C# Language

Lecturer: Le Thi Bich Tra

Content

- Part 1: C# Basic
- Part 2: ASP.NET MVC Basic
- Part 3: Report

Part 1: C# Basic

- Unit 1.1:.C# language basic
- Unit 1.2: OOP in C#
- Unit 1.3: Collection and Generics
- Unit 1.4: LINQ

Part 2: ASP.NET MVC 5

- Unit 2.1: Introduction to ASP.NET MVC
- Unit 2.2: Controllers in an MVC application
- Unit 2.3: Views in an MVC application
- Unit 2.4: Html Helper in MVC
- Unit 2.5: Validation

Requirements

- Attend and actively participate in class
- Dilience: 10%
- Tests: 10%
- Mid-Term test: 30%
- Final Term test: 50%

Reference books

- Illustrated C# 2012, Daniel Solis
- Pro C# 5.0 and .NET 4.5 Framework, Andrew Troelsen.
- C# 6.0 and the .NET Framework, Seventh Edition, Andrew Troelsen and Philip Japikse.
- Programming Microsoft ASP.NET MVC, Third Edition, Dino Esposito.
- Pro ASP.NET MVC 5, Fifth Edition, Adam Freeman
- ASP.NET MVC 4 in Action
- Entity Framework 4 in Action,Stefano Mostarda, Marco De Sanctis and Daniele Bochicchio.
- Some tutorials about ASP.NET MVC in the Internet.

Reference books

- Some tutorials about ASP.NET MVC in the Internet.
- https://www.youtube.com/watch?v=M0jdFS4ZyEk&list=PLRhlTlpDUWsyK1TIsewrQ7WwC7QkCSCPD
- https://www.youtube.com/watch?v=izSNQbKhlEY&list=PLJbBHp6iPUiFw23Fijb-Jebnzm_gdruRr
- https://www.youtube.com/watch?v=-pzwRwYlXMw&list=PL6n9fhu94yhVm6S8I2xd6nYz2ZORd7X2v

Software for the C# subject

- Microsoft Visual Studio Community 2015

https://www.microsoft.com/en-us/download/details.aspx?id=48146

- Microsoft SQL Server 2014 Express

https://www.microsoft.com/en-US/download/details.aspx?id=42299

Thank You !

Unit 1.1:
.NET Framework and C# language

Lecturer: Le Thi Bich Tra

Content

- Lession 1:.NET Framework
- Lession 2: C# fundamentals
- Lession 3: Methods
- Lession 4: Enumeration type
- Lession 5: Array

Lession 1: .NET Framework

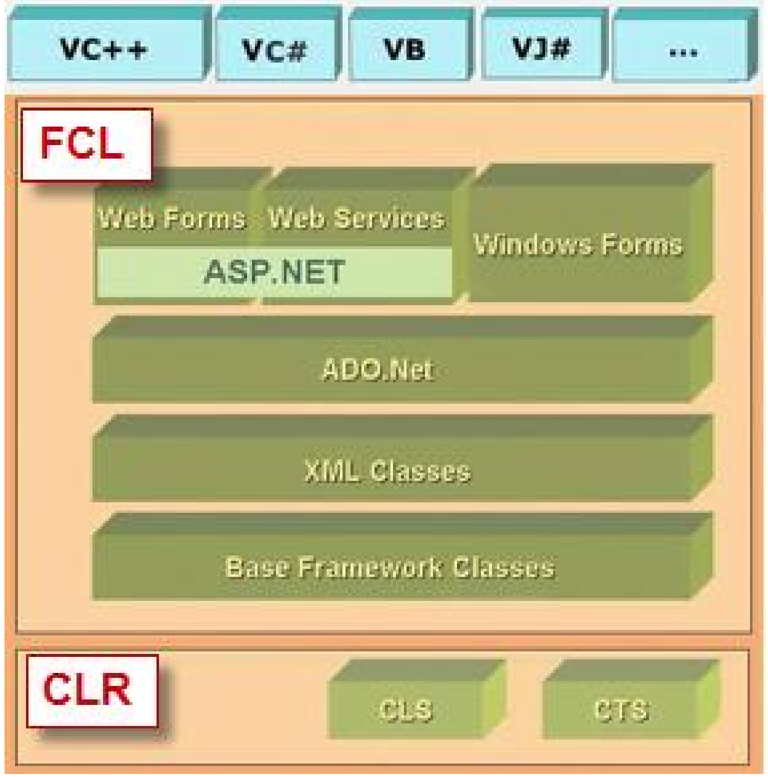
- Overview
- Architecturer
- How to complie .NET language

What is .NET Framework?

