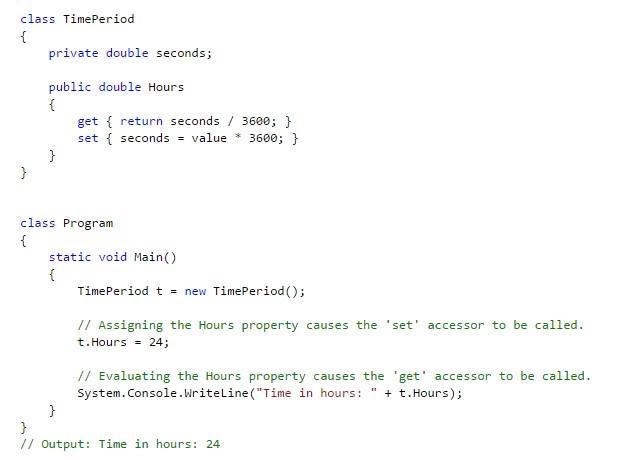
OBJECTIVES:

1. Understanding properties and working with them.
2. Understanding methods, extension methods and working with them.
3. Submitting data from one form to other and different ways to do that.
4. Understanding and working with picture box control.
5. Practice activities.

**OBJECTIVE 1:** Understanding properties and working with them.

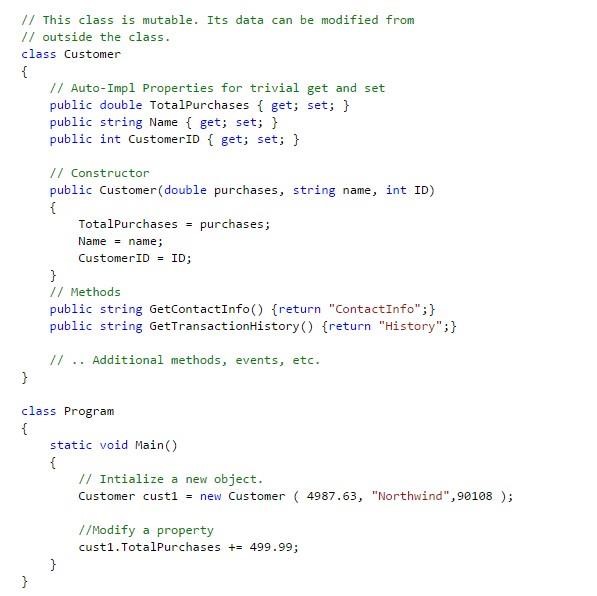
# Properties

* A property is a member that provides a flexible mechanism to read, write, or compute the value of a private field.
* Properties can be used as if they are public data members, but they are actually special methods called *accessors.*
* This enables data to be accessed easily and still helps promote the safety and flexibility of methods. For example:



o Properties enable a class to expose a public way of getting and setting values, while hiding implementation or verification code.

* A get property accessor is used to return the property value, and a set accessor is used to assign a new value.

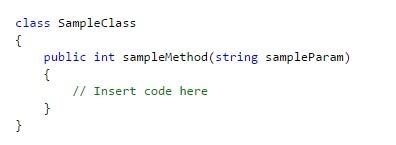


* The value keyword is used to define the value being assigned by the **set** accessor. o Properties that do not implement a **set** accessor are read only. **Auto implemented properties** o Auto-implemented properties make property-declaration more concise when no additional logic is required in the property accessors.
* When you declare a property as shown in the following example, the compiler creates a private, anonymous backing field that can only be accessed through the property's get and set accessors. o For example:

**OBJECTIVE 2:** Understanding methods, extension methods and working with them.

# Methods

A *method* is an action that an object can perform. To define a method of a class:

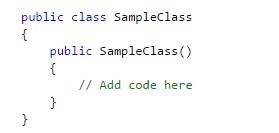


A class can have several implementations, or *overloads*, of the same method that differ in the number of parameters or parameter types. To overload a method:



# Constructors

* Constructors are class methods that are executed automatically when an object of a given type is created.
* Constructors usually initialize the data members of the new object.
* A constructor can run only once when a class is created.
* The code in the constructor always runs before any other code in a class.
* One can create multiple constructor overloads in the same way as for any other method.



