OBJECTIVES:

1. Understanding code snippets, code snippet manager and creating code snippets.
2. Understanding documentation and creating documentation.
3. Understanding debugging and performing debugging in visual studio 2015.
4. Understanding and developing windows forms applications.
5. Practice Activities.

<u>OBJECTIVE 1:</u> Understanding code snippets, code snippet manager and creating code snippets.

Code Snippets

- ⇨ Snippets are pre-built code chunks/ lines of code used for rapid application development.
- ⇨ One can use already created snippets or can also create their own snippets.
- ⇨ There are two ways to add snippets.
    - Insert snippet: inline insertion of snippet.
    - Surround snippet: snippet that surrounds lines of code by selecting them and adding a snippet.

- For inserting code snippet, choose **Edit**, **IntelliSense**, **and Insert Snippet**

-OR-

- Open the shortcut menu in a file and choose **Insert Snippet**.

-OR-

- One can insert code snippet by writing snippet keyword/shortcut and pressing two times tab key.

- For example: if + tab + tab will insert if code snippet/chunk. **-OR-**
- Pressing CTRL + K + X will open a pop up menu for insertion of snippet.

-OR-

- Pressing CTRL + K + S will open a pop up menu for insertion of surround snippet.

OBJECTIVE 2: Understanding documentation and creating documentation.

Documentation

- ⇨ It is description of types and type members of system software created to understand the code written for the project.
- ⇨ For developing readable documentation three steps should be performed.
- ⇨ At first step, one should write comments in xml elements for the desired lines of code using special comment field starts with ///.
- ⇨ For example: writing ///<summary> comments goes here </summary>
- ⇨ At second step one should create an xml file from the xml comments created for code.
- ⇨ XML file can be created mainly in two ways:
  - o Using compiler: using \doc option for c-sharp compiler.
  - o Using visual studio: by checking documentation checkbox in properties of project.

- ⇨ At third step one should use a tool like **sandcastle and sandcastle help file builder** to generate a readable file from xml file created in second step.

- ⇨ XML file can also be created using third party extensions.
- ⇨ For example: using GhostDoc extension of visual studio that can insert xml comment elements automatically.

- ⇨ Pressing **CTRL + SHIFT + D** key combination will automatically insert xml comments in the code in case of extension (**GhostDoc**) that is installed with visual studio.

- ⇨ With the help of same extension xml file and documentation can also be created.

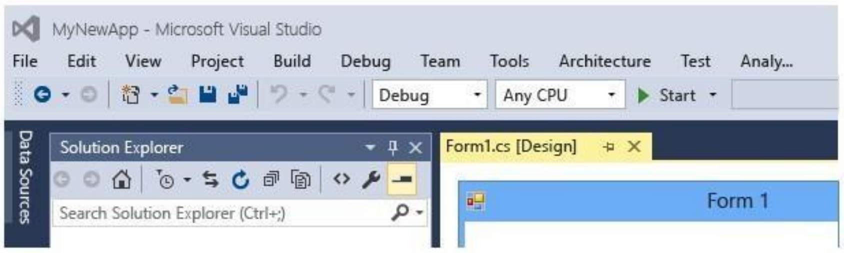
RECOMMENDED XML TAGS FOR DOCUMENTATION

- ⇨ <summary> Summary </summary> o The <summary> tag should be used to describe a type or a type member.
- ⇨ <returns> Description </returns> o A description of the return value.
- ⇨ <param name=”name”> Description </param> o Name: name of a method parameter. Enclose the name in double quotation marks (" ") o Description: A description for the parameter.
- ⇨ <value> Property Description</value> o The <value> tag lets you describe the value that a property represents

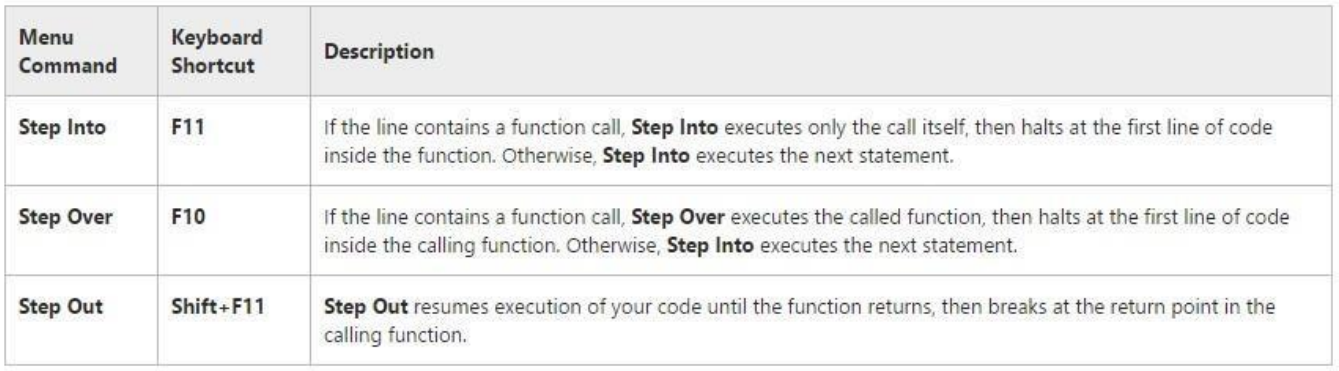
OBJECTIVE 3: Understanding debugging and performing debugging in visual studio 2015.

Debugging It is finding errors and removing/resolving that errors from code. ⇨ There are two types of build configuration: Debug and Release.

- ⇨ Debug: produces a slower, larger executable that allows for a richer interactive run-time debugging experience but should not be shipped.
- ⇨ In debug mode one can stop or pause the execution and execute the code in step wise manner by navigating consequences of every operation.
- ⇨ The default build configuration is **Debug**.
- ⇨ Release: builds a faster, more optimized executable that's appropriate to ship (at least from the perspective of the compiler).


MyNewApp - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Architecture Test Analy...
Debug
Any CPU
Start
Data Sources
Solution Explorer
Search Solution Explorer (Ctrl+;)
Form1.cs [Design]
Form 1

- ⇨ For debugging code one can insert break points in a code.
- ⇨ A breakpoint indicates where Visual Studio should suspend your running code so you can take a look at the values of variables, or the behavior of memory, or whether or not a branch of code is getting run. You do NOT need to rebuild a project after setting and removing breakpoints.
- ⇨ Along with breakpoints, **trace points** are a new debugger feature in Visual Studio. A trace point is a breakpoint with a custom action associated with it. When a trace point is hit, the debugger performs the specified trace point action instead of, or in addition to, breaking program execution.
- ⇨ For inserting break point select line of code and right click for inserting break point.
- ⇨ After inserting desired number of break points one can execute code in debugging mode.
- ⇨ Following shortcuts can be used for stepping into code and so on.



Menu Command	Keyboard Shortcut	Description
Step Into	**F11**	If the line contains a function call, **Step Into** executes only the call itself, then halts at the first line of code inside the function. Otherwise, **Step Into** executes the next statement.
Step Over	**F10**	If the line contains a function call, **Step Over** executes the called function, then halts at the first line of code inside the calling function. Otherwise, **Step Into** executes the next statement.
Step Out	**Shift+F11**	**Step Out** resumes execution of your code until the function returns, then breaks at the return point in the calling function.

<u>OBJECTIVE 4:</u> Understanding and developing windows forms applications.

Windows Forms Applications

Windows Forms is the platform for developing windows applications based on .NET framework.

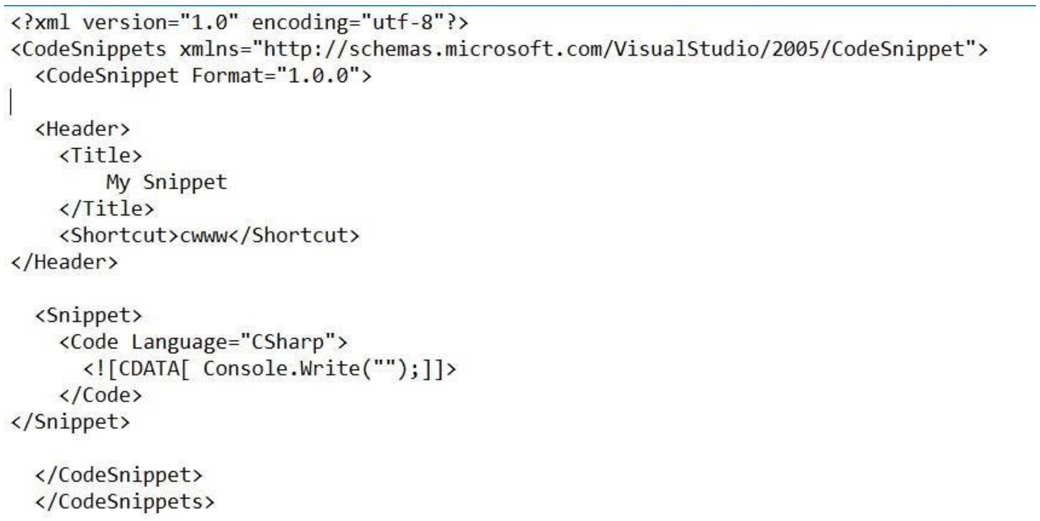
In .NET development, a Windows GUI application is called a Windows Forms (or *Winforms*) application.

Form

- ⇨ A form is a bit of screen, usually rectangular, that you can use to present information to the user and to accept input from the user.
- ⇨ Forms can be standard windows, multiple document interface windows, dialog boxes, or display surfaces for graphical routines.
- ⇨ Forms are objects that expose properties which define their appearance, methods which define their behavior, and events which define their interaction with the user.
- ⇨ By setting the properties of the form and writing code to respond to its events, you customize the object to meet the requirements of your application.
- ⇨ The form you create with the Windows Forms Designer is a class, and when you display an instance of the form at run time, this class is the template used to create the form.
- ⇨ Additionally, forms are controls, because they inherit from the Control class.

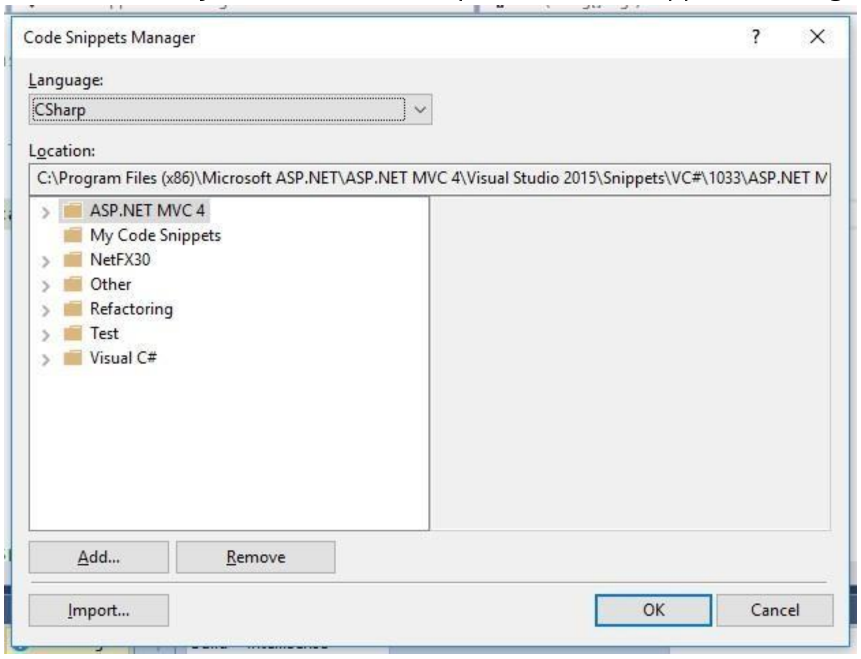
ACTIVITIES SECTION OBJECTIVE 1: ACTIVITIES
Understanding and creation of code snippets **ACTIVITY 1:**
STEPS

1. Create console application.
2. Create an xml file with extension .snippet.
3. Add following lines of code in the file.



```
<?xml version="1.0" encoding="utf-8"?>
<CodeSnippets xmlns="http://schemas.microsoft.com/VisualStudio/2005/CodeSnippet">
  <CodeSnippet Format="1.0.0">

  <Header>
    <Title>
        My Snippet
    </Title>
    <Shortcut>cwww</Shortcut>
</Header>

  <Snippet>
    <Code Language="CSharp">
      <![CDATA[ Console.Write("");]]>
    </Code>
</Snippet>

  </CodeSnippet>
  </CodeSnippets>
```

4. Open code snippets manager from Tools > Code Snippets Manager. -OR-
5. CTRL + K + B keys combination opens Code Snippets Manager directly.