- The .NET framework is a software development framework from Microsoft.
- It provides a controlled programming environment where software can be developed, installed and executed on Windows-based operating systems.

Architecturer

- The .NET framework architecture comprises 2 key components:
  1. The .NET Framework Class Library (FCL) or Base Class Library (BCL)
  2. The Common Language Runtime (CLR)

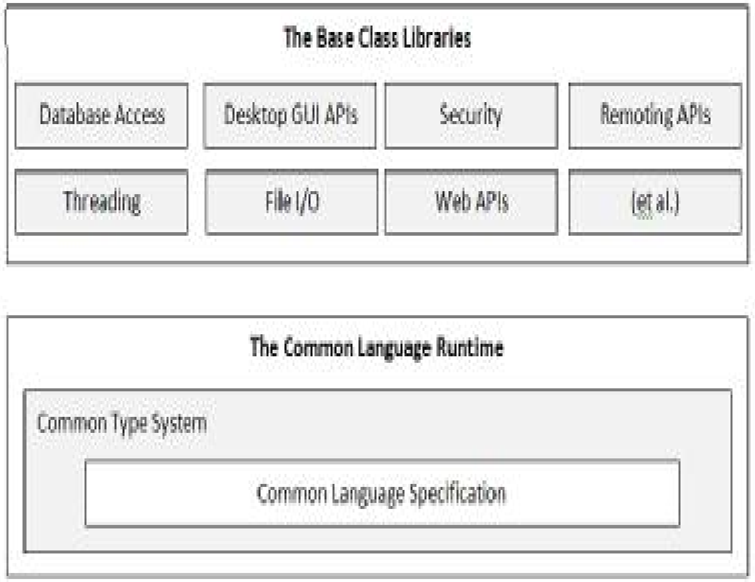


The Base Class Library

- BCL is a huge collection of reusable classes , interfaces, and value types whic can be used with any programming language which implements .NET
- All classes implemented in BCL are organized into namespaces

Example:

- System
- System.IO
- System.Collections
- System.Data
- System.XML,...



The Base Class Libraries

Database Access | Desktop GUI APIs | Security | Remoting APIs

Threading | File I/O | Web APIs | (et al.)

The Common Language Runtime

Common Type System

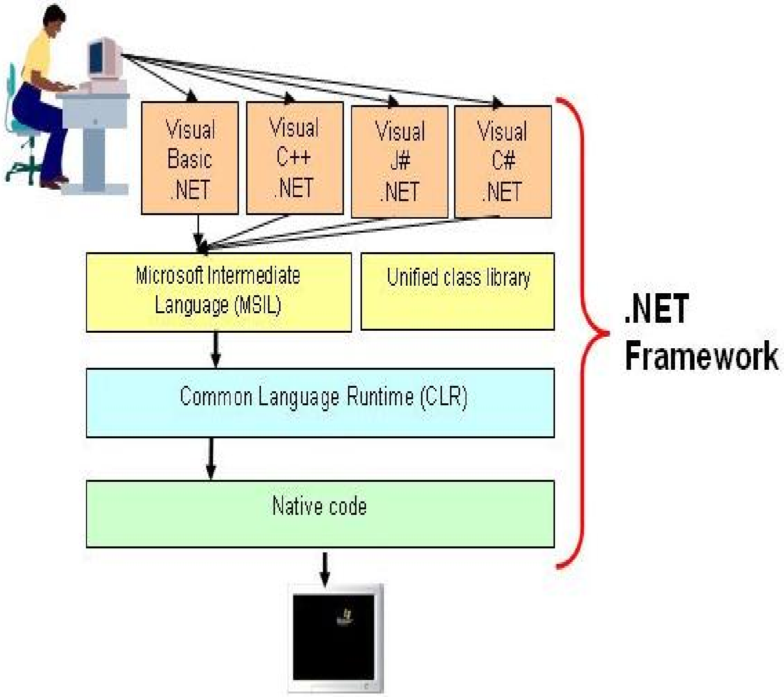
Common Language Specification

The Base Class Library

- BCL define types that can be used to build any type of software application:
  - ASP.NET: Provides a set of classes to design a website.
  - WCF: Provides a set of classes to design forms for the web pages similar to the HTML forms
  - Web Services: This includes a set of classes to design applications that can be accessed using a standard set of protocols
  - Windows Forms: Provides a set of classes to design forms for windows-based applications
  - ADO.NET: Provides classes to interact with databases.
  - XML Classes: Enables XML manipulation, searching and translations
  - Base Framework Classes: These classes provide basic functionality such as input/output, string manipulation, securitv management, network communication and so on

The common language runtime (CLR)