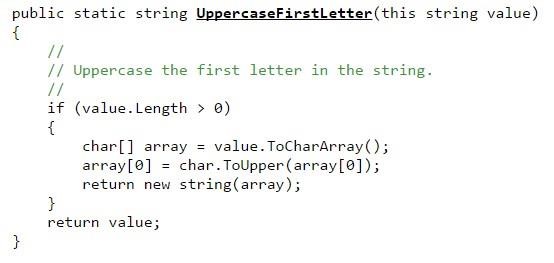
# Destructors

* Destructors are used to destruct instances of classes.
* The garbage collector automatically manages the allocation and release of memory for the managed objects in your application. However, you may still need destructors to clean up any unmanaged resources that your application creates.
* There can be only one destructor for a class.
* Destructor can be created using ~ sign in front of constructor name in constructor signature.

For example: ~Name\_Of\_A\_Class\_That\_is\_also\_name\_of\_Constructor()

# Extension methods

* An extension method has simplified calling syntax.
* It represents static methods as instance methods.
* An extension method uses this-keyword in its parameter list with first parameter.
* It must be located in a static class. For example:



**OBJECTIVE 3:** Submitting data from one form to other and different ways to do that.

* Create static fields in separate class that will hold values of first form, second form and so on.
* Create static fields in form1 that can be accessed in form2 by using name of form1.
* Constructor Approach: during creation of second form pass it data using its constructor.

**OBJECTIVE 4:** Understanding and working with picture box control.

Picture Box control

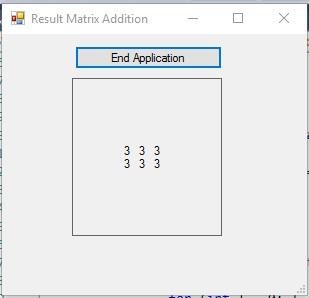
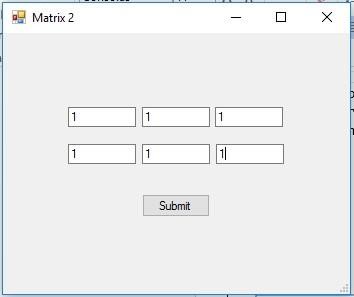
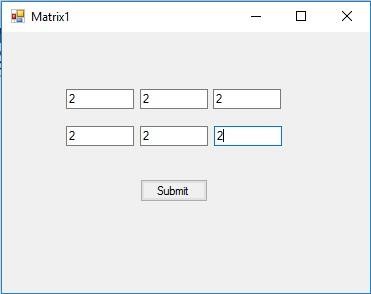
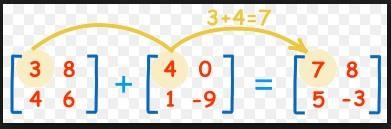
* Represents a Windows picture box control for displaying an image.
* The picture that is displayed is determined by the Image or ImageLocation property.

# ACTIVITIES SECTION ACTIVITY 1: STEPS

* Create a windows forms application project.
* Create three forms in project with names frmMatrix1, frmMatrix2, and frmMatrix3.

* Create 2X3 matrix of text boxes on first form.
* Place button named btnSubmit on first form for submitting the values.
* On click event of btnSubmit first form should be invisible and second form should be visible on screen.

* Write code in second form to hold data of first form passed to it.
* Create 2X3 matrix of text boxes on second form.
* Place button named btnSubmit on second form for submitting the values.
* On click event of btnSubmit second form should be invisible and third form should be visible on screen.
* Pass data of first form and of second form to third form.
* Write code in third form to get data of first and second form passed to it.
* In last addition of both matrices should be displayed on a third form. **[Note]: Use**
* Hide() method of form to hide visible form: for example: form1.Hide();  Used ShowDialog() to display next form: for example: object\_of\_second\_form.ShowDialog();
* BorderStyle=”FixedSingle” property of label to give it borders.
* Application.Exit() method to end the application.
* You can use arrays to hold values: o for example int[] array\_name = new int[3]; ando int[,] twoDArray=new int[2,3];



# Code:

Form1:

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 WindowsFormsApplication1

{

public partial class Form1 : Form

{

public int[,] matric = new int[2, 3];

public Form1()

{

InitializeComponent();

}

public void add()

{

matric[0, 0] = Convert.ToInt32(textBox1.Text);

matric[0, 1] = Convert.ToInt32(textBox2.Text);

matric[0, 2] = Convert.ToInt32(textBox3.Text);

matric[1, 0] = Convert.ToInt32(textBox4.Text);

matric[1, 1] = Convert.ToInt32(textBox5.Text);

matric[1, 2] = Convert.ToInt32(textBox6.Text);

}

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

{

add();

this.Hide();

secondform form = new secondform();

form.ShowDialog();

}

}

}

Form2:

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 WindowsFormsApplication1

{

public partial class secondform : Form

{

public int[,] matrix = new int[2, 3];

public int[,] result = new int[2, 3];

Form1 form = new Form1();

thirdform test = new thirdform();

public secondform()

{

InitializeComponent();

}

public void ADD()

{

matrix[0, 0] = Convert.ToInt32(textBox1.Text);

matrix[0, 1] = Convert.ToInt32(textBox2.Text);

matrix[0, 2] = Convert.ToInt32(textBox3.Text);

matrix[1, 0] = Convert.ToInt32(textBox4.Text);

matrix[1, 1] = Convert.ToInt32(textBox5.Text);

matrix[1, 2] = Convert.ToInt32(textBox6.Text);

}

public void Addition()

{

result[0,0]=form.matric[0, 0] + matrix[0, 0];

result[0, 1] = form.matric[0, 1] + matrix[0, 1];

result[0, 2] = form.matric[0, 2] + matrix[0, 2];

result[1, 0] = form.matric[1, 0] + matrix[1, 0];

result[1, 1] = form.matric[1, 1] + matrix[1, 1];

result[1, 2] = form.matric[1, 2] + matrix[1, 2];

}

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

{

ADD();

this.Hide();

test.ShowDialog();

Addition();

test.addresult();

}

}

}

Form3:

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 WindowsFormsApplication1

{

public partial class thirdform : Form

{

public thirdform()

{

InitializeComponent();

}

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

{

secondform sample = new secondform();

label1.Text = sample.result[0, 0].ToString();

label2.Text = sample.result[0, 1].ToString();

label3.Text = sample.result[0, 2].ToString();

label4.Text = sample.result[1, 0].ToString();

label5.Text = sample.result[1, 1].ToString();

label6.Text = sample.result[1, 2].ToString();

}

public void addresult()

{

secondform sample = new secondform();

label1.Text = sample.result[0,0].ToString();

label2.Text = sample.result[0, 1].ToString();

label3.Text = sample.result[0, 2].ToString();

label4.Text = sample.result[1, 0].ToString();

label5.Text = sample.result[1, 1].ToString();

label6.Text = sample.result[1, 2].ToString();

}

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

{

Application.Exit();

label1.Text = "";

label2.Text = "";

label3.Text = "";

label4.Text = "";

label5.Text = "";

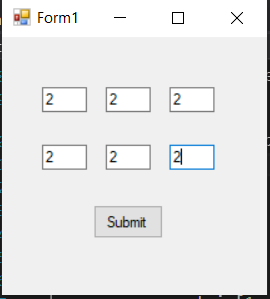
label6.Text = "";

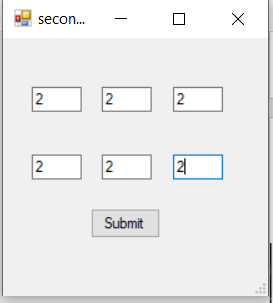
}

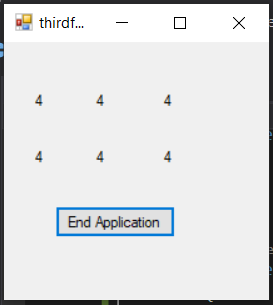
}

}

OUTPUT:



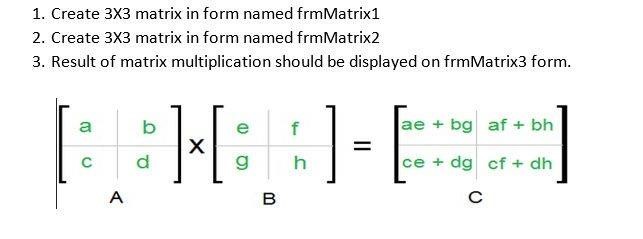




# ACTIVITY 2: STEPS

Create windows forms application named matrixMultiplication.

Number of columns in first matrix must be equal to number of rows in second



Code:

Form 1:

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 Matricmulti

{

public partial class Form1 : Form

{

public int[,] matrix\_1 = new int[3, 3];

public Form1()

{

InitializeComponent();

}

public void add()

{

matrix\_1[0, 0] = Convert.ToInt32(textBox1.Text);

matrix\_1[0, 1] = Convert.ToInt32(textBox2.Text);

matrix\_1[0, 2] = Convert.ToInt32(textBox3.Text);

matrix\_1[1, 0] = Convert.ToInt32(textBox4.Text);

matrix\_1[1, 1] = Convert.ToInt32(textBox5.Text);

matrix\_1[1, 2] = Convert.ToInt32(textBox6.Text);

matrix\_1[2, 0] = Convert.ToInt32(textBox7.Text);

matrix\_1[2, 1] = Convert.ToInt32(textBox8.Text);

matrix\_1[2, 2] = Convert.ToInt32(textBox9.Text);

}

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

{

add();

formpart2 test = new formpart2();

test.adda(matrix\_1);

this.Hide();

test.ShowDialog();

}

}

}

Form 2:

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 Matricmulti

{

public partial class formpart2 : Form

{

int[,] mat = new int[3, 3];

int[,] a = new int[3, 3];

int[,] result = new int[3, 3];

public formpart2()

{

InitializeComponent();

}

public void multiplication()

{

result[0, 0] = (a[0, 0] \* mat[0, 0]) + (a[0, 1] \* mat[1, 0]) + (a[0, 2] + mat[2, 0]);

result[0, 1] = (a[0, 0] \* mat[0, 1]) + (a[0, 1] \* mat[1, 1]) + (a[0, 2] + mat[2, 1]);

result[0, 2] = (a[0, 0] \* mat[0, 2]) + (a[0, 1] \* mat[1, 2]) + (a[0, 2] + mat[2, 2]);

result[1, 0] = (a[1, 0] \* mat[0, 0]) + (a[1, 1] \* mat[1, 0]) + (a[1, 2] + mat[2, 0]);

result[1, 1] = (a[1, 0] \* mat[0, 1]) + (a[1, 1] \* mat[1, 1]) + (a[1, 2] + mat[2, 1]);

result[1, 2] = (a[1, 0] \* mat[0, 2]) + (a[1, 1] \* mat[1, 2]) + (a[1, 2] + mat[2, 2]);

result[2, 0] = (a[2, 0] \* mat[0, 0]) + (a[2, 1] \* mat[1, 0]) + (a[2, 2] + mat[2, 0]);

result[2, 1] = (a[2, 0] \* mat[0, 1]) + (a[2, 1] \* mat[1, 1]) + (a[2, 2] + mat[2, 1]);

result[2, 2] = (a[2, 0] \* mat[0, 2]) + (a[2, 1] \* mat[1, 2]) + (a[2, 2] + mat[2, 2]);

}

public void adda(int[,] form1)

{

Console.WriteLine(form1);

this.a = form1;

}

public void mix()

{

mat[0, 0] = Convert.ToInt32(textBox1.Text);

mat[0, 1] = Convert.ToInt32(textBox2.Text);

mat[0, 2] = Convert.ToInt32(textBox3.Text);

mat[1, 0] = Convert.ToInt32(textBox4.Text);

mat[1, 1] = Convert.ToInt32(textBox5.Text);

mat[1, 2] = Convert.ToInt32(textBox6.Text);

mat[2, 0] = Convert.ToInt32(textBox7.Text);

mat[2, 1] = Convert.ToInt32(textBox8.Text);

mat[2, 2] = Convert.ToInt32(textBox9.Text);

}

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

{

mix();

part3 task = new part3();

multiplication();

task.addresult(result);

this.Hide();

task.ShowDialog();

}

}

}

Form 3:

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 Matricmulti

{

public partial class part3 : Form

{

int[,] result = new int[3, 3];

public part3()

{

InitializeComponent();

}

public void addresult(int[,] s)

{

this.result = s;

}

public void addLabel(int[,] set)

{

label1.Text = set[0, 0].ToString();

label2.Text = set[0, 1].ToString();

label3.Text = set[0, 2].ToString();

label4.Text = set[1, 0].ToString();

label5.Text = set[1, 1].ToString();

label6.Text = set[1, 2].ToString();

label7.Text = set[2, 0].ToString();

label8.Text = set[2, 1].ToString();

label9.Text = set[2, 2].ToString();

}

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

{

Application.Exit();

}

private void part3\_Load(object sender, EventArgs e)

{

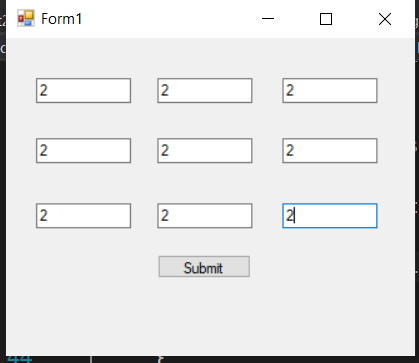
addLabel(result);

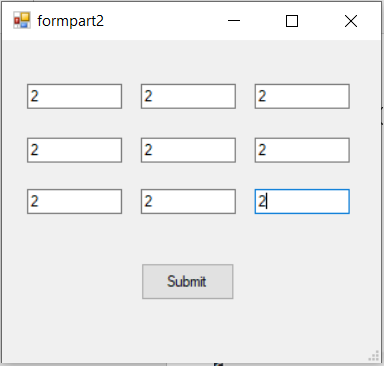
}

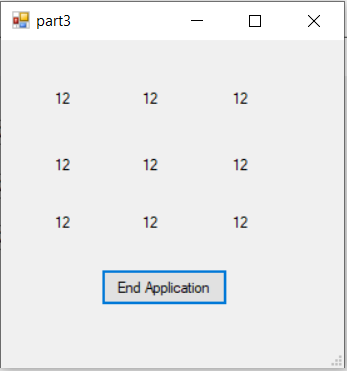
}

}

Output:









 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 pictureShow

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

string pic = "D:\\vp\\LABS\\Lab-04\\pictureShow\\wallpapers\\";

int count = 1;

int prev;

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

{

}

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

{

this.Dispose();

}

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

{

pictureBox1.ImageLocation = pic + count + ".jpg";

count++;

prev = count - 1;

if (count > 5)

{

MessageBox.Show("No More Pics !");

}

}

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

{

pictureBox1.ImageLocation = pic + prev + ".jpg";

prev--;

if (count < 1)

{

MessageBox.Show("No More Pics !");

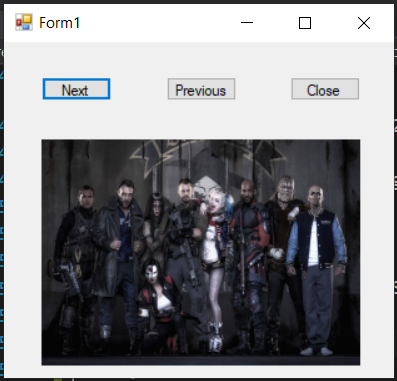
}

}

}

}

Output:





















# ACTIVITY 4: STEPS

* Create a windows forms application named PuzzleGame.
* Create a form named frmPuzzle.
* Place following controls on form.
* All controls are buttons with names described in picture. Write code according to conditions given below.
* Initially set picture into btnPic button using Image property of button.
* Display given information into the btnInfo button using its Text property.

* Set 16 images into rest of 16 buttons out of those 3 buttons should have the same image as of btnPic button.
* Initially user will not be able to see the images of 16 buttons.
* Total incorrect attempts allowed to user are 3.
* Game must be terminated if user has reached to incorrect attempts.
* Game must be terminated if user has guessed the puzzle correctly.

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 guessthepic

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

static Random rnd = new Random();

int num = rnd.Next(1, 6);

int set = rnd.Next(1, 6);

int i = 3;

int button;

int win\_cont = 0;

public void reset()

{

button1.Image = null;

button2.Text = "GUESS PICTURE THREE TIMES FROM PUZZLE (Attempts Allowed = 3)";

button3.Image = null;

button4.Image = null;

button5.Image = null;

button6.Image = null;

button7.Image= null;

button8.Image = null;

button9.Image = null;

button10.Image = null;

button11.Image = null;

button12.Image = null;

button13.Image = null;

button14.Image = null;

button15.Image = null;

button16.Image = null;

button17.Image = null;

button18.Image = null;

i = 3;

}

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

{

button = num;

this.button1.Image = Image.FromFile(@"D:\vp\LABS\Lab-04\guessthepic\pics\"+num+ ".jfif");

num = rnd.Next(1, 6);

}

public void game\_click(object sender, EventArgs e)

{

Button spn = (Button)sender;

spn.Image = Image.FromFile(@"D:\vp\LABS\Lab-04\guessthepic\pics\" + button + ".jfif");

spn.Click -= new EventHandler(function\_click);

button2.Text = "GUESS PICTURE THREE TIMES FROM PUZZLE (Attempts Allowed = " + i + ")";

win\_cont++;

if (win\_cont == 3)

{

MessageBox.Show("You Won!");

reset();

}

}

public void function\_click(object sender , EventArgs e)

{

Button btn = (Button)sender;

i--;

btn.Image = Image.FromFile(@"D:\vp\LABS\Lab-04\guessthepic\pics\" + set + ".jfif");

set = rnd.Next(1, 6);

btn.Click -= new EventHandler(function\_click);

button2.Text = "GUESS PICTURE THREE TIMES FROM PUZZLE (Attempts Allowed = " + i + ")";

if (i == 0)

{

MessageBox.Show("Game Over");

reset();

}

}

}

}

OUTPUT:

