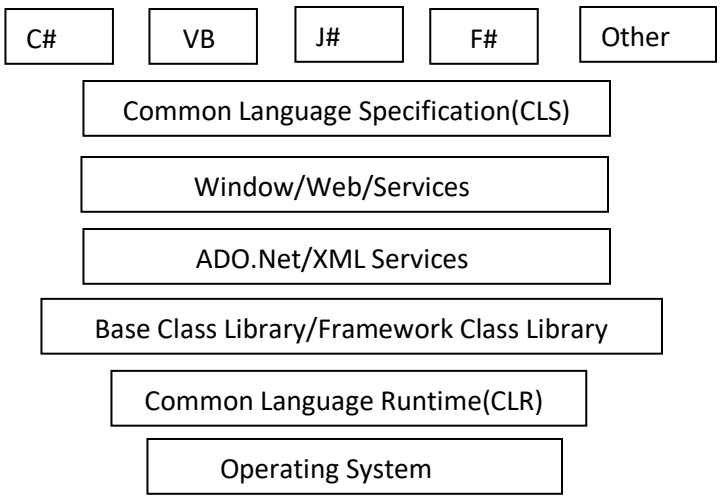
.**Net Framework Architecture**



Common Language Runtime: It provides Virtual (buffer) Environment.

- Memory management
- Garbage collection
- Thread management: Operating system reads the main method.
- Exception HDLINE
- Code Access Security: Like permissions denied.

Technology: the application of scientific knowledge for practical purposes, especially in industry. It will reduce the ability to spend money on new technology.

Application: An application is any program, or group of programs, that is designed for the end user. Application software can be divided into two general classes: systems software and application software.

Program: A computer program is a collection of instructions that performs a specific task when executed by a computer. A computer requires programs to function, and typically executes the programs instruction in a central processing unit.

Language: Collection of various components. Also in which specifies the syntax and semantics of a database programming language. The interactions so described and so implemented are to be performed on behalf of application programs written in programming language.

Framework 4.6

Dot Net has

- Pre build components
- Pre build library

- Smart /intelligence
- 42 Languages (in this 28 are in build and rest can call from DLL and scripts).
- Support inter operability concept.

C# + VB = Application

Dot Net: has no compiler because dot net is not as language. But own language has individual compiler.

Note: If you are a client and you want to use .net application then you need only .net framework according to operating system.

Developer ⟶ Microsoft /operating System Client ⟶ Framework

IDE: Integrated Development Environment

It is intellisense is best.

Pre-build component.

Pre buid library.

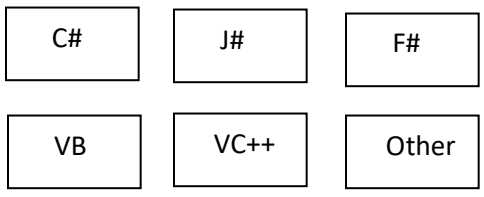
Prebuild CGI & GUI controls, templates.

Library

25,000 name spaces.

Mscorb.dll (Dynamic Link Library)

In DLL: we can also say Framework Class Library/ Base Class Library.



Dynamic Link Library (DLL): a DLL is a library that contains code and data that can be used by more than one program at the same time. This helps promote code reuse and efficient memory usage.

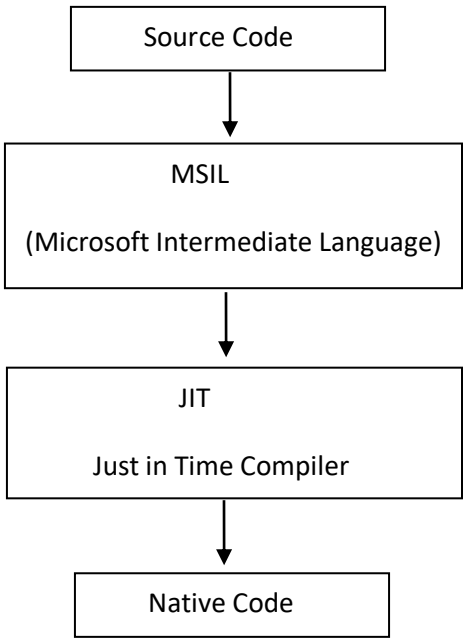
Advantage

1. When multiple programs use the same library of functions, a DLL can reduce the duplication of code that is loaded.

Name Space used in C#: Called collection of classes.

Using System;

Here Using is keyword and System is namespace.



- It is a platform independent
- It checks the library
- JIT creates native code
- Compatible with CPU

Pre defined data types: They are data types provided by Microsoft, it cannot be changed, increased or decreased. This can be used only as it is.

User Defined data types: These data types are defined by the user and it can be modified according to requirement.

.**Net Framework Architecture**

Languages used: The most valuable feature of the .Net framework is its compatibility to work with different languages and even to perform the cross language options. This means that the .Net developer can use different languages in is work and still the application the cross language feature has its roots in the common runtime of the used languages. There are about 42 languages such as VB, ASP, C?#, ada, F#, J#, L# etc.