- The CLR provides the appearance of an application virtual machine so that programmers need not consider the capabilities of the specific CPU that will execute the program



The common language runtime (CLR)

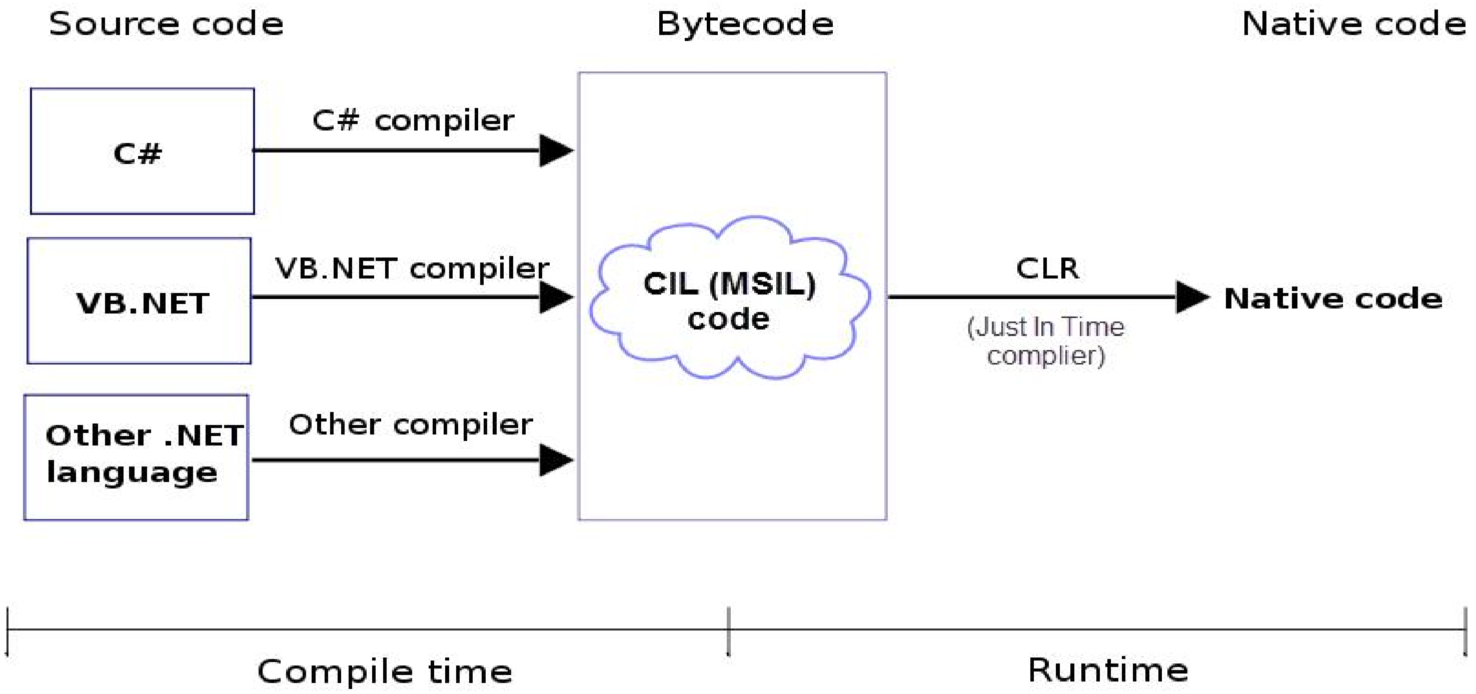
- The common language runtime (CLR) is the backbone of .NET Framework. It performs various functions such as:
  - Memory management
  - Code execution
  - Error handling
  - Code safety verification
  - Garbage collection

The common language runtime (CLR)

- CLR:
  - Common Language Specification (CLS): These are a set of rules that any .NET language should follow to create applications that are interoperable with other languages.
  - Common Type System (CTS): Describes how data types are declared, used and managed in the runtime and facilitates the use of types across various languages.

The common language runtime (CLR)

- CIL (Common Intermediate Language) or Microsoft Intermediate Language (MSIL) or IL



The common language runtime (CLR)

- Compiling to the Common Intermediate Language:
  - The compiler for a .NET language takes a source code file and produces an output file called an assembly
  - An assembly is either an executable or a DLL.
  - The code in an assembly isn’t native machine code but an intermediate language called the Common Intermediate Language (CIL).
- Compiling to Native Code and Execution
  - The executable code in the assembly is compiled to native code by the JIT compiler only as it’s needed

Lession 2: C# Fundamentals

- Introduction to C#
- IDE
- First program C#
- Variable, constant, data type
- Input and output with Console
- C# programming constructs