Code Snippets Manager

Language: CSharp

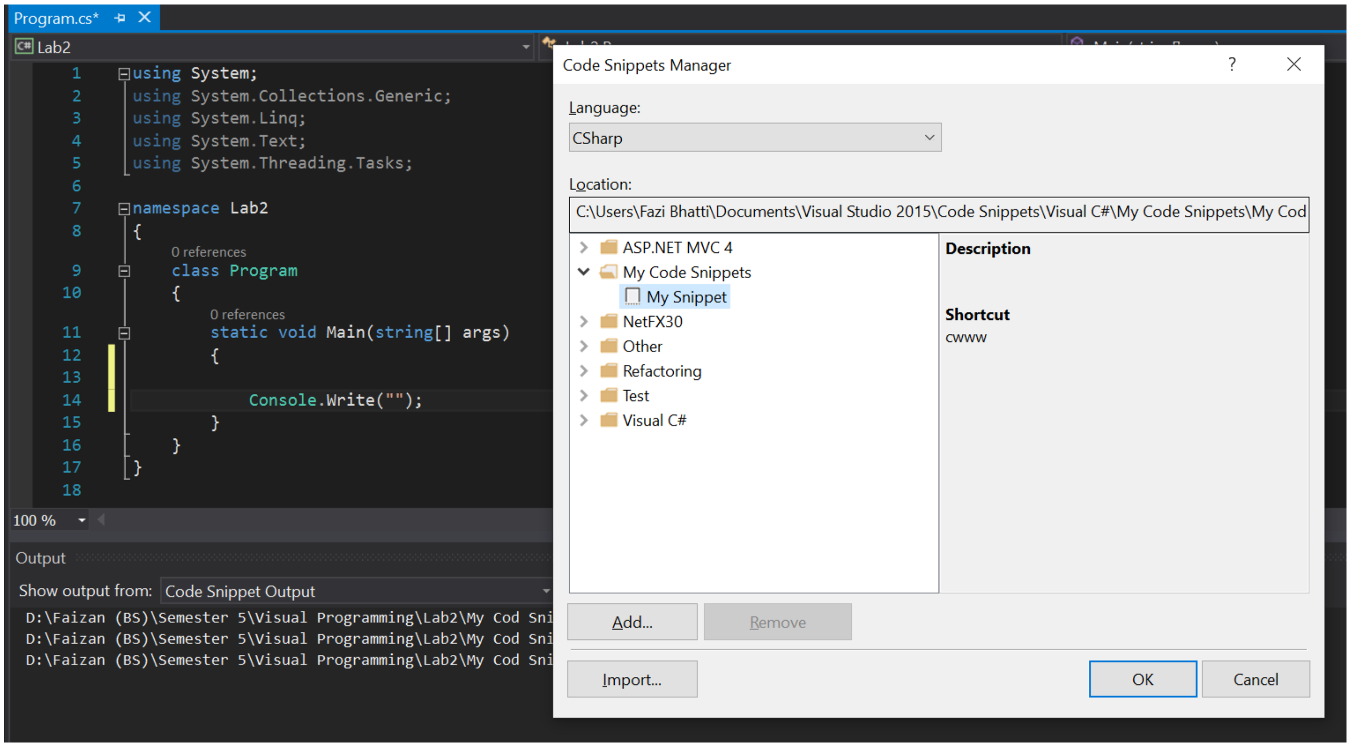
Location: C:\Program Files (x86)\Microsoft ASP.NET\ASP.NET MVC 4\Visual Studio 2015\Snippets\VC#\1033\ASP.NET M

- ASP.NET MVC 4
- My Code Snippets
- NetFX30
- Other
- Refactoring
- Test
- Visual C#

6. Using Code snippets manager one can remove the existing and can add the new code snippets.

7. Select Visual C# from folders.
8. Press Import button.
9. Import the snippet that you have created, select location for it, and use the snippet in the code by typing cwww + two time tab key.

Solution:



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab2
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {

            Console.Write("");
        }
    }
}
```

Show output from: Code Snippet Output

```
D:\Faizan (BS)\Semester 5\Visual Programming\Lab2\My Cod Sni
D:\Faizan (BS)\Semester 5\Visual Programming\Lab2\My Cod Sni
D:\Faizan (BS)\Semester 5\Visual Programming\Lab2\My Cod Sni
```

Code Snippets Manager

Language: CSharp

Location: C:\Users\Fazi Bhatti\Documents\Visual Studio 2015\Code Snippets\Visual C#\My Code Snippets\My Cod

- ASP.NET MVC 4
- My Code Snippets
  - My Snippet
- NetFX30
- Other
- Refactoring
- Test
- Visual C#

Description

Shortcut
cwww

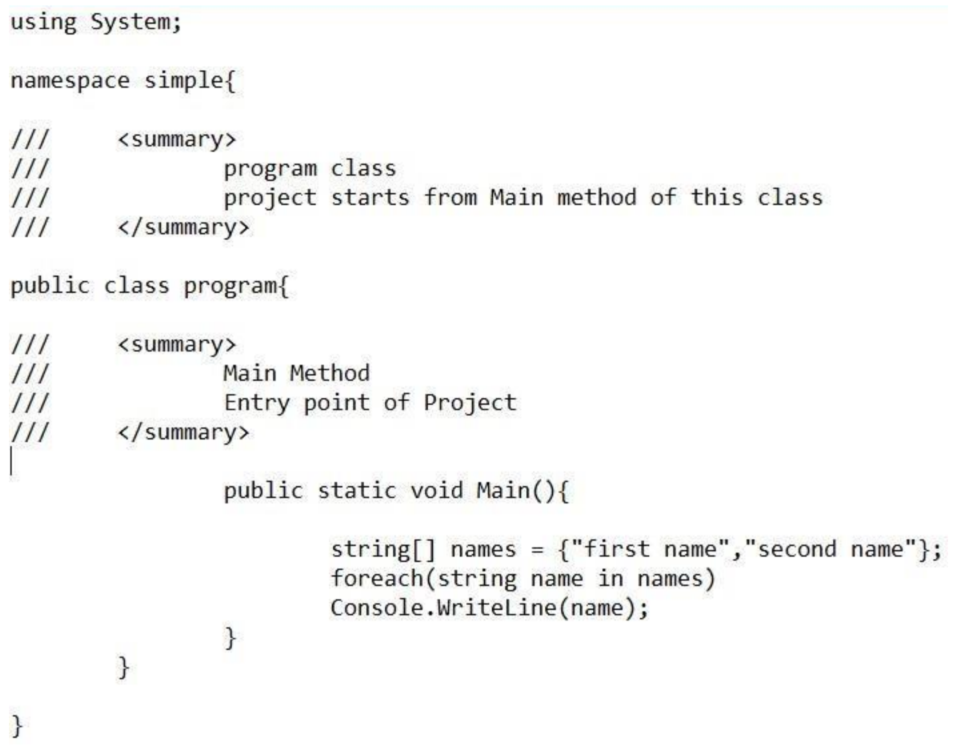
OBJECTIVE 2: ACTIVITIES

Understanding documentation and creating documentation.

ACTIVITY 2

STEPS

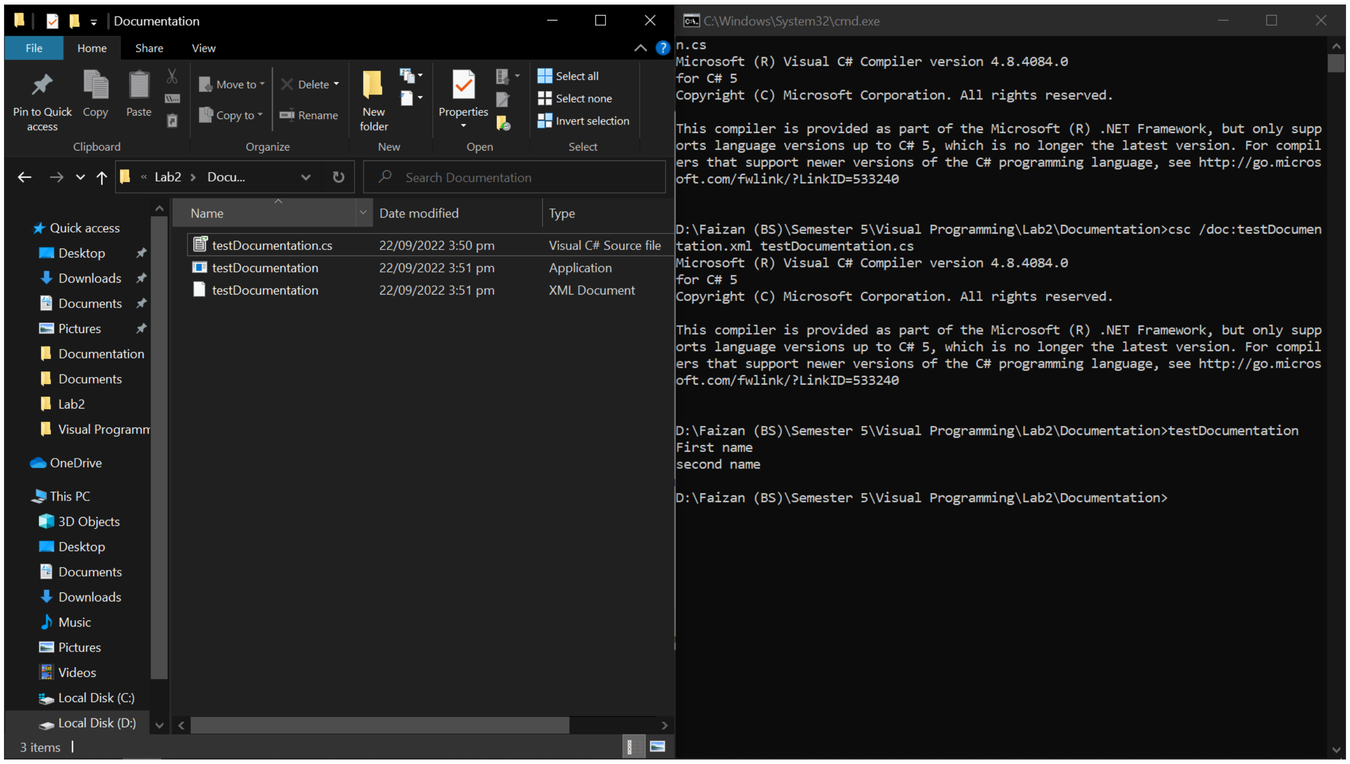
1. Create a Folder named Documentation
2. Create a file named testDocumentation.cs 3. Open it with notepad.
4. Write following code in it.



```
using System;

namespace simple{

///     <summary>
///             program class
///             project starts from Main method of this class
///     </summary>

public class program{

///     <summary>
///             Main Method
///             Entry point of Project
///     </summary>

                public static void Main(){

                        string[] names = {"first name","second name"};
                        foreach(string name in names)
                        Console.WriteLine(name);
                }
        }

}
```

5. /// is special field comment to write documentation comments for code.
6. Compile code using c-sharp compiler into the same folder.
7. Documentation>csc /doc:name_of_the_file.xml name_of_the_file.cs
8. /doc is a flag that is providing extra information to c-sharp compiler for generating documentation of code.
9. Observe the results after compiling an xml file must be created into the folder Documentation with the name specified in command.
10. Now we can create .chm file or any other file from this xml file using tools like sand castle.

Solution:



Name	Date modified	Type
testDocumentation.cs	22/09/2022 3:50 pm	Visual C# Source file
testDocumentation	22/09/2022 3:51 pm	Application
testDocumentation	22/09/2022 3:51 pm	XML Document

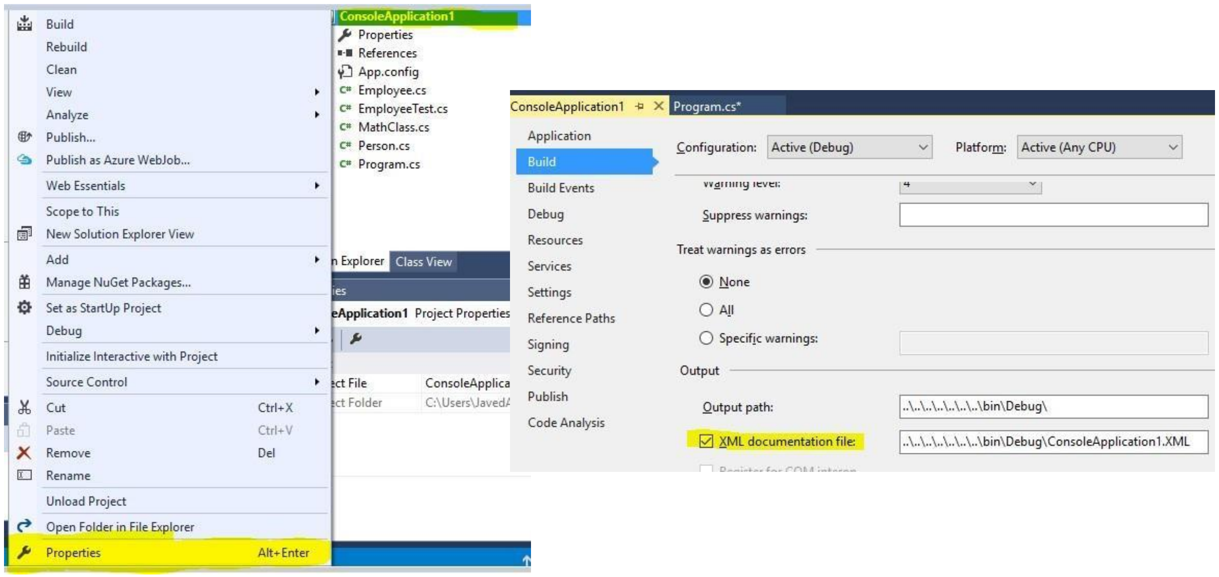
```
n.cs
Microsoft (R) Visual C# Compiler version 4.8.4084.0
for C# 5
Copyright (C) Microsoft Corporation. All rights reserved.

This compiler is provided as part of the Microsoft (R) .NET Framework, but only supp
orts language versions up to C# 5, which is no longer the latest version. For compil
ers that support newer versions of the C# programming language, see http://go.micros
oft.com/fwlink/?LinkID=533240


D:\Faizan (BS)\Semester 5\Visual Programming\Lab2\Documentation>csc /doc:testDocumen
tation.xml testDocumentation.cs
Microsoft (R) Visual C# Compiler version 4.8.4084.0
for C# 5
Copyright (C) Microsoft Corporation. All rights reserved.

This compiler is provided as part of the Microsoft (R) .NET Framework, but only supp
orts language versions up to C# 5, which is no longer the latest version. For compil
ers that support newer versions of the C# programming language, see http://go.micros
oft.com/fwlink/?LinkID=533240


D:\Faizan (BS)\Semester 5\Visual Programming\Lab2\Documentation>testDocumentation
First name
second name

D:\Faizan (BS)\Semester 5\Visual Programming\Lab2\Documentation>
```

ACTIVITY 3 STEPS

1. Create console application named ConsoleApplication1.
2. Write xml comments for documentation in default code generated by visual studio like in the file given below.

```
/// <summary>
///     program class
///     project starts from Main method of this class
/// </summary>
class Program
{
    /// <summary>
    ///     Main Method
    ///     Entry point of Project
    /// </summary>

    static void Main(string[] args)...

}
```

3. Go to the properties of a project by right clicking the project name that is ConsoleApplication1.
4. Select Build tab and check the XML documentation file checkbox for creating xml file.



Configuration: Active (Debug) Platform: Active (Any CPU)

Suppress warnings:

Treat warnings as errors

None [✓] All Specific warnings:

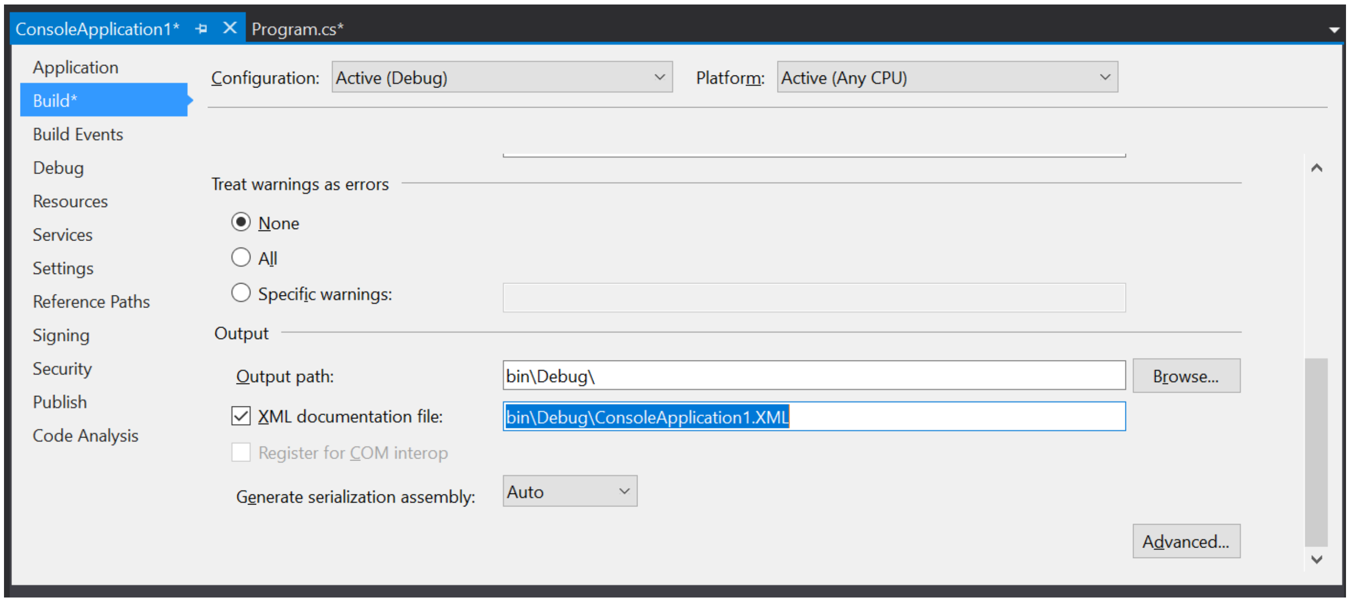
Output

Output path: ..\..\..\..\..\..\bin\Debug\

[✓] XML documentation file: ..\..\..\..\..\..\bin\Debug\ConsoleApplication1.XML

5. After the build operation now xml file will automatically be created in to the destination folder.

Solution:



Configuration: Active (Debug) Platform: Active (Any CPU)

Treat warnings as errors

[✓ None] All Specific warnings:

Output

Output path: bin\Debug\

[✓] XML documentation file: bin\Debug\ConsoleApplication1.XML

Register for COM interop

Generate serialization assembly: Auto

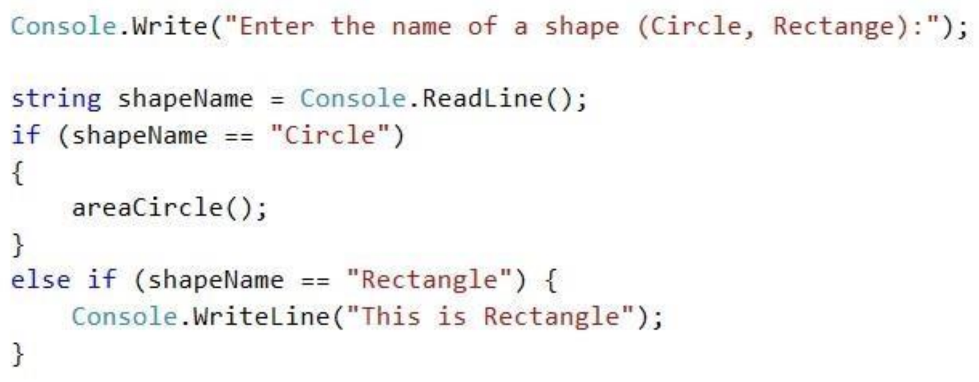
OBJECTIVE 3: ACTIVITIES

Understanding debugging and performing debugging in visual studio 2015.

ACTIVITY 4: STEPS

⇨ Create a console application.

⇨ Write following code in Main method.



```
Console.Write("Enter the name of a shape (Circle, Rectange):");