CLS: The common language specification offers the new set of common standards in the .Net framework. This calls library, compiler and environment

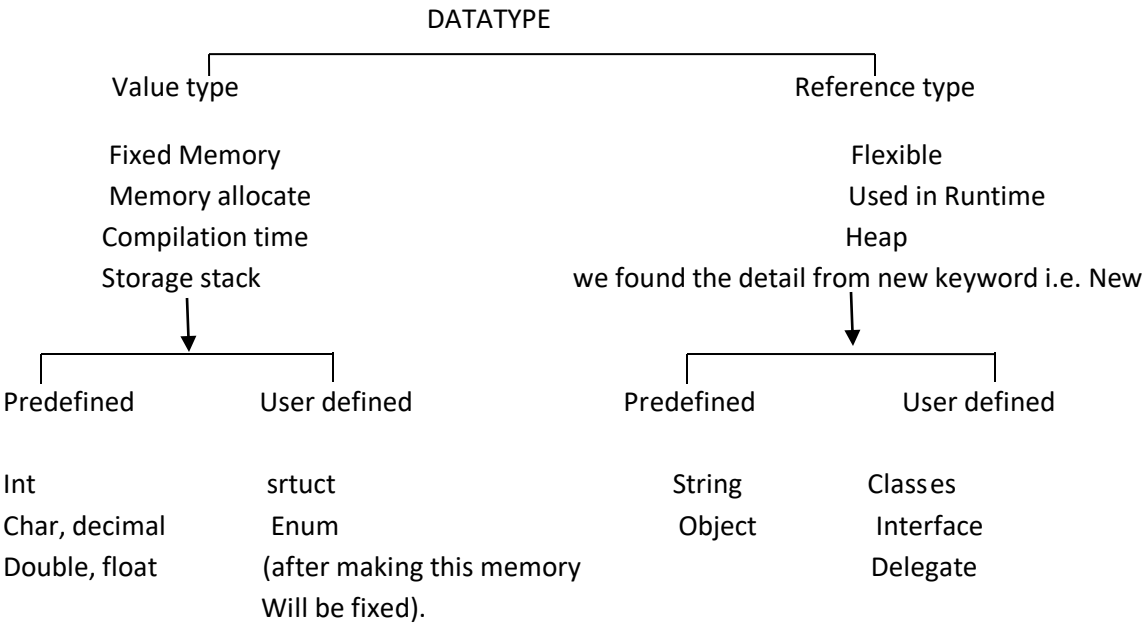
ADO.Net: This is used for database and services etc.

FCL: Base class library or framework class library.

All controls are made in the XML (Extensible Markup Language). XML is platform independent language.

About C#

- ✓ Object oriented programming language.
- ✓ Line terminated by semiconductor
- ✓ Body defines with **{}. I**t is called **scope.**
- ✓ C# compiler name is CSC.



Predefined data types: They are data types provided by Microsoft, it can't be changed, increased or decreased. This can be used only as it is.

User defined data types: These data types are defined by the user and it can be modified according to requirements.

Method:

- **Return type**
  Method type must be same; method returns type (it, float, and char).

- **Non-return type**
  Method type must be wide and method will return nothing. OR Wide

```
Access modifier    Datatype Method Name (Parameter optional)
{
 Int a;
}
```

- ✓ If you want to declare any method then it needs **structure and class.**

Classes

Public	**Private**
Within class or struct OR **Outer of** class/Struct OR Outer of scope/ {}	**With in** class Sturct or /within scope {}

Int a: is a variable; Variable can be int is a value type of Datatype.
Char a: a is a character.
Class a: a is reference variable.

Note: By default the private is access modifier.
Ex: namespace xyz

```
{
 Struct abed
 {
   Public int a;
   Public int add(int a, int b);
   {
     Int c= a+b;
     Return c;
  }
 }
}
```

```
Public Static Void Main();
```

Public: Access modifier
Static: Create memory clone OR fixed memory (Datatype & reference value will be fixed).
Automatically memory creation.
Void: NULL value.
Main(): Operating system reads the code(compilation and interpretation). Reads Memory of the OS.

Ex: Static
Static int a=10 [value can be overwrite]

clone

A 10	10

DATATYPE

A particular kind of data item, as defined by the values it can take, the programming language used , or the operations that can be performed on it.

Ques: Difference between Datatype in value type and reference type.

Value type:

It stored on stack. All value types are derived implicitly from the System.Value type. Value type are stored on stack , contains actual value.

Unlike reference types, a value type cannot contain the **null** value. However, the nullable types feature does allow for values types to be assigned to null.

Memory is allocated at compile time.

Types of Value type:

1. **Predefined**
   Int, float, char
2. **User defined**
   struct, Enum

The value types consist of two main categories:

- Structs
- Enumerations

Structs are into these categories:

- Numeric types
  - Integral types.
  - Floating-point types.
  - Decimal
- Bool and user defined stucts

Struct: is a value type that is typically used to encapsulate small groups of related variables, such as the co ordinates of a rectangle or the characteristics of an item in an inventory.