Introduction to C#

- C# (C Sharp) is a simple and powerful object-oriented language.
- C# can be used to create various types of applications:
  - Web applications
  - Windows graphical user interface (GUI) applications
  - Console-based applications

Introduction to C#

- C# versions:

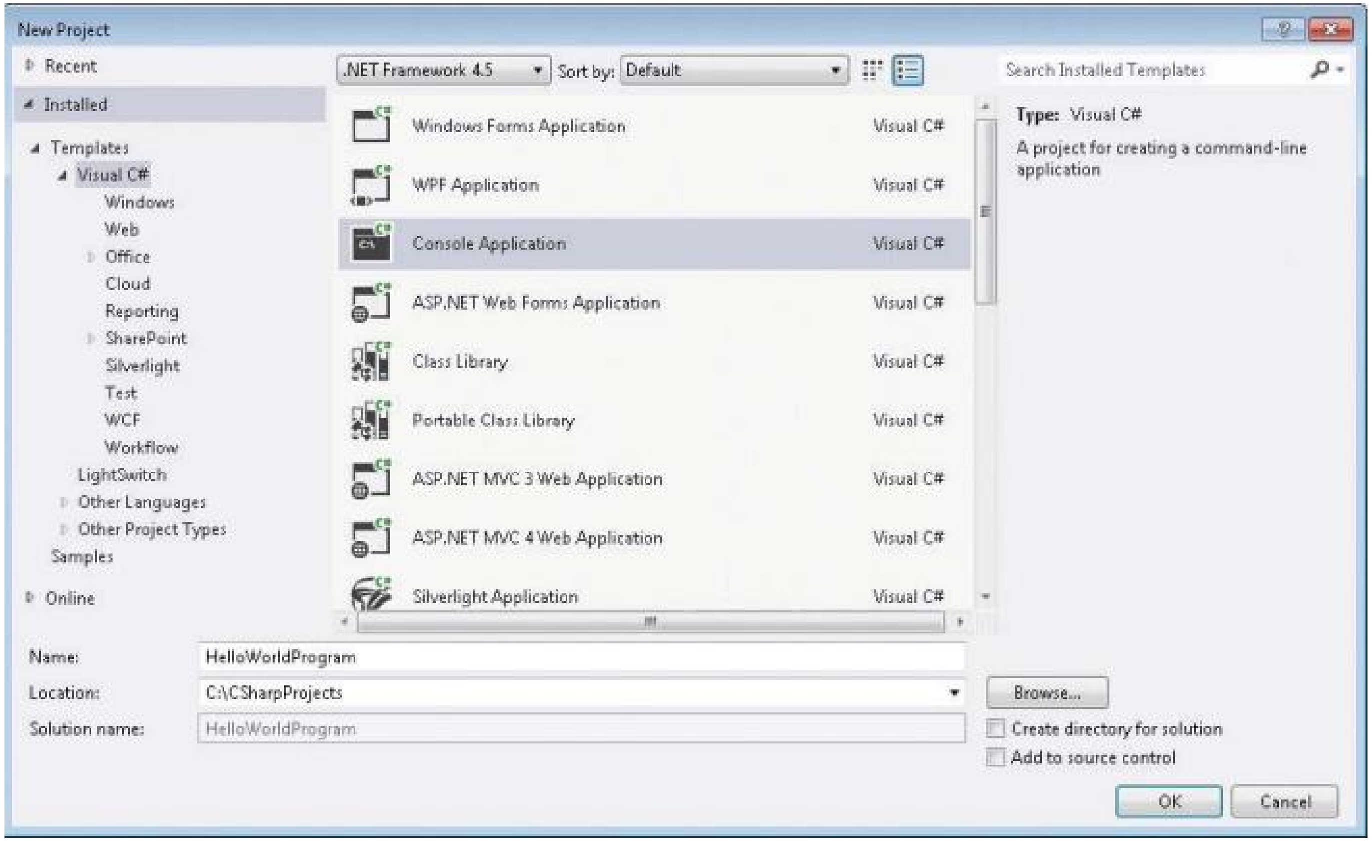
Version	.NET Framework	Visual Studio	Features Focus
C# 1.0	Framework 1.0/1.1	Visual Studio .NET 2002	C# basic
C# 2.0	.NET Framework 2.0	Visual Studio 2005	Generics
C# 3.0	.NET Framework 3.0\3.5	Visual Studio 2008	LINQ
C# 4.0	.NET Framework 4.0	Visual Studio 2010	Named and Optional Parameters
C# 5.0	.NET Framework 4.5	Visual Studio2012/2013	Async
C# 6.0	.NET Framework 4.6	Visual Studio 2013/2015	Expression Bodied Methods Auto-property initializer nameof Expression Primary constructor Await in catch block Exception Filter String Interpolation

IDE

- .NET Framework
- Integrated Development Environment (IDE):
  - An IDE is a tool that helps you write your programs.
  - Visual Studio is an IDE provided by Microsoft to write the code in languages such as C#, F#,VB.NET, etc.
  - Use Visual Studio 2010/2012/2013 based on the C# version you want to work with.
  - Visual Studio is a licensed product, so you must buy a license for commercial use. However, Visual Studio Express is free for learning purpose

First program

- Create a Console Application



New Project
Recent
Installed
Templates
Visual C#
Windows
Web
Office
Cloud
Reporting
SharePoint
Silverlight
Test
WCF
Workflow
LightSwitch
Other Languages
Other Project Types
Samples
Online
Search Installed Templates
Windows Forms Application Visual C#
WPF Application Visual C#
Console Application Visual C#
ASP.NET Web Forms Application Visual C#
Class Library Visual C#
Portable Class Library Visual C#
ASP.NET MVC 3 Web Application Visual C#
ASP.NET MVC 4 Web Application Visual C#
Silverlight Application Visual C#
Browse...
OK
Cancel

.NET Framework 4.5

Sort by: Default

Type: Visual C#

A project for creating a command-line application

Name: HelloWorldProgram

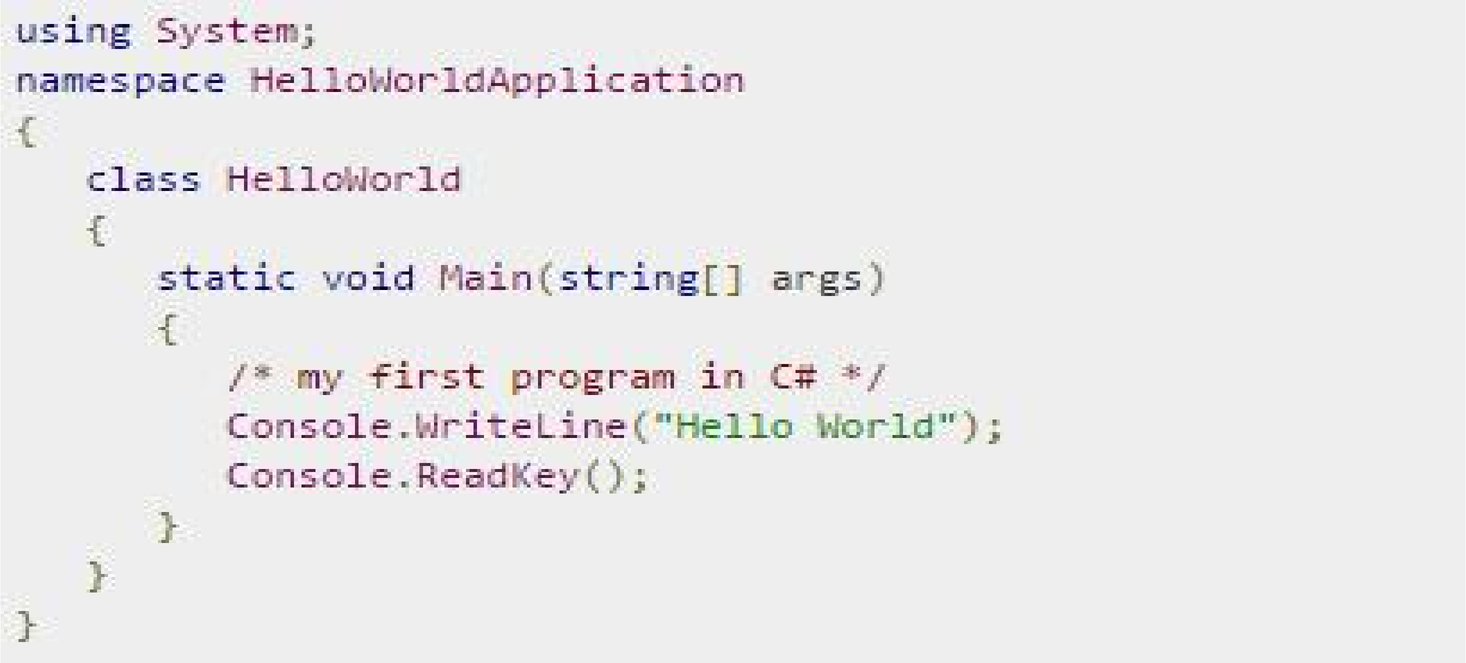
Location: C:\CSharpProjects

Solution name: HelloWorldProgram

□ Create directory for solution

□ Add to source control

First program



```
using System;
namespace HelloWorldApplication
{
    class HelloWorld
    {
        static void Main(string[] args)
        {
            /* my first program in C# */
            Console.WriteLine("Hello World");
            Console.ReadKey();
        }
    }
}
```