string shapeName = Console.ReadLine();
if (shapeName == "Circle")
{
    areaCircle();
}
else if (shapeName == "Rectangle") {
    Console.WriteLine("This is Rectangle");
}
```

⇨ Create static method named areaCircle() with any lines of code.

⇨ Press F9 by standing at if statement this will add a break point at this line.

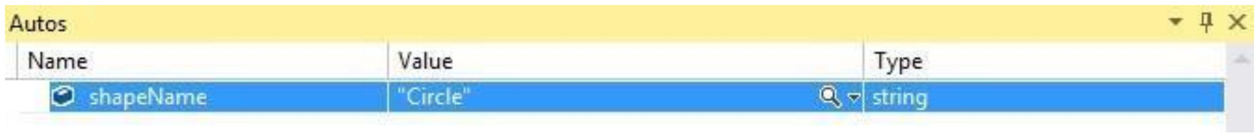
⇨ Press F9 by standing at areaCircle() method call.

⇨ Press F5 to execute application in Debug Mode.

⇨ At runtime provide argument “Circle” in command prompt.

⇨ Now observe following windows in visual studio.

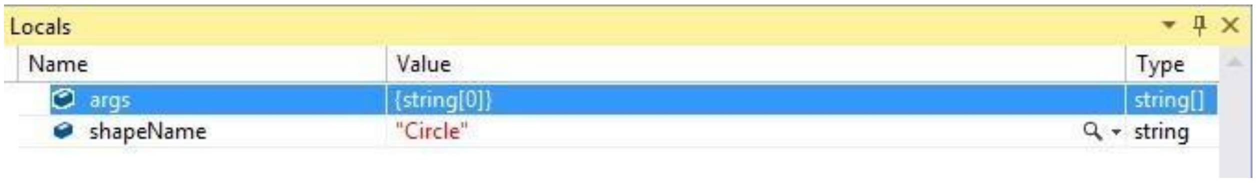
⇨ Autos window: shows values of variables and their type



Autos

Name	Value	Type
shapeName	"Circle"	string

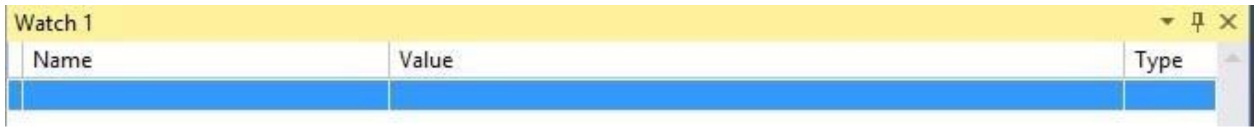
⇨ Locals window: shows values of local variables.



Locals

Name	Value	Type
args	{string[0]}	string[]
shapeName	"Circle"	string

⇨ Watch window: In this window, we can check values of local variables and also perform some other operations, like comparison of values and so on.

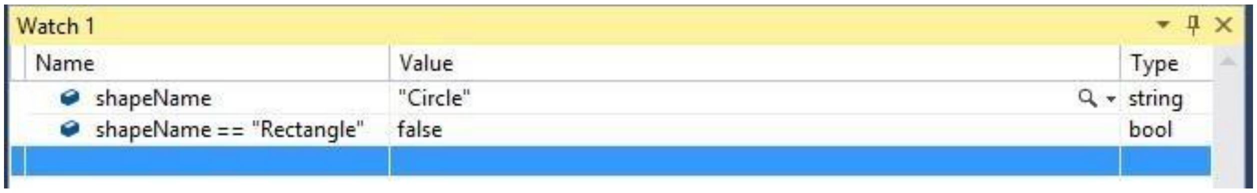


Watch 1

Name	Value	Type

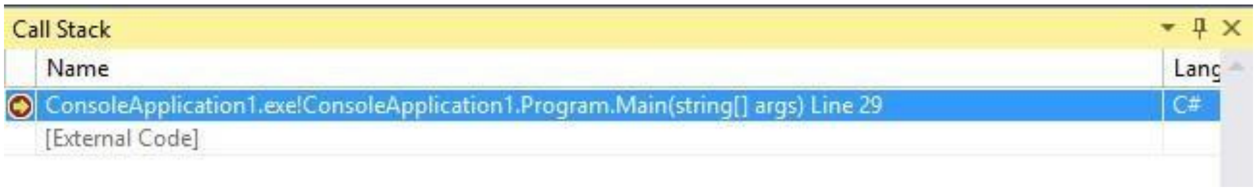
- At first there will be nothing in this watch window.

- Double click in the blank space below Name column to write something like name of variables and so on. ○ For example: As in given diagram two things are there:
- First: name of shapeName variable and value associated with it.
- Second: Usage of equality comparison operator on value of shapeName and its result in Value column.


Watch 1

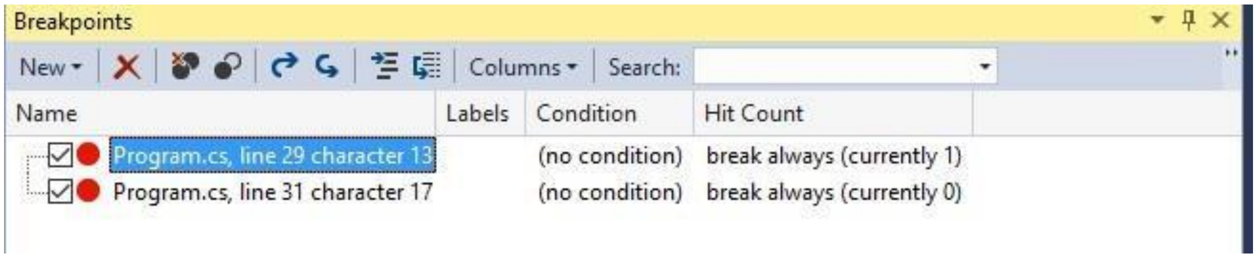
Name	Value	Type
shapeName	"Circle"	string
shapeName == "Rectangle"	false	bool

⇨ Call Stack window: this shows stack of method calls, their hierarchy and line number.


Call Stack

Name	Lang
ConsoleApplication1.exe!ConsoleApplication1.Program.Main(string[] args) Line 29	C#
[External Code]	

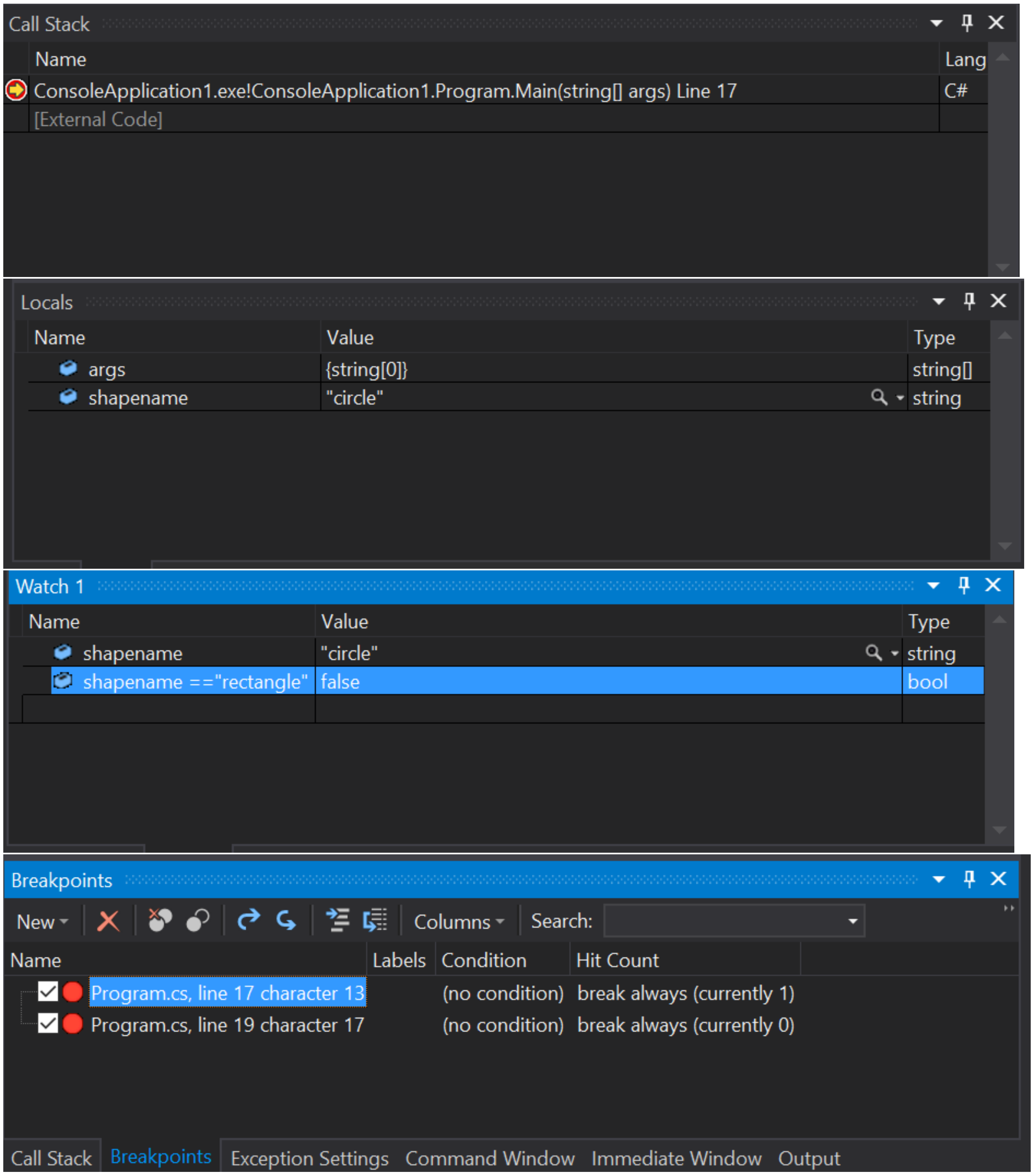
⇨ Break points window: shows list of break points in application along with labels, condition and hit count of break points.


Breakpoints

Name	Labels	Condition	Hit Count
Program.cs, line 29 character 13		(no condition)	break always (currently 1)
Program.cs, line 31 character 17		(no condition)	break always (currently 0)

⇨ **After observing all windows, Press Step into, Step over and Step out options and write notes on the functionality that you've observed.**

Solution:



Call Stack

Name	Lang
ConsoleApplication1.exe!ConsoleApplication1.Program.Main(string[] args) Line 17	C#
[External Code]	

Locals

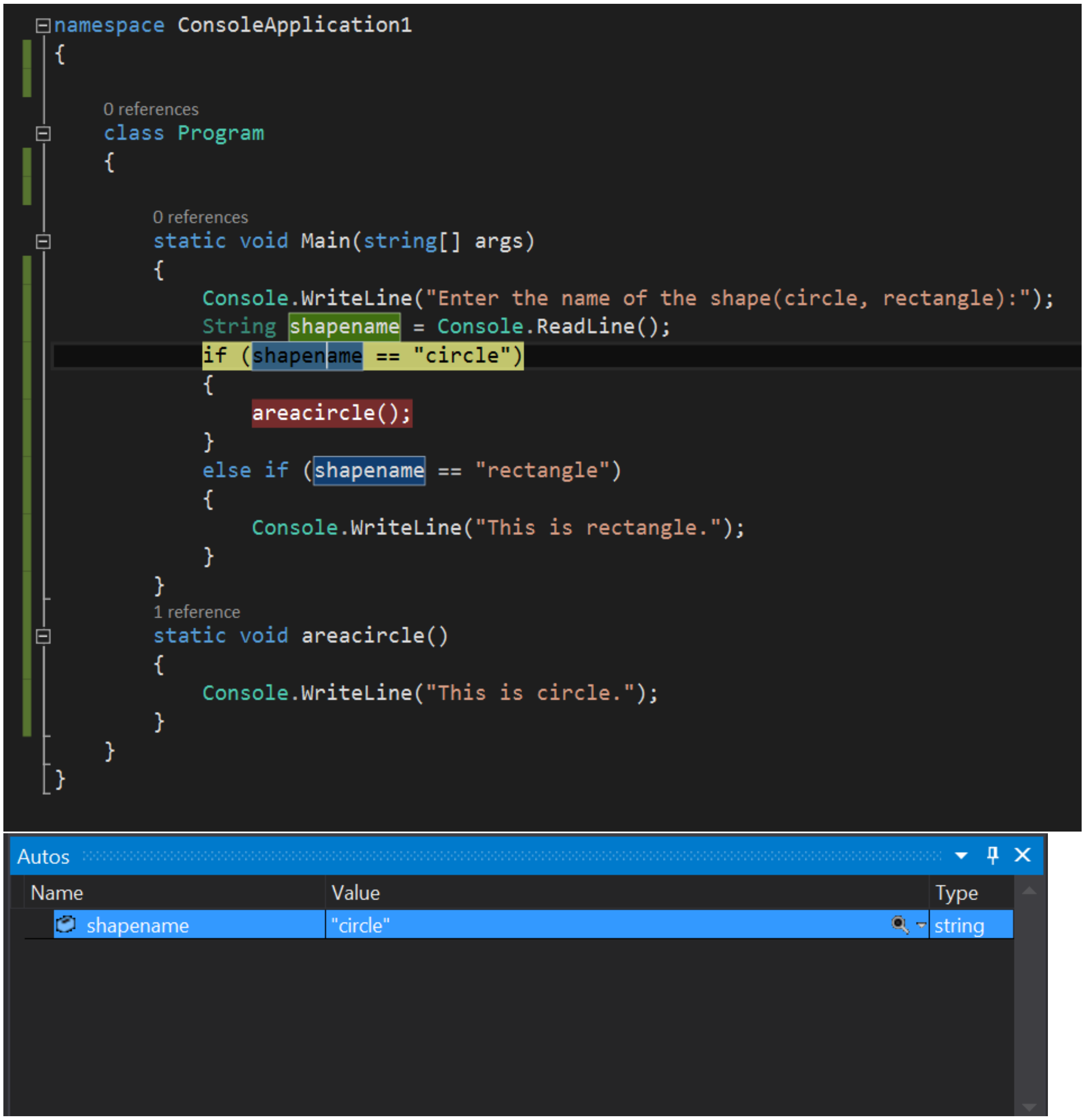
Name	Value	Type
args	{string[0]}	string[]
shapename	"circle"	string

Watch 1

Name	Value	Type
shapename	"circle"	string
shapename =="rectangle"	false	bool

Breakpoints

Name	Labels	Condition	Hit Count
✓ Program.cs, line 17 character 13		(no condition)	break always (currently 1)
✓ Program.cs, line 19 character 17		(no condition)	break always (currently 0)

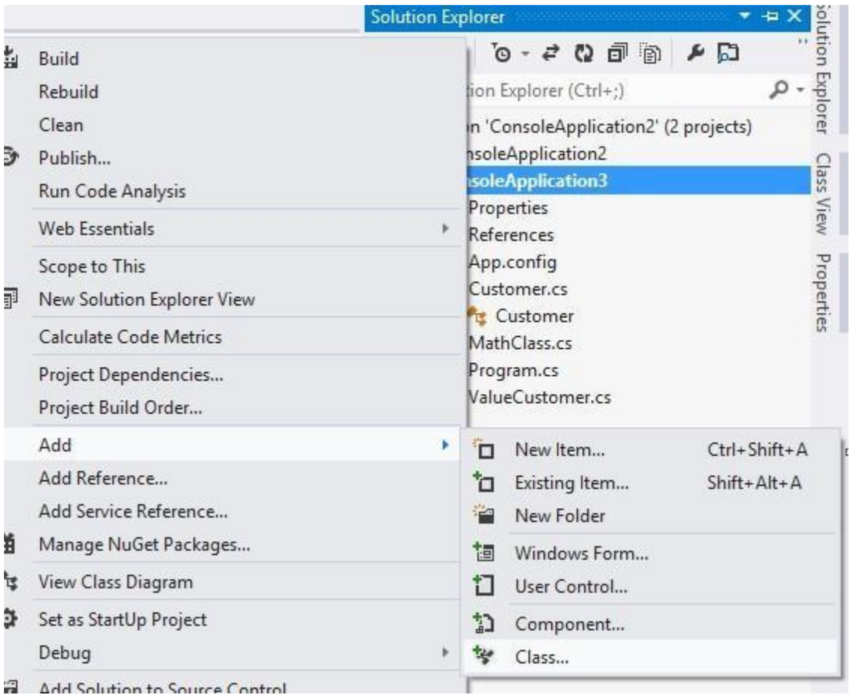


```
namespace ConsoleApplication1
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            Console.WriteLine("Enter the name of the shape(circle, rectangle):");
            String shapename = Console.ReadLine();
            if (shapename == "circle")
            {
                areacircle();
            }
            else if (shapename == "rectangle")
            {
                Console.WriteLine("This is rectangle.");
            }
        }
        1 reference
        static void areacircle()
        {
            Console.WriteLine("This is circle.");
        }
    }
}
```

Name	Value	Type
shapename	"circle"	string

ACTIVITY 5: THIS ACTIVITY IS RELATED TO FIRST LAB: STEPS

1. Create a console Application using visual studio 2012 or some other version if available.
2. Add new class named MathClass in application by right clicking on the name of console application in solution explorer.

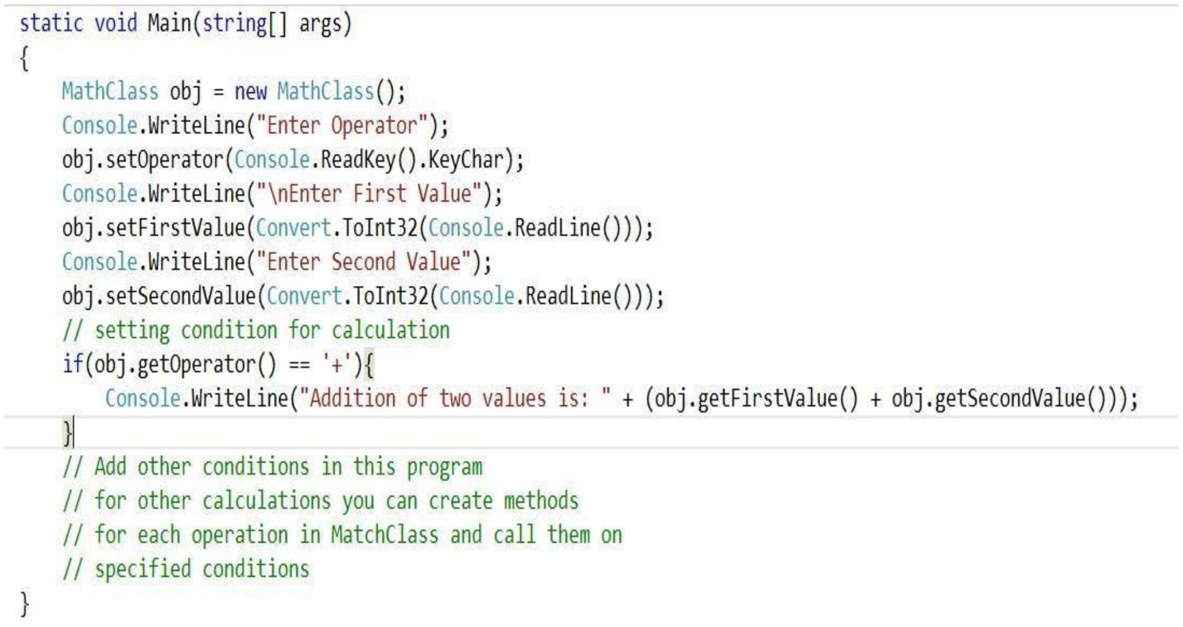


3. Create three variables in MathClass
   - String op
   - Int firstValue
   - Int secondValue

4. Create setter and getter methods for each of the variable like this one given in example.

```
public void setOperator(string op) {
    this.op = op;
}
public string getOperator() {
    return this.op;
}
```

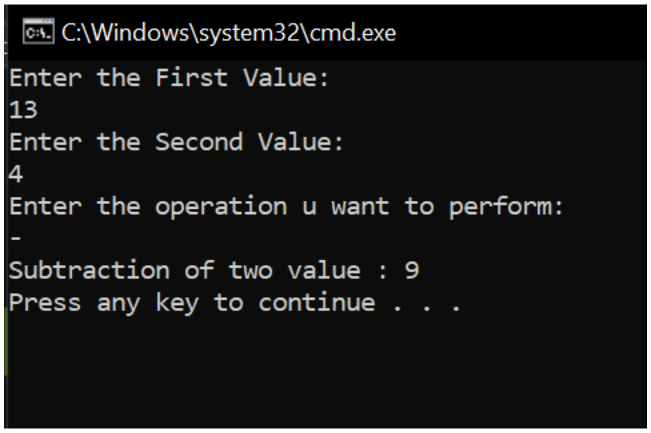
5. Here is a code of Main method in Program.cs class



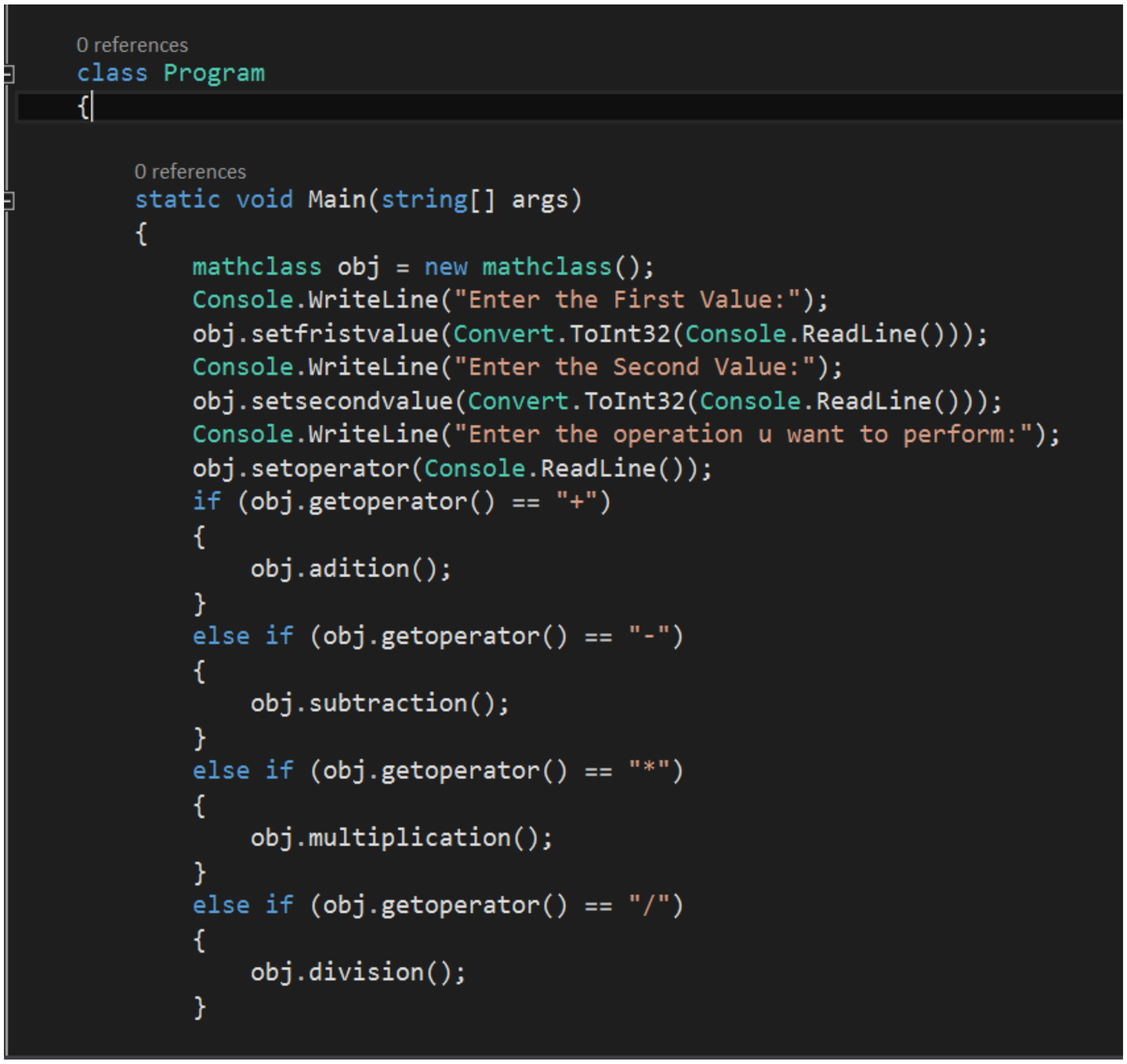
```
static void Main(string[] args)
{
    MathClass obj = new MathClass();
    Console.WriteLine("Enter Operator");
    obj.setOperator(Console.ReadKey().KeyChar);
    Console.WriteLine("\nEnter First Value");
    obj.setFirstValue(Convert.ToInt32(Console.ReadLine()));
    Console.WriteLine("Enter Second Value");
    obj.setSecondValue(Convert.ToInt32(Console.ReadLine()));
    // setting condition for calculation
    if(obj.getOperator() == '+'){
        Console.WriteLine("Addition of two values is: " + (obj.getFirstValue() + obj.getSecondValue()));
    }
    // Add other conditions in this program
    // for other calculations you can create methods
    // for each operation in MatchClass and call them on
    // specified conditions
}
```

6. Create Methods for other operations in the MathClass.
7. Execute the program.

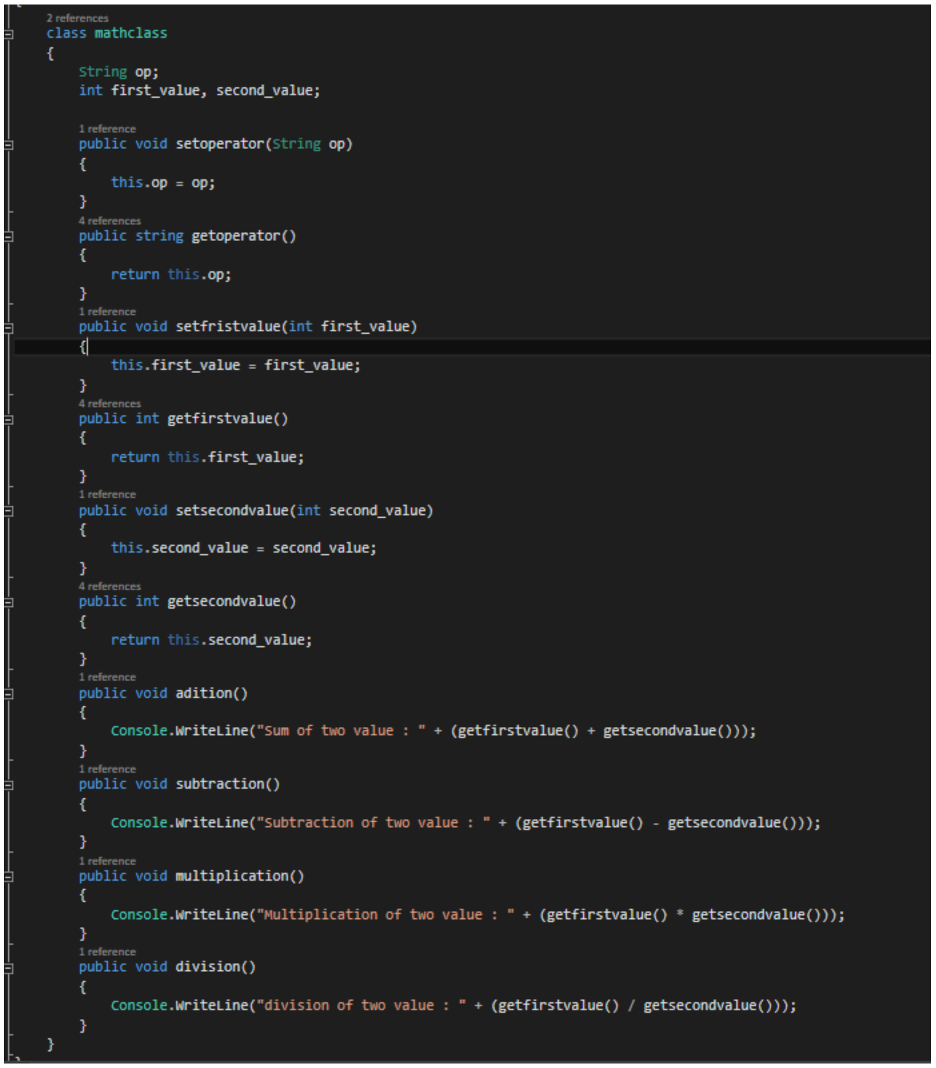
Solution:



```
Enter the First Value:
13
Enter the Second Value:
4
Enter the operation u want to perform:
-
Subtraction of two value : 9
Press any key to continue . . .
```



```
0 references
class Program
{

    0 references
    static void Main(string[] args)
    {
        mathclass obj = new mathclass();
        Console.WriteLine("Enter the First Value:");
        obj.setfristvalue(Convert.ToInt32(Console.ReadLine()));
        Console.WriteLine("Enter the Second Value:");
        obj.setsecondvalue(Convert.ToInt32(Console.ReadLine()));
        Console.WriteLine("Enter the operation u want to perform:");
        obj.setoperator(Console.ReadLine());
        if (obj.getoperator() == "+")
        {
            obj.adition();
        }
        else if (obj.getoperator() == "-")
        {
            obj.subtraction();
        }
        else if (obj.getoperator() == "*")
        {
            obj.multiplication();
        }
        else if (obj.getoperator() == "/")
        {
            obj.division();
        }
```



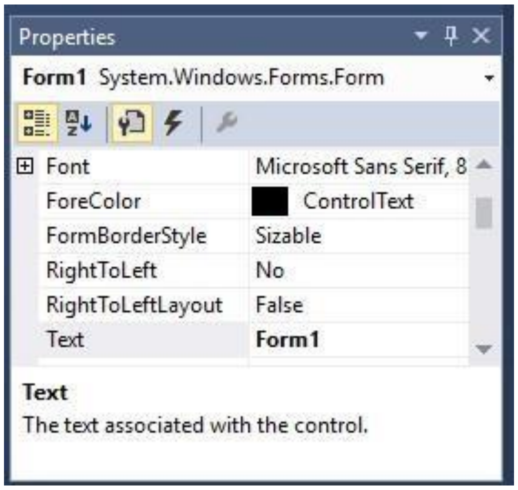
```
2 references
class mathclass
{
    String op;
    int first_value, second_value;