Ex: public struct Book

{ Public decimal price;

Public string title;

Public string author; }

Enumerations: Enum or Enumeration, a distinct type that consists of a set of named constants called enumerator.

An enum can also be nested within a class or struct.

By default, the first enumerator has the value 0, and the value of each successive enumerator is increased by 1. Ex. Sat is 0, sun is 1, Mon is 2 and so forth.

Strack : Last in first out like values are stored on one another like a stack.

Reference type:

They are stored on heap. It may contain reference to a value. Also can contain null values. This type required garbage collector to free memory.

Memory is allocated at run time.

Types of Value type:

1. **Predefined**
   String, object
2. **User defined**
   Classes, interface, delegate.

String

14-01-2016

MSIL has Link building (parent to child)

↓

Check the library

Types of Library

↓

Predefined User defined 3rd party Library

We can use other language, library like JAVAs, MySQL, and Libraries.

Found in Bin folder (3rd library) of .net.

We can't share 3r library that's why we store into BIN.

Link building is also called Assembly.

Compile + Build+ Debug

Compile: It checks the block of space in compile time. Also fixed it. If increases then it gives overflow exception.

Runtime: it is flexible, it may increase blank space  .Memory allocation done at runtime only.

Build: Compile +Link building(after that it create **dll).**

Debug: Build+Execution.

Using System.Data; //Subnamespace

Using System.SqlClient; //subnamespace

Namespace:

Namespace is used to call parent class.

Like class1 and class2 with same namespace. so, namespace will call as parent namespace.

```
Using demofirst;

Namespace demofirstApps

{  public class Class1

   {   static void main()

       { Class2 aa=new Class2();
```

```
using demofirst;

namespace demofirst

{    public class Class2

     { ---  }
```

Manage Code **Unmanaged Code**

1. Run under CLR Non CLR execution or develop.

2. OR CLR under execution develop lib. Bin Floder contains the libraries. Bin also contain 3rd party

Accessmodifier Datatype MethodName(Parameter optional)

{

}

Ex: namespace xyz

{ struct add

```
{   public int a;
    public int add(int a, int b)
   {  int c=a+b;
      Return c;
  } }}
```

Public static Void Main()

Access Modifier — Create Memory: Clone OR Fixed memory (datatype & reference values will be fixed.) — NULL — Operating system reads the code: Memory of os.

Static: Automatically memory creation.

(): called modulator & **{} :** Called scope.

Static: static int a=10;

Method Overloading

Method Overloading is a feature that allows a class to have two or more methods having same name, if their argument lists are different. In the last tutorial we discussed constructor overloading that allows a class to have more than one constructors having different argument lists.

Argument lists could differ in –

1. Number of parameters.
2. Data type of parameters.
3. Sequence of Data type of parameters.

Method overloading is also known as **Static Polymorphism**

In other words, Method & class is same but parameter are same.

WriteLine is *Widetype method.*

ReadLine: *Using ReadLine; Here Executer checks may programmer want to write more so readline is used to hold the screen*.

Write: *for output same line* Hello India

WriteLine: *for output in different line* Hello

India

Note: ReadLine: String type

WriteLine: Wide type.

String Type :

The **string** type represents a sequence of zero or more Unicode characters. **string** is an alias for String in the .NET Framework.

Although **string** is a reference type, the equality operators (**==** and **!=**) are defined to compare the valuesof **string** objects, not references. This makes testing for string equality more intuitive. For example:

The [] operator can be used for readonly access to individual characters of a **string**:

```
    string str = "test";
char x = str[2];  // x = 's';
```

String literals are of type **string** and can be written in two forms, quoted and @-quoted. Quoted string literals are enclosed in double quotation marks ("):

```
class SimpleStringTest
{
   static void Main()
   {
      string a = "\u0068ello ";
      string b = "world";
      Console.WriteLine( a + b );
      Console.WriteLine( a + b == "Hello World" ); // == performs a case-
sensitive comparison
   }
}
/* Output:
    hello world
    False
```

Left side data type must be equal to right side data type.
String a=10; **[Note:** Wrong data types are not same on both side.]
Int b="Hello";