- Complie Run C# program:
  - Press Ctrl + F5 or
  - click the "Debug" menu -> "Start Without Debugging“
  - Click Run or F5

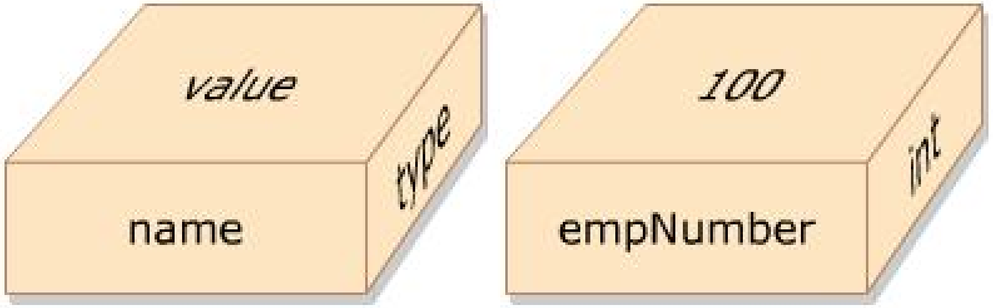
First program

- Comment: 3 types
- A solution file may consist of one or more projects

  The solution file ends with .sln extension.
- C# is case sensitive.
- The program execution starts at the Main method
- Unlike Java, program file name could be different from the class name

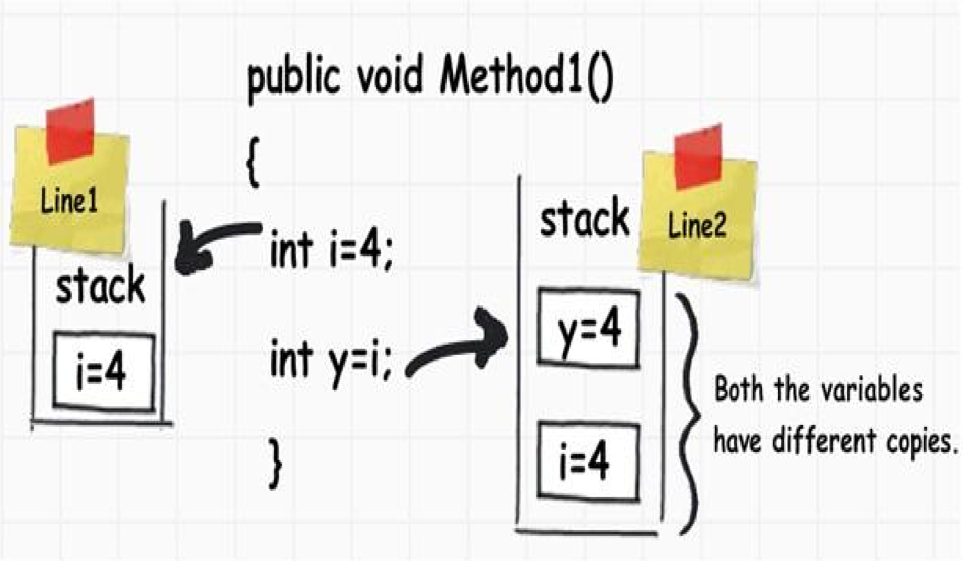
Variables

- A variable is an entity whose value can keep changing.

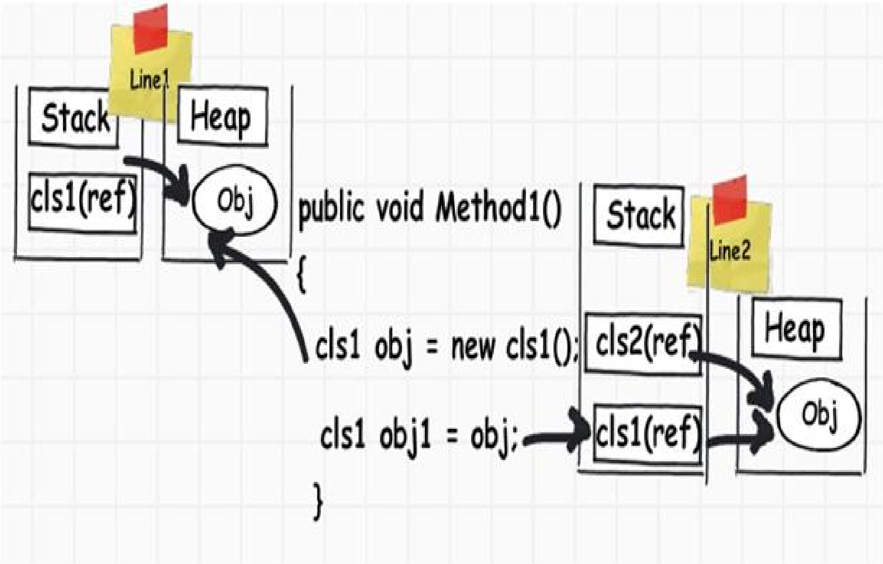


Data type

- Divided two type:
  - Value types:
  - Reference types



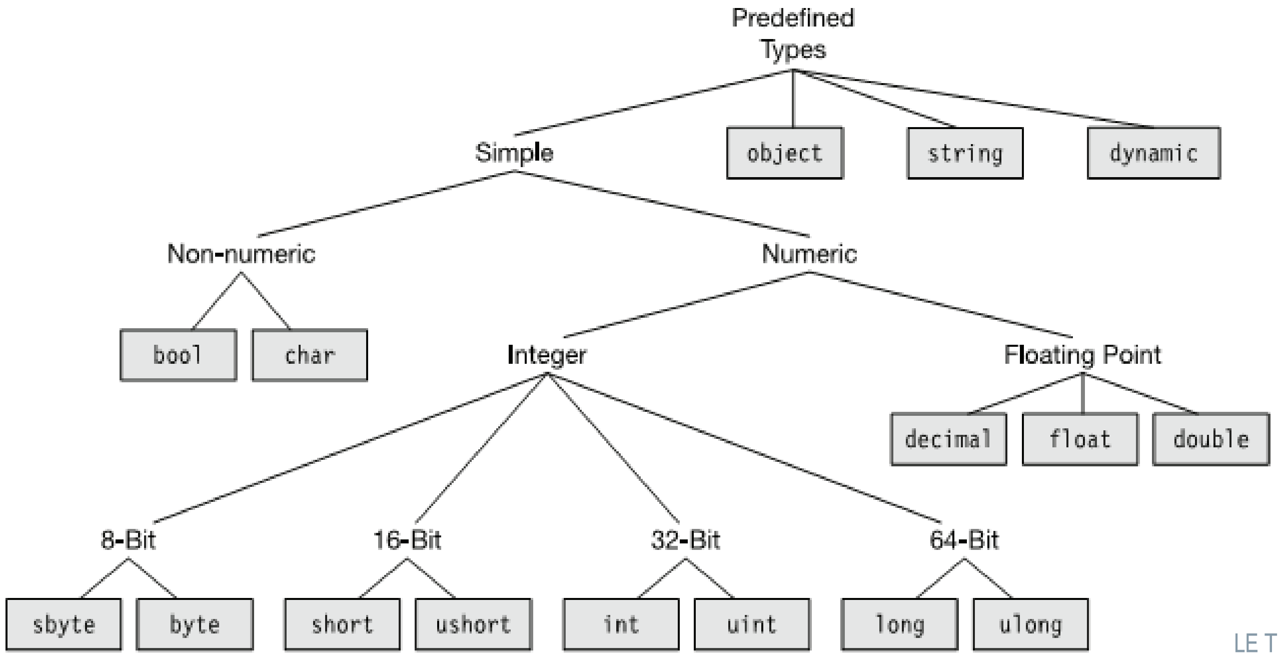
```
public void Method1()
{
int i=4;
int y=i;
}
```



```
public void Method1()
{
cls1 obj = new cls1();
cls1 obj1 = obj;
}
```

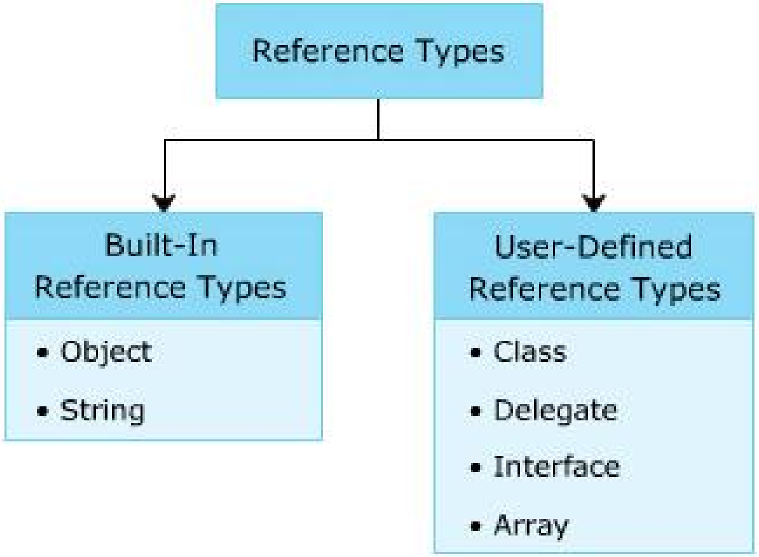
Data type

- Value type: Variable of value types store actual values. These values are store in a stack.



Data type

- Reference type: Variables of reference type store the memory address of other variables in a heap.



Reference Types

Built-In Reference Types

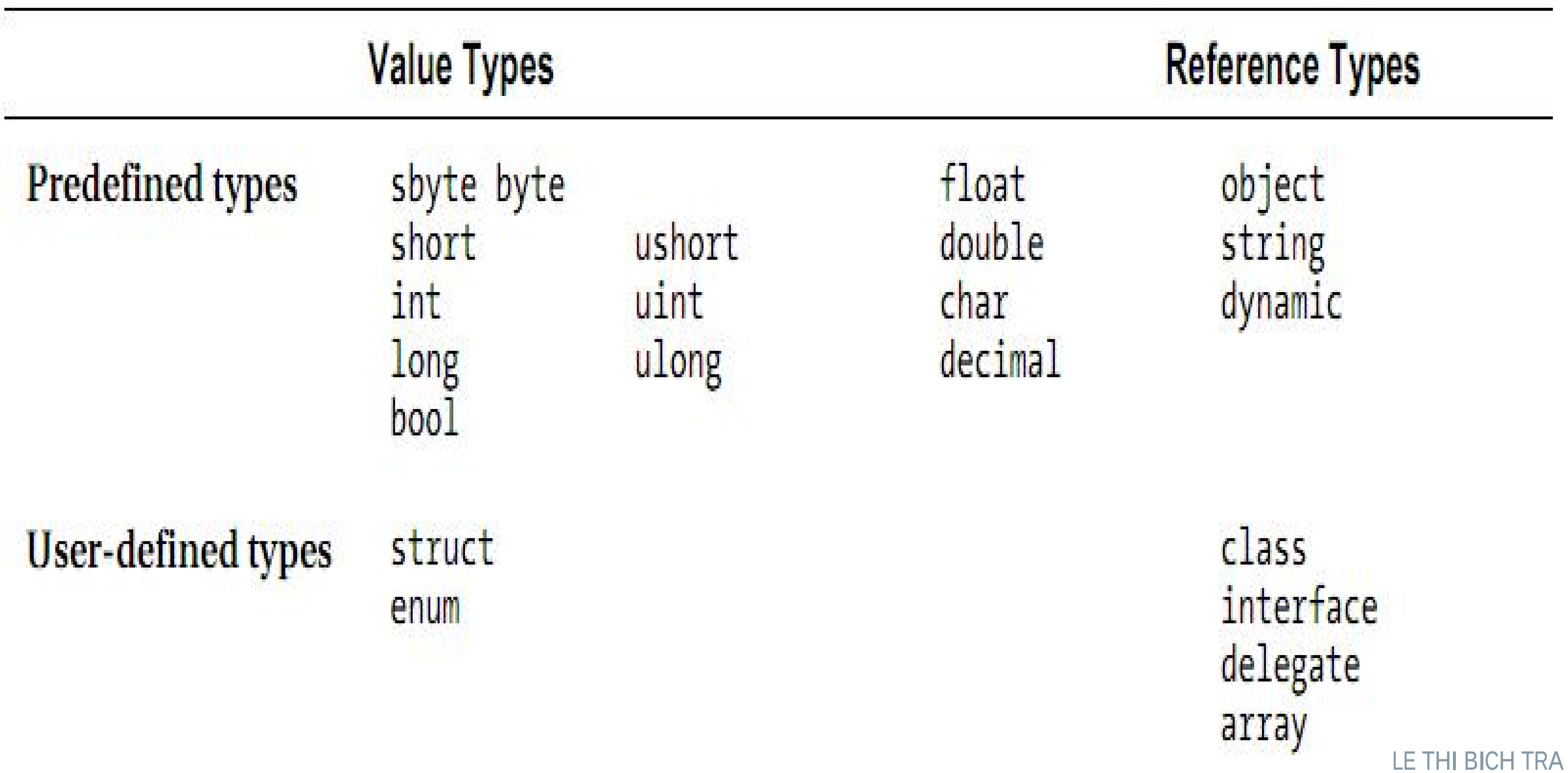
- Object
- String

User-Defined Reference Types

- Class
- Delegate
- Interface
- Array

Data type

