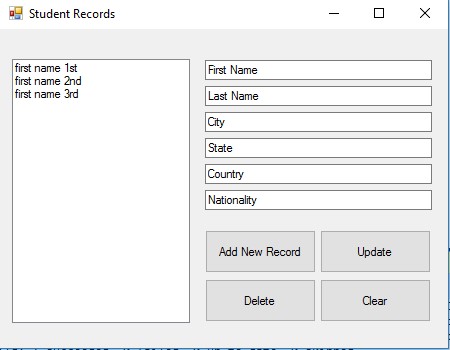


Create a windows forms application named StudentManagement.

Create interface of form according to given image. Before execution.



After Execution



[NOTE: Use **ExecuteNonQuery()** method of Command object for dml operations]

**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;

using System.Data.SqlClient;

namespace STDBMS

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

string connectionstring = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=STDDB;Integrated Security=True;";

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

{

//step 1 Create SQL Connection

SqlConnection connection = new SqlConnection();

connection.ConnectionString = connectionstring;

connection.Open();

//step 2 Create sql command

SqlCommand commands = new SqlCommand();

commands.Connection = connection;

// step 3 run quesries

commands.CommandText = @"SELECT \*FROM stdDB";

SqlDataReader datareader = commands.ExecuteReader();

while (datareader.Read())

{

listBox1.Items.Add(datareader.GetValue(0));

}

}

public void reset()

{

textBox1.Text = "First Name";

textBox2.Text = "Last Name";

textBox3.Text = "City";

textBox4.Text = "State";

textBox5.Text = "Country";

textBox6.Text = "Nationality";

}

private void button1\_Click(object sender, EventArgs e)

{

//step 1 Create SQL Connection

SqlConnection connection = new SqlConnection();

connection.ConnectionString = connectionstring;

connection.Open();

//step 2 Create sql command

SqlCommand commands = new SqlCommand();

commands.Connection = connection;

int i = 2;

// step 3 run quesries

commands.CommandType = CommandType.Text;

commands.CommandText = "INSERT into stdDB (First\_Name,Last\_Name,City,State,Country,Nationality) VALUES ('"+textBox1.Text+"','"+textBox2.Text+"','"+textBox3.Text+"','"+textBox4.Text+"','"+textBox5.Text+"','"+textBox6.Text+"')";

commands.ExecuteNonQuery();

MessageBox.Show("SuccessFully Updated");

listBox1.Items.Add(textBox1.Text);

reset();

i++;

connection.Close();

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

}

private void button4\_Click(object sender, EventArgs e)

{

reset();

}

private void button3\_Click(object sender, EventArgs e)

{

//step 1 Create SQL Connection

SqlConnection connection = new SqlConnection();

connection.ConnectionString = connectionstring;

connection.Open();

//step 2 Create sql command

SqlCommand commands = new SqlCommand("delete from stdDb where First\_Name=@First\_Name");

commands.Connection = connection;

commands.Parameters.AddWithValue("@First\_Name",textBox1.Text);

SqlDataAdapter da = new SqlDataAdapter(commands);

DataSet ds = new DataSet();

da.Fill(ds);

MessageBox.Show("Deleted Successfully");

connection.Close();

listBox1.Items.Remove(textBox1.Text);

}

private void button2\_Click(object sender, EventArgs e)

{

//step 1 Create SQL Connection

SqlConnection connection = new SqlConnection();

connection.ConnectionString = connectionstring;

connection.Open();

//step 2 Create sql command

SqlCommand commands = new SqlCommand();

commands.Connection = connection;

// step 3 run quesries

SqlCommand command = new SqlCommand("Update stdDb set First\_Name=@First\_Name , Last\_Name=@Last\_Name , City=@City , State=@State , Country=@Country , Nationality=@Nationality where First\_Name=@First\_Name");

command.Connection = connection;

command.Parameters.AddWithValue("@First\_Name", textBox1.Text);

command.Parameters.AddWithValue("@Last\_Name", textBox2.Text);

command.Parameters.AddWithValue("@City", textBox3.Text);

command.Parameters.AddWithValue("@State", textBox4.Text);

command.Parameters.AddWithValue("@Country", textBox5.Text);

command.Parameters.AddWithValue("@Nationality", textBox6.Text);

command.ExecuteNonQuery();

MessageBox.Show("Updated Successfully");

connection.Close();

}

private void listBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

//step 1 Create SQL Connection

SqlConnection connection = new SqlConnection();

connection.ConnectionString = connectionstring;

connection.Open();

//step 2 Create sql command

SqlCommand commands = new SqlCommand();

commands.Connection = connection;

// step 3 run quesries

commands.CommandText = @"SELECT \*FROM stdDB where First\_Name='"+listBox1.Text+"'";

SqlDataReader datareader = commands.ExecuteReader();

while (datareader.Read())

{

textBox1.Text =datareader.GetValue(0).ToString();

textBox2.Text = datareader.GetValue(1).ToString();

textBox3.Text = datareader.GetValue(2).ToString();

textBox4.Text = datareader.GetValue(3).ToString();

textBox5.Text = datareader.GetValue(4).ToString();

textBox6.Text = datareader.GetValue(5).ToString();

}

}

}

}

**OUTPUT:**