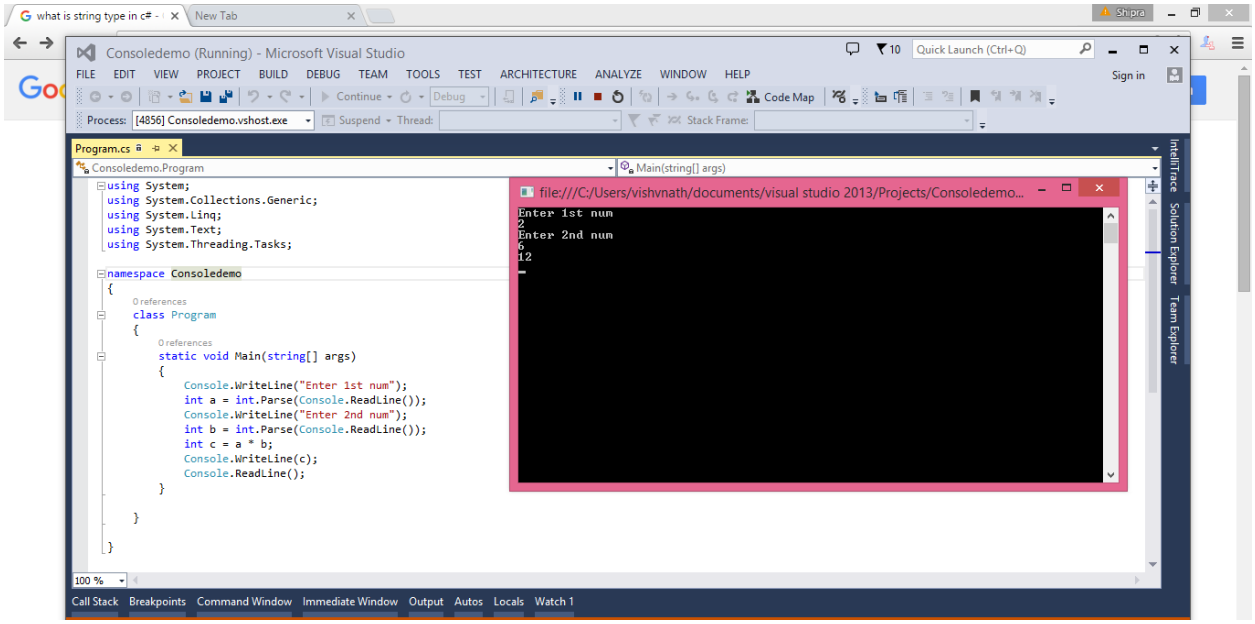
Ex. Console.Write("Hello");
Console.Write("India");

Output: Hello India

Console.WriteLine("Hello");
Console.WriteLine("India");

Hello
India

Console.WriteLine: Parse is a method to convert one data type into another.



21-Jan-2016

Implicit : Is pure conversion/strong conversion(Compile time).

It return 0 value if error occur

Convert one data type to another data type.

Explicit: is impure/not string.

Convert multiple data type into another data types.

One class with multiple data items, it should be matched with another.

It never return ***Null value*** if error occur in runtime.

Ex. Runtime ***Null value***

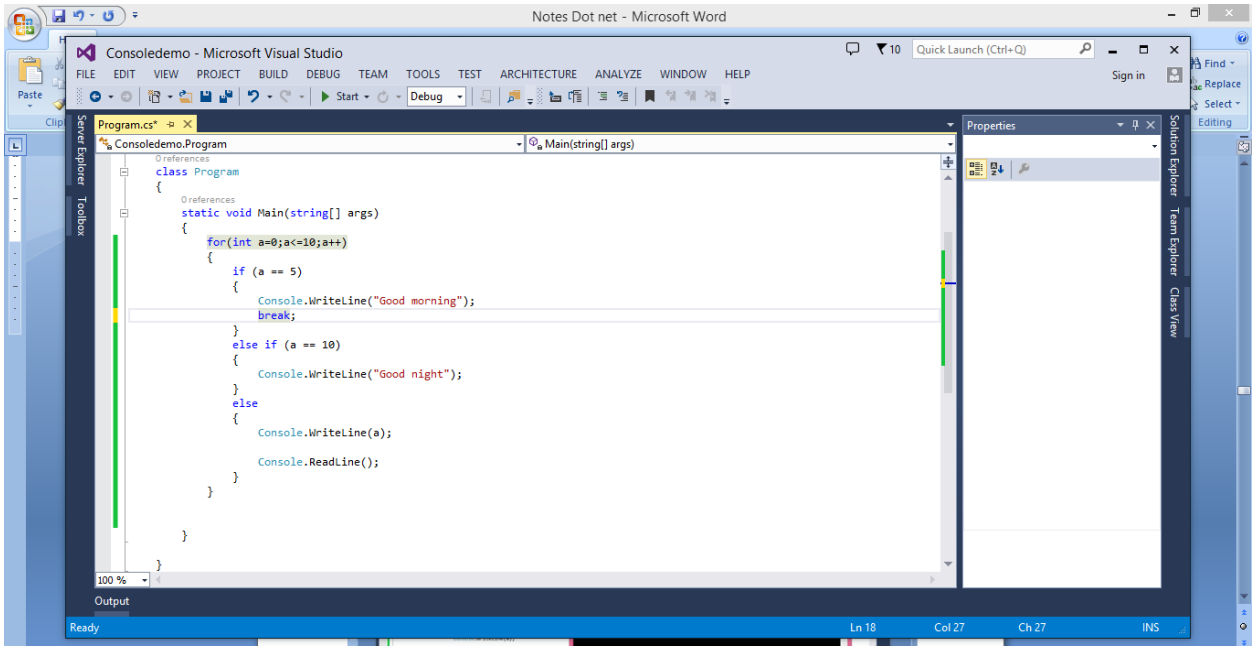
Convert.ToInt32()

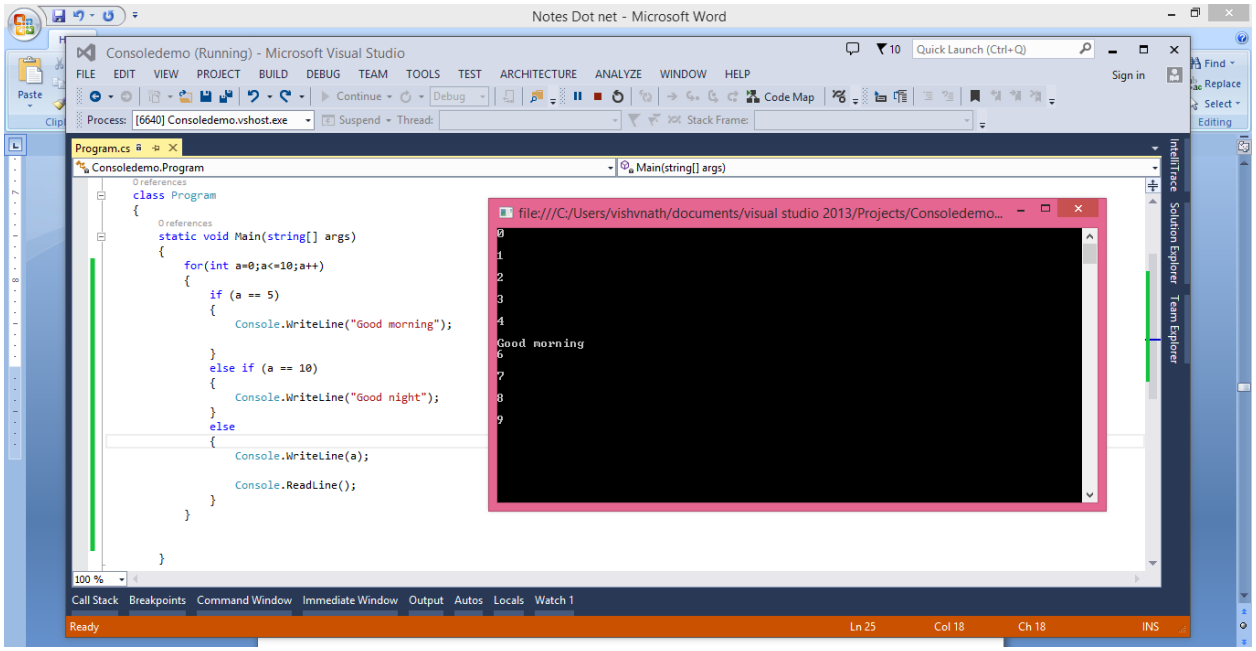
Int.Parse(

Compile (empty value)

Condition: Boolean

Q. if we wish to stop the check in between then use *break.*





Without **break**

27-Jan-2016

Struct: Never inherit and never re-used. But we can done multiple program (using multiple logics).

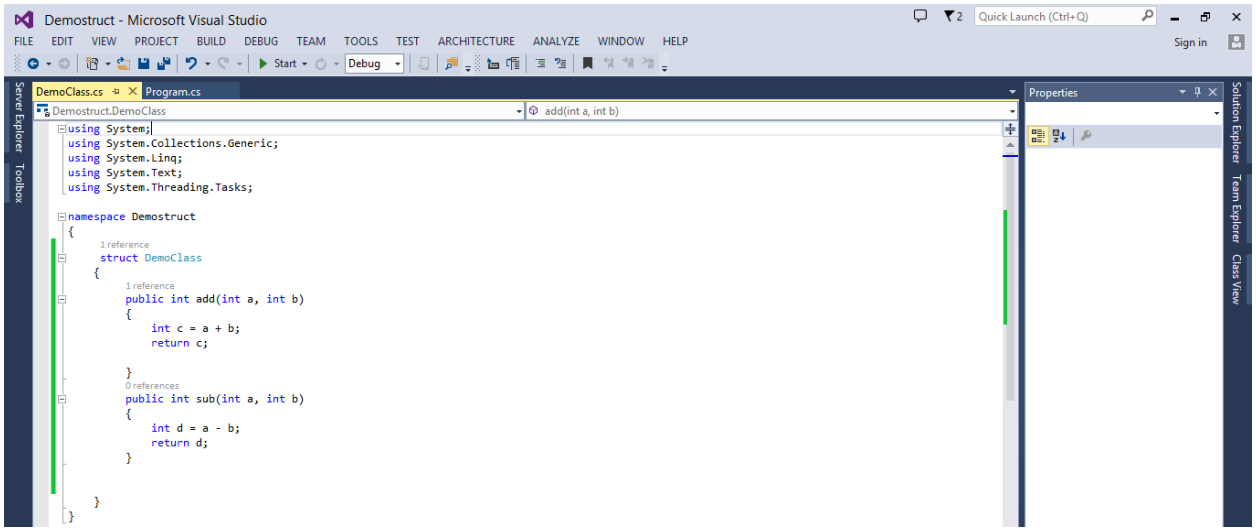
It has pre-build struct.

```
Ex: stuct Demostruct

{   public int a=10;

   Int add(int a, int b)

 Return c;

}

Int sub(int a, int b)

{  return a-b;

}}
```

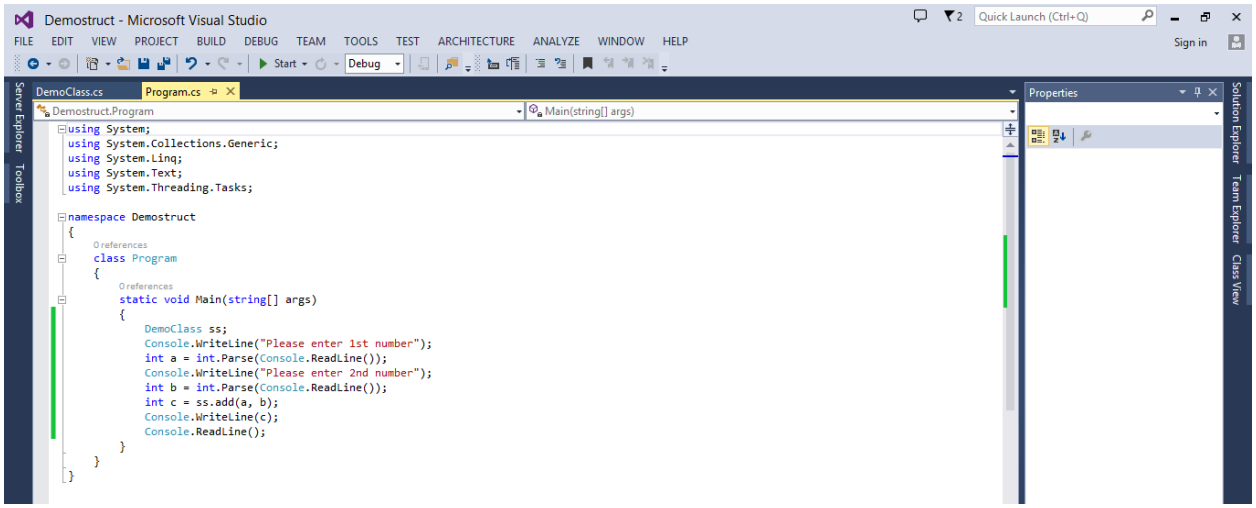
Note: Here datatypes is integer type(add) and also sub method consist integer type. Hence it will return integer type values.

1. Take console application
2. Add new item ⟶ Class(class Demostruct)

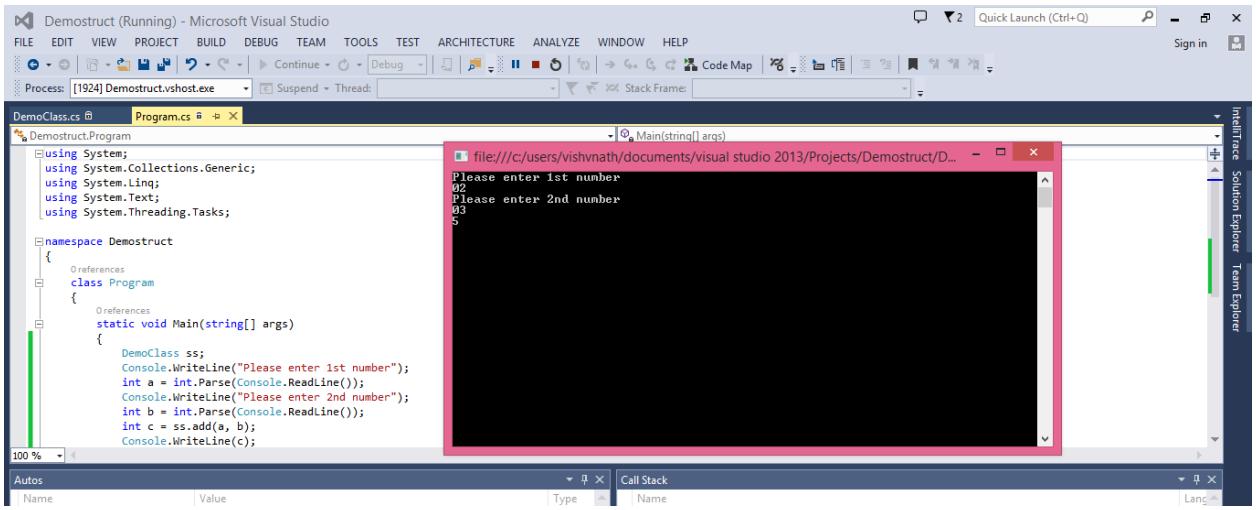
Create Class: `DemoClass`



Create Console App: `Demostruct`

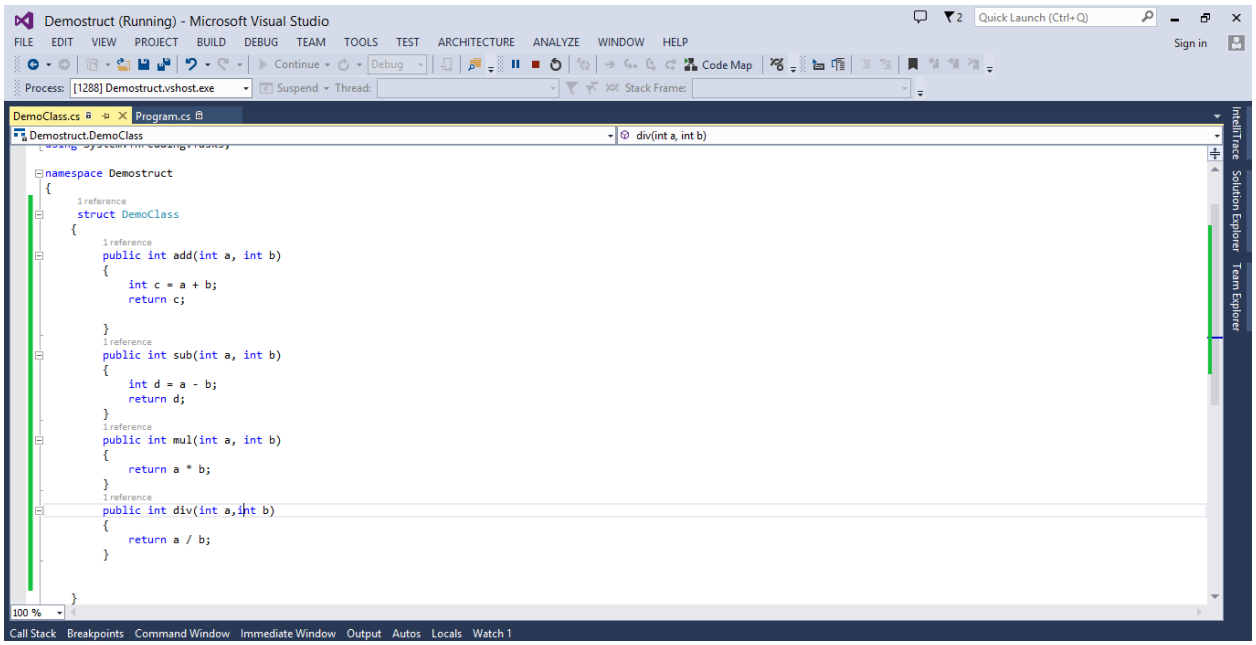


Output

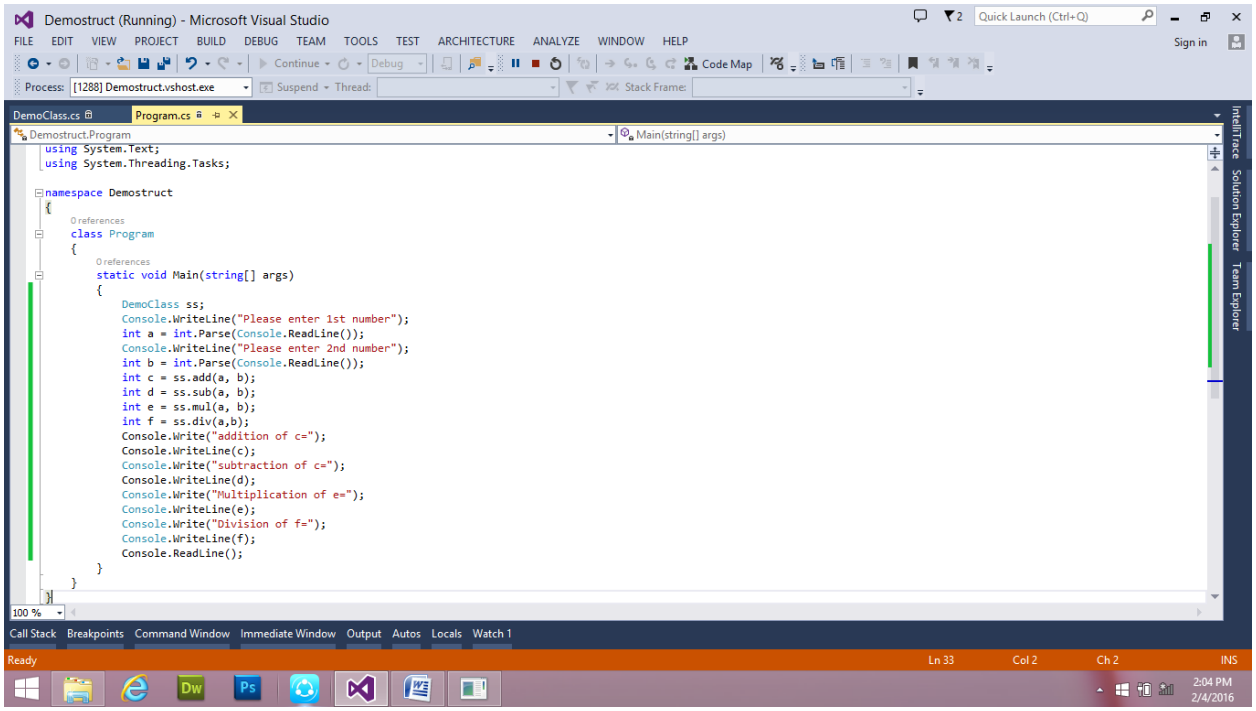


Ques: how to use strut for *add, sub, mul, div.*

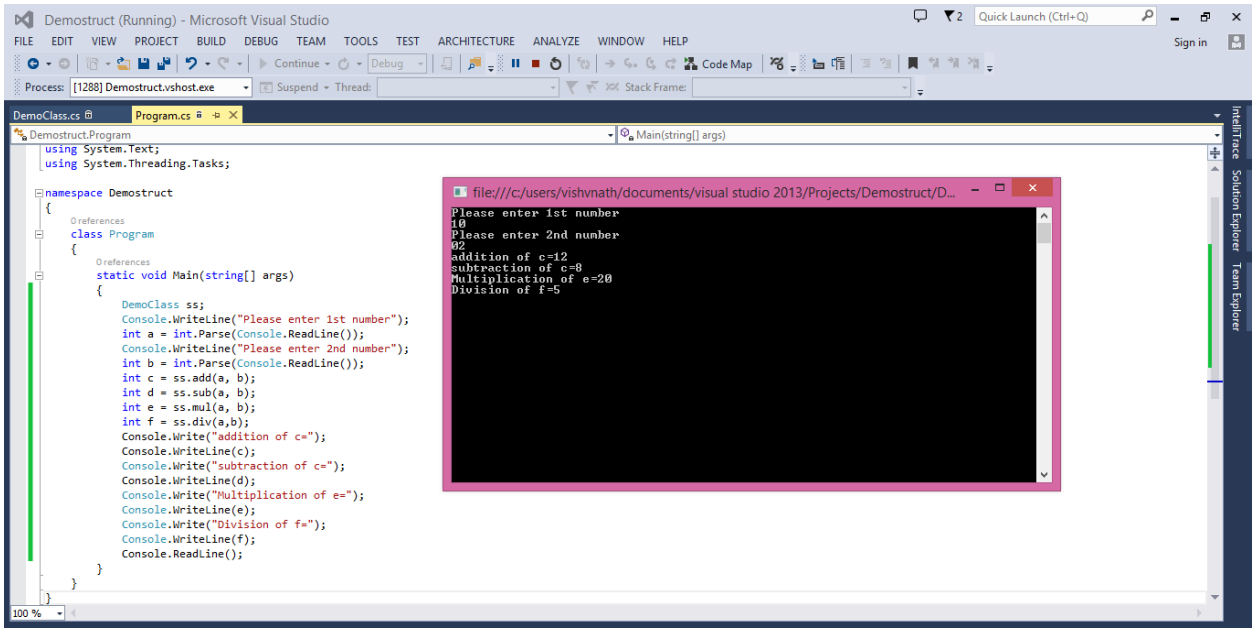
Create Class: `DemoClass`



Create Class: `Demostruct`



Output


Demostruct (Running) - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Sign in
Quick Launch (Ctrl+Q)
Continue
Debug
Code Map
Process: [1288] Demostruct.vshost.exe
Suspend
Thread:
Stack Frame:
DemoClass.cs
Program.cs
Demostruct.Program
Main(string[] args)
using System.Text;
using System.Threading.Tasks;
namespace Demostruct
{
0 references
class Program
{
0 references
static void Main(string[] args)
{
DemoClass ss;
Console.WriteLine("Please enter 1st number");
int a = int.Parse(Console.ReadLine());
Console.WriteLine("Please enter 2nd number");
int b = int.Parse(Console.ReadLine());
int c = ss.add(a, b);
int d = ss.sub(a, b);
int e = ss.mul(a, b);
int f = ss.div(a,b);
Console.Write("addition of c=");
Console.WriteLine(c);
Console.Write("subtraction of c=");
Console.WriteLine(d);
Console.Write("Multiplication of e=");
Console.WriteLine(e);
Console.Write("Division of f=");
Console.WriteLine(f);
Console.ReadLine();
}
}
}
100 %
file:///c:/users/vishvnath/documents/visual studio 2013/Projects/Demostruct/D...
Please enter 1st number
10
Please enter 2nd number
02
addition of c=12
subtraction of c=8
Multiplication of e=20
Division of f=5
IntelliTrace
Solution Explorer
Team Explorer