    1 reference
    public void setoperator(String op)
    {
        this.op = op;
    }
    4 references
    public string getoperator()
    {
        return this.op;
    }
    1 reference
    public void setfristvalue(int first_value)
    {
        this.first_value = first_value;
    }
    4 references
    public int getfirstvalue()
    {
        return this.first_value;
    }
    1 reference
    public void setsecondvalue(int second_value)
    {
        this.second_value = second_value;
    }
    4 references
    public int getsecondvalue()
    {
        return this.second_value;
    }
    1 reference
    public void adition()
    {
        Console.WriteLine("Sum of two value : " + (getfirstvalue() + getsecondvalue()));
    }
    1 reference
    public void subtraction()
    {
        Console.WriteLine("Subtraction of two value : " + (getfirstvalue() - getsecondvalue()));
    }
    1 reference
    public void multiplication()
    {
        Console.WriteLine("Multiplication of two value : " + (getfirstvalue() * getsecondvalue()));
    }
    1 reference
    public void division()
    {
        Console.WriteLine("division of two value : " + (getfirstvalue() / getsecondvalue()));
    }
}
```

OBJECTIVE 4: ACTIVITIES

Understanding and developing windows forms applications.

ACTIVITY 6 STEPS

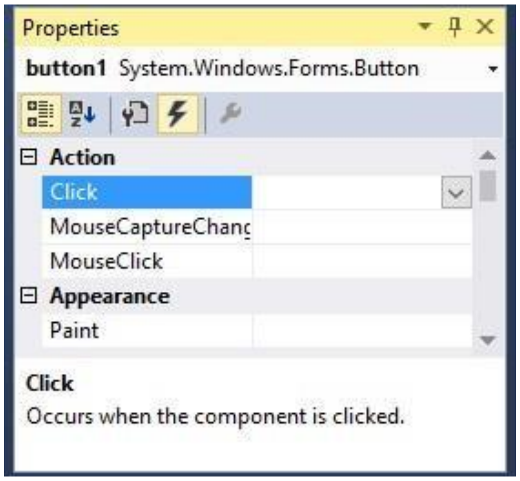
1. Create your first Windows forms application named DemoWindowsForms.
2. Form1, a default form as a startup form will be created by Visual studio in your application.
3. Now add following controls to your form by dragging controls from toolbox (CTRL + ALT + X to view toolbox) to form and set their properties and events if any from properties windows.


Properties

Form1 System.Windows.Forms.Form

Font	Microsoft Sans Serif, 8
ForeColor	■ ControlText
FormBorderStyle	Sizable
RightToLeft	No
RightToLeftLayout	False
Text	**Form1**

Text
The text associated with the control.

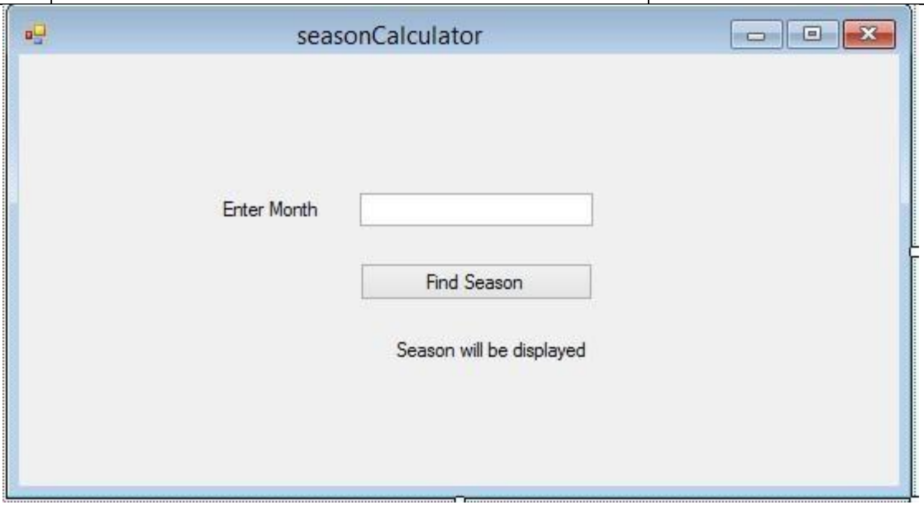

Properties

button1 System.Windows.Forms.Button

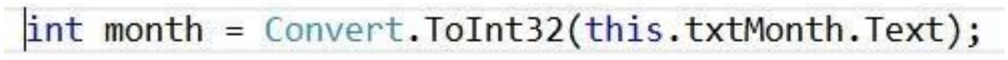
Action
Click
MouseCaptureChang
MouseClick
Appearance
Paint

Click
Occurs when the component is clicked.

Control	Set following properties	Events
Label1	1. Name = lblMonth 2. AutoSize = false 3. Text = “Enter Month Number ”	
Label2	1. Name = lblSeason 2. AutoSize = false 3. Text = “Here Season for specific month will be displayed”	
TextBox1	1. Name = txtMonth	
Button1	1. Name = btnSeason 2. Text = “Find Season”	1. Click Event : to add it double click the button



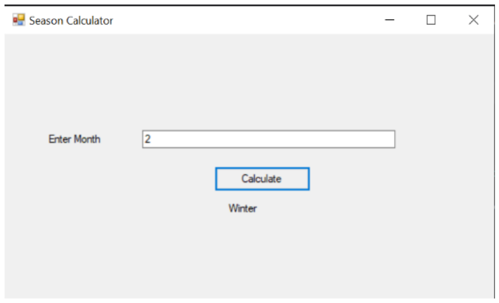
4. Get value of txtMonth (TextBox) in event handler method that will be invoked when you will click the Find Season button



```
int month = Convert.ToInt32(this.txtMonth.Text);
```

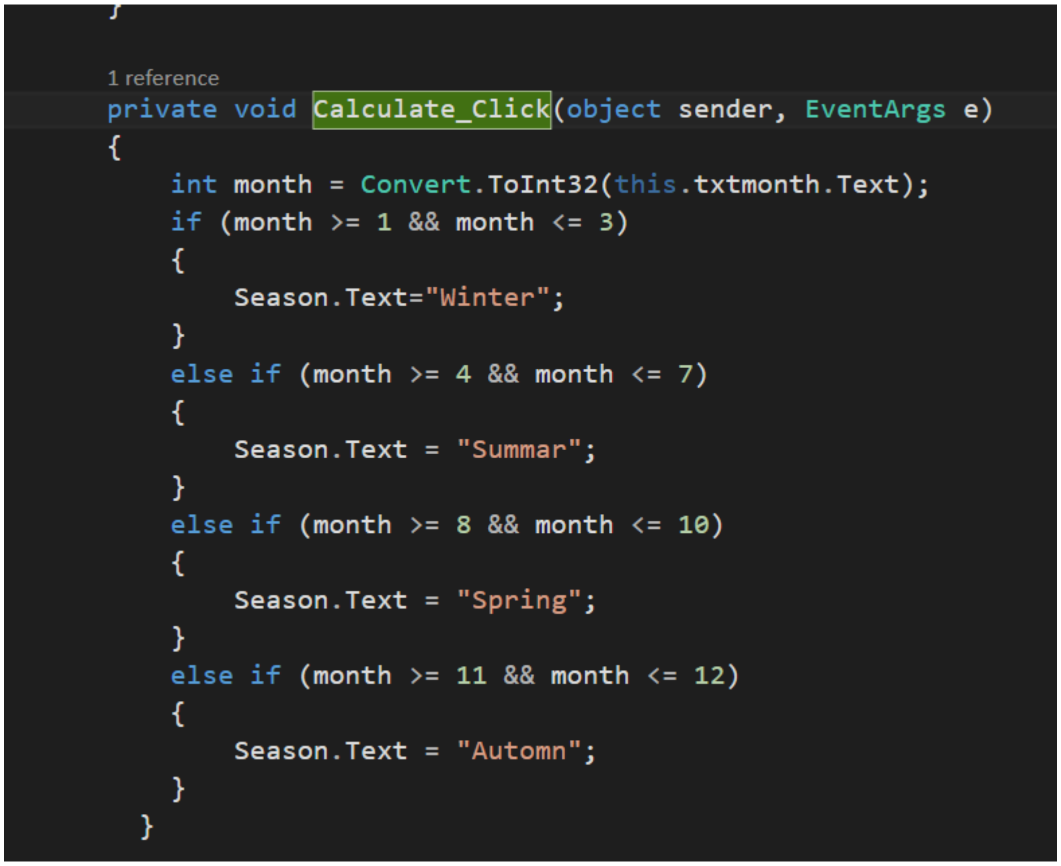
5. You can define method name that should be invoked on btnName_Click event of button using properties window in event section.
6. Write code to find out season using value of month variable.
7. Execute the application and validate the results.

Solution:



Enter Month 2

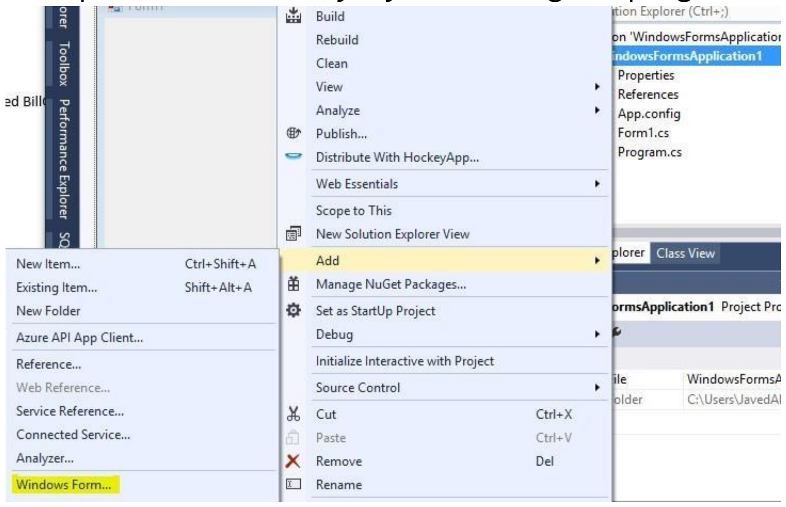
Winter



```
        }

        1 reference
        private void Calculate_Click(object sender, EventArgs e)
        {
            int month = Convert.ToInt32(this.txtmonth.Text);
            if (month >= 1 && month <= 3)
            {
                Season.Text="Winter";
            }
            else if (month >= 4 && month <= 7)
            {
                Season.Text = "Summar";
            }
            else if (month >= 8 && month <= 10)
            {
                Season.Text = "Spring";
            }
            else if (month >= 11 && month <= 12)
            {
                Season.Text = "Automn";
            }
          }
```

ACTIVITY 7: STEPS

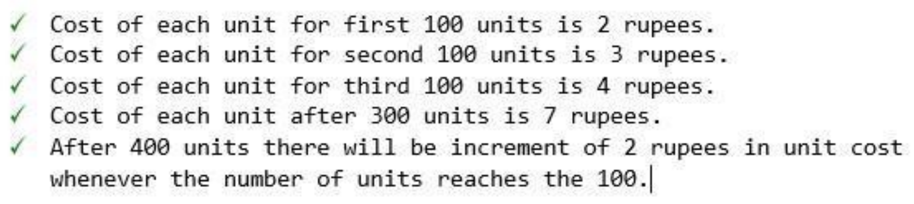
1. Add new form named BillCalculation in windows forms application created in previous activity by following steps given in snap shot.



2. Now add following controls on form.

Control	Set following properties	Events
Label1	1. Name = lblUnits 2. AutoSize = false 3. Text = “Enter Units ”	
Label2	1. Name = lblBill 2. AutoSize = false 3. Text = “Here Bill will be displayed”	
TextBox1	1. Name = txtUnits	
Button1	1. Name = btnCalculateBill 2. Text = “Calculate Bill”	2. Click Event : to add it double click the button

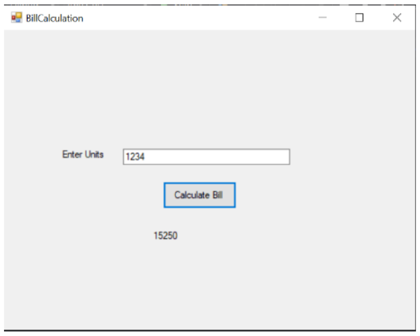
3. Get value of units from textbox txtUnits.
4. Write code for calculation of bill from units following these conditions.



- ✓ Cost of each unit for first 100 units is 2 rupees.
- ✓ Cost of each unit for second 100 units is 3 rupees.
- ✓ Cost of each unit for third 100 units is 4 rupees.
- ✓ Cost of each unit after 300 units is 7 rupees.
- ✓ After 400 units there will be increment of 2 rupees in unit cost whenever the number of units reaches the 100.

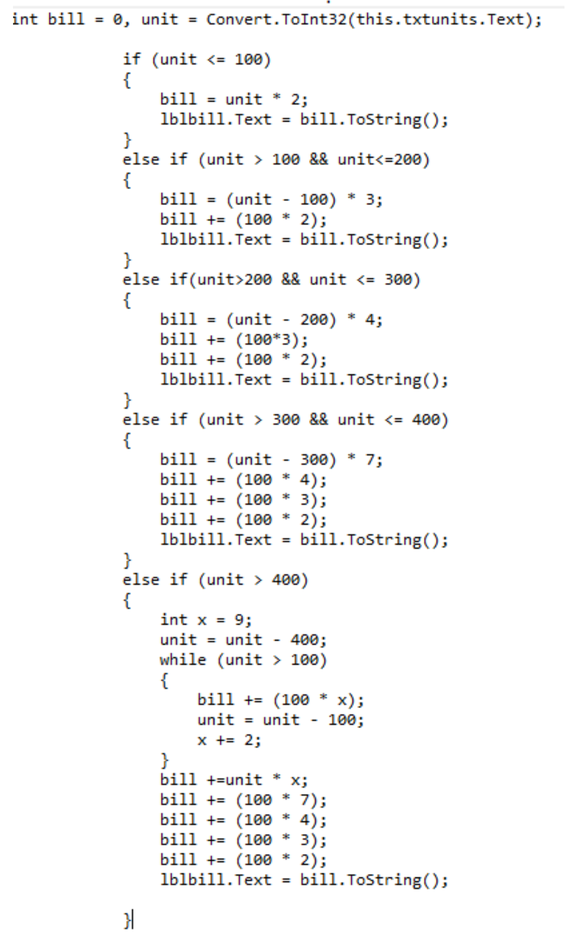
5. Execute the Program and validate the results.

Solution:



Enter Units 1234

15250



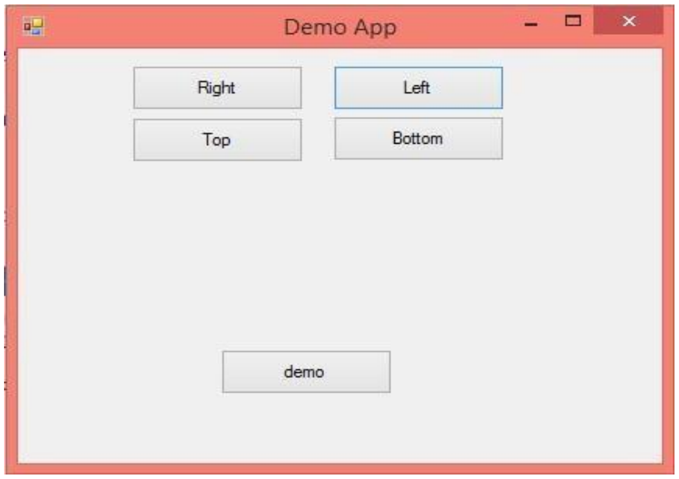
```
int bill = 0, unit = Convert.ToInt32(this.txtunits.Text);

            if (unit <= 100)
            {
                bill = unit * 2;
                lblbill.Text = bill.ToString();
            }
            else if (unit > 100 && unit<=200)
            {
                bill = (unit - 100) * 3;
                bill += (100 * 2);
                lblbill.Text = bill.ToString();
            }
            else if(unit>200 && unit <= 300)
            {
                bill = (unit - 200) * 4;
                bill += (100*3);
                bill += (100 * 2);
                lblbill.Text = bill.ToString();
            }
            else if (unit > 300 && unit <= 400)
            {
                bill = (unit - 300) * 7;
                bill += (100 * 4);
                bill += (100 * 3);
                bill += (100 * 2);
                lblbill.Text = bill.ToString();
            }
            else if (unit > 400)
            {
                int x = 9;
                unit = unit - 400;
                while (unit > 100)
                {
                    bill += (100 * x);
                    unit = unit - 100;
                    x += 2;
                }
                bill +=unit * x;
                bill += (100 * 7);
                bill += (100 * 4);
                bill += (100 * 3);
                bill += (100 * 2);
                lblbill.Text = bill.ToString();

            }
```

ACTIVITY 8: STEPS

1. Create windows form application named DemoApplication using visual studio or some other version if available.
2. Change name of Form1 to frmDemo and text to Demo App.
3. Add following controls on the form:

Control	Set following properties	Events
Button1	1. Name = btnDemo 2. Text = “Demo”	No event
Button2	1. Name = btnRight 2. Text = “Right”	btnRight_click()

Button3	1. Name = btnLeft 2. Text = “Left”	btnLeft_click()
Button4	1. Name = “btnTop” 2. Text = “Top”	btnTop_click()
Button5	1. Name = btnBottom 2. Text=”Bottom”	btnBottom_click()



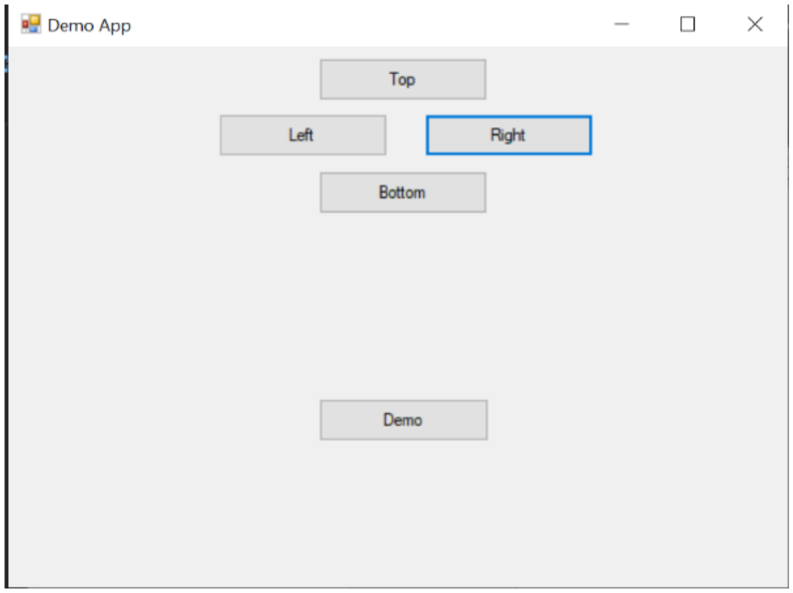
4. Write following code in click event of **btnRight** button.

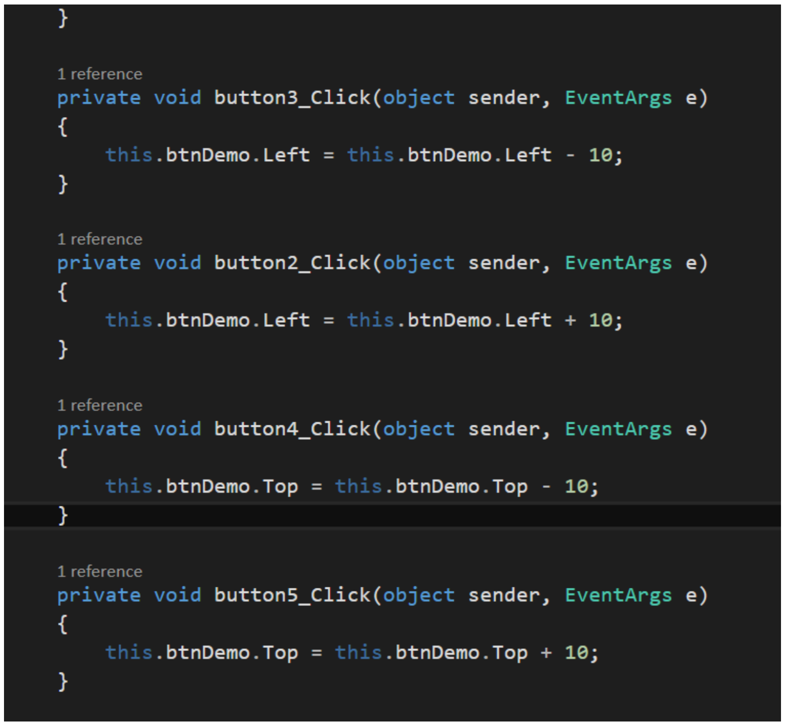
```
private void btnRight_Click(object sender, EventArgs e)
{
    this.btnDemo.Left = this.btnDemo.Left + 10;
}
```

5. Execute the code and analyze the output.
6. Write code for other buttons to achieve functionality indicated by their labels.

   Note: Use Top property of button control to move it downwards.

Solution:





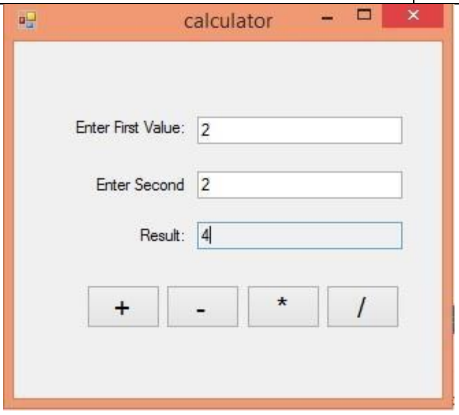
```
}

1 reference
private void button3_Click(object sender, EventArgs e)
{
    this.btnDemo.Left = this.btnDemo.Left - 10;
}

1 reference
private void button2_Click(object sender, EventArgs e)
{
    this.btnDemo.Left = this.btnDemo.Left + 10;
}

1 reference
private void button4_Click(object sender, EventArgs e)
{
    this.btnDemo.Top = this.btnDemo.Top - 10;
}

1 reference
private void button5_Click(object sender, EventArgs e)
{
    this.btnDemo.Top = this.btnDemo.Top + 10;
}
```

ACTIVITY 9: STEPS

1. Add new form in DemoApplication for simple calculator.
2. Add following controls on form.

Control	Set following properties	Events

TextBox1	1. Name = “txtFirstValue”	No event
TextBox2	1. Name= “txtSecondValue”	No event
TextBox3	1. Name=”txtResult” 2. Readonly = true	No event
Button1	1. Name = btnAdd 2. Text = “+”	Double click button to generate click event automatically as below: btnAdd_click()
Button2	1. Name = btnSubtract 2. Text = “-”	btnSubtract_click()
Button3	1. Name = “btnMultiply” 2. Text = “*”	btnMultiply_click()
Button4	1. Name = “btnDivide” 2. Text=”/”	btnDivide_click()

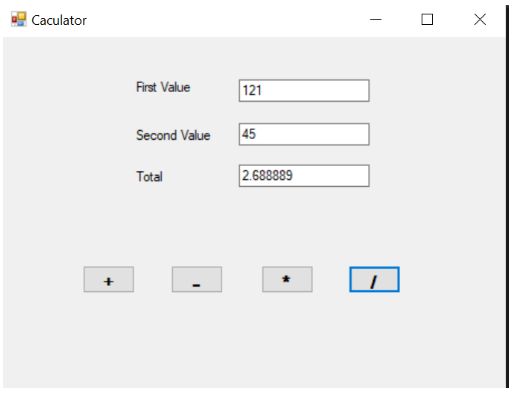


3. Add following code for Add operation and create code for other operations as well.

```
int firstValue;
int secondValue;

public calculator()
{
    InitializeComponent();
}

private void btnAdd_Click(object sender, EventArgs e)
{
    firstValue =  Convert.ToInt32(this.txtFirstValue.Text);
    secondValue = Convert.ToInt32(this.txtSecondValue.Text);
    this.txtResult.Text = "" + (firstValue+secondValue);
}
```

4. Write code for other buttons and add one extra button for clearing values of text boxes named btnClear.
5. Execute code and validate the results.

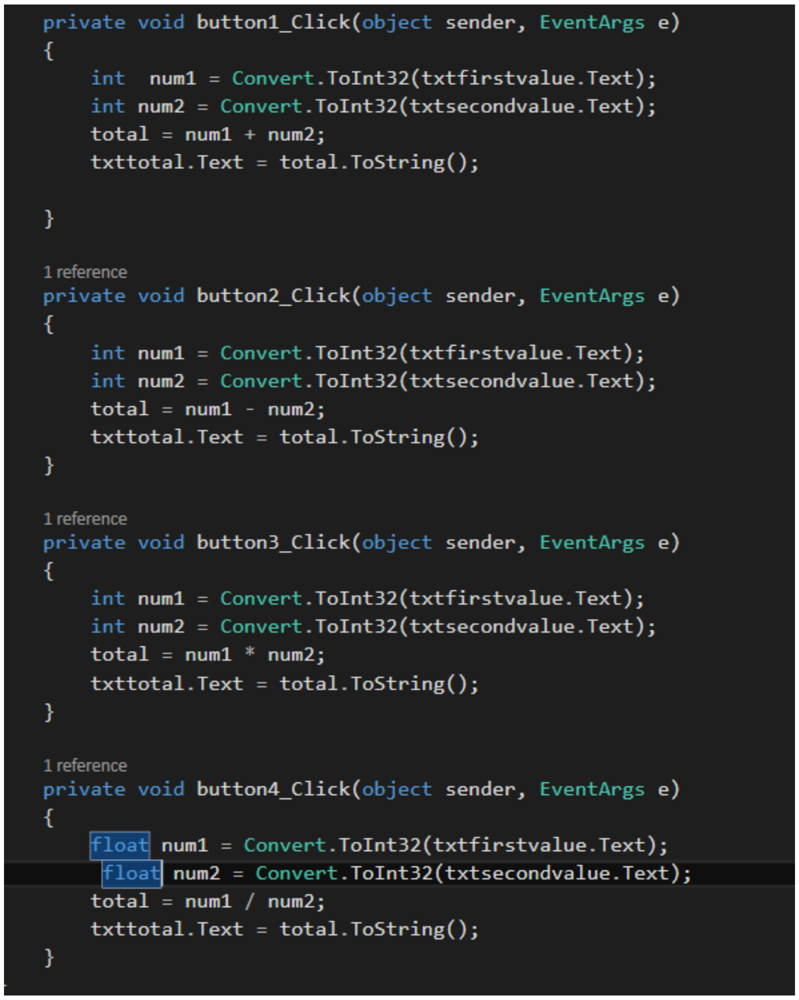
Solution:


Caculator

First Value 121

Second Value 45

Total 2.688889

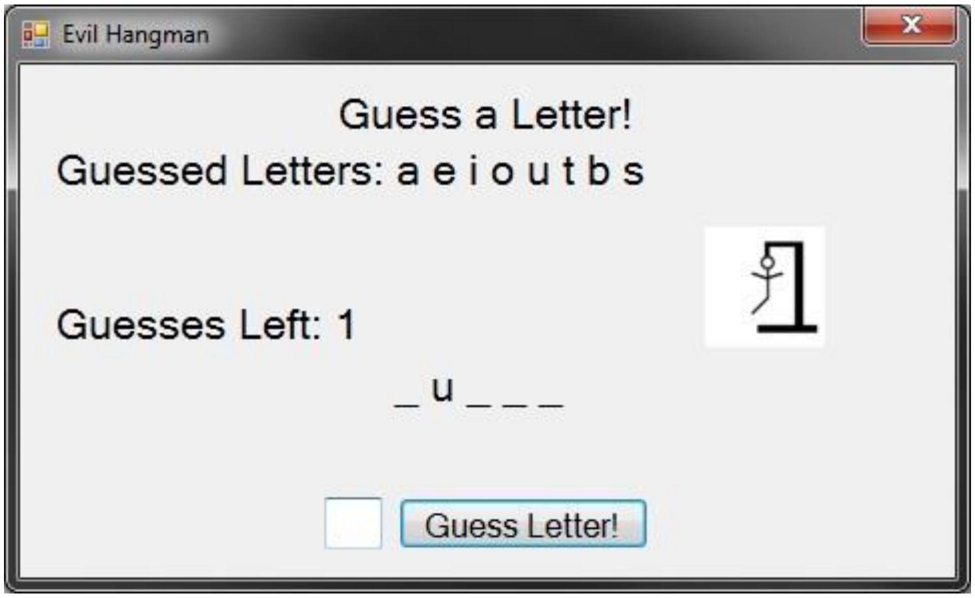

```
private void button1_Click(object sender, EventArgs e)
{
    int  num1 = Convert.ToInt32(txtfirstvalue.Text);
    int num2 = Convert.ToInt32(txtsecondvalue.Text);
    total = num1 + num2;
    txttotal.Text = total.ToString();

}

1 reference
private void button2_Click(object sender, EventArgs e)
{
    int num1 = Convert.ToInt32(txtfirstvalue.Text);
    int num2 = Convert.ToInt32(txtsecondvalue.Text);
    total = num1 - num2;
    txttotal.Text = total.ToString();
}

1 reference
private void button3_Click(object sender, EventArgs e)
{
    int num1 = Convert.ToInt32(txtfirstvalue.Text);
    int num2 = Convert.ToInt32(txtsecondvalue.Text);
    total = num1 * num2;
    txttotal.Text = total.ToString();
}

1 reference
private void button4_Click(object sender, EventArgs e)
{
    float num1 = Convert.ToInt32(txtfirstvalue.Text);
     float num2 = Convert.ToInt32(txtsecondvalue.Text);
    total = num1 / num2;
    txttotal.Text = total.ToString();
}
```

ACTIVITY 10: Making a Hangman Game

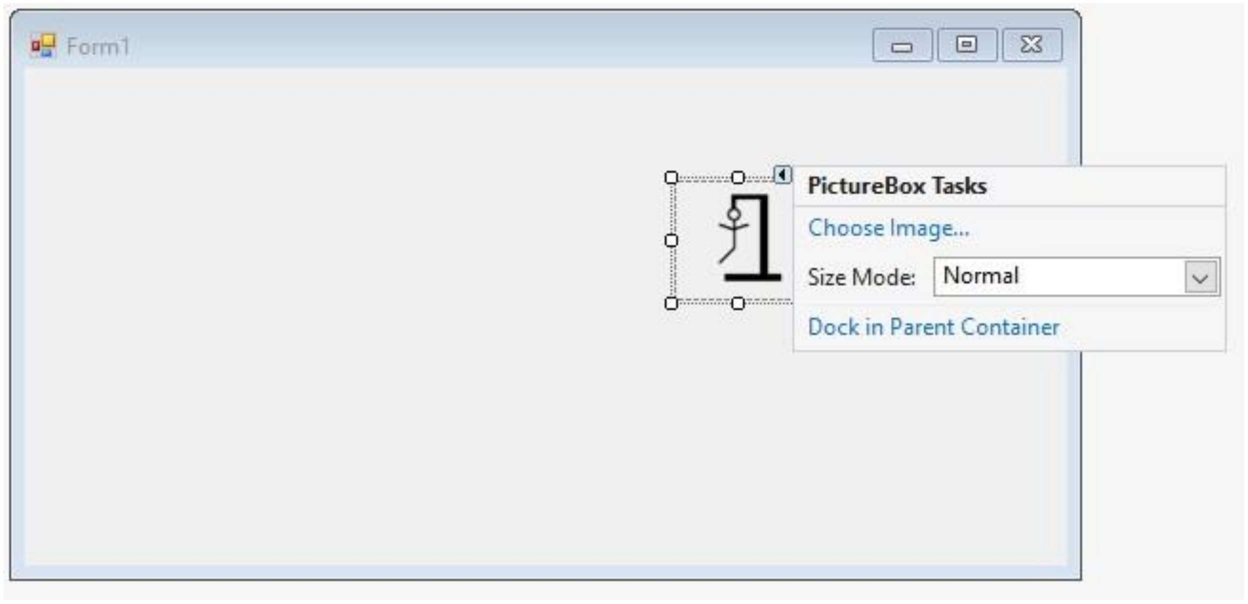
1. Create a new project named HangmanGame. 2. Add following controls on form

Control	Set following properties	Events
Label1	Guess a Letter!	
Label2	Guessed Letters:	
Label3	Guesses Left	
Label4	(A random name with random spaces)	
TextBox1	1. Name = GuessBox 2. MaxLength = 1	

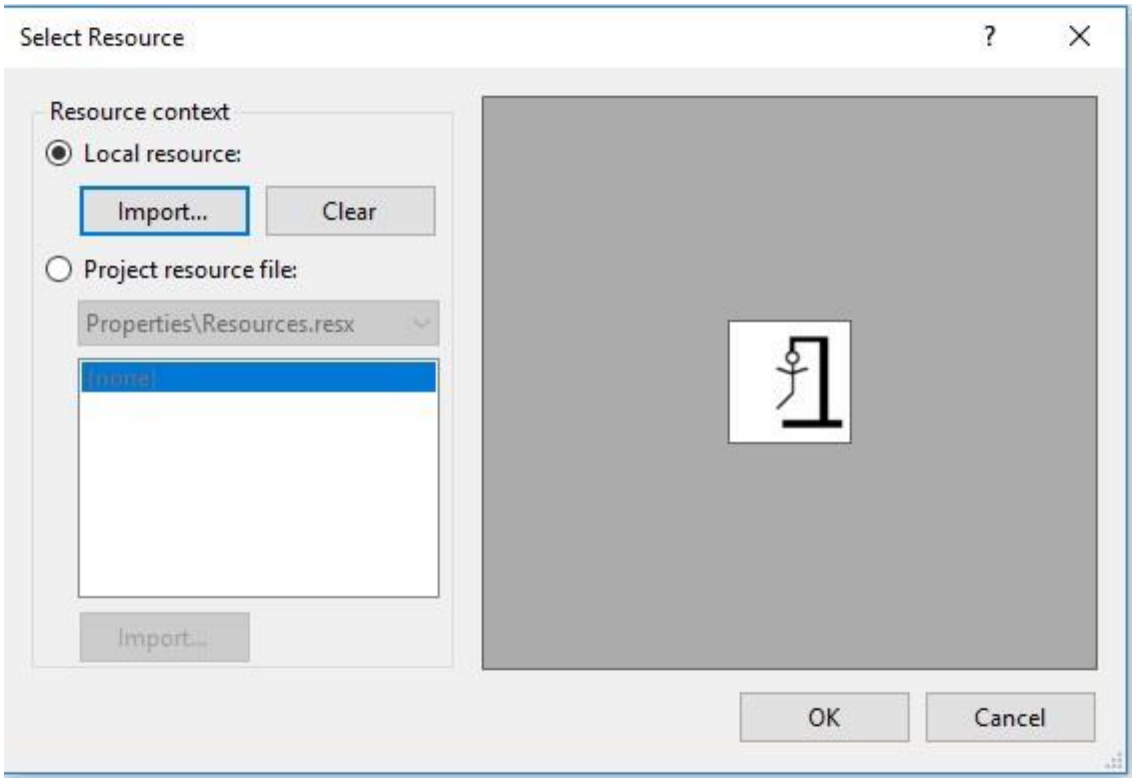
Button1	1. Name = btnGuess 2. Text = “Guess Letter!”	btnGuess_click()
PictureBox1	1. Name = picGuess	



3. Use following steps to add images to the PictureBox 4. Drag and drop PictureBox on the form1.
5. Click on the play button to open a context menu.

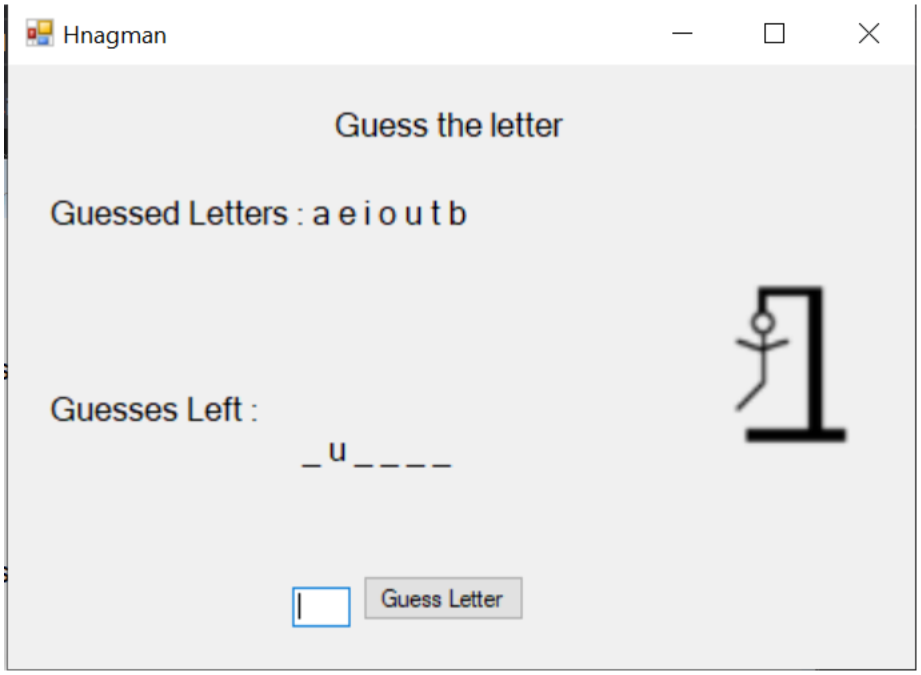


6. Click choose Image
7. On the new popup go for the Import Image from local resource



8. Click ok and the image will be imported in the selected PictureBox.

Solution:



Guess the letter

Guessed Letters : a e i o u t b

Guesses Left :

_ u _ _ _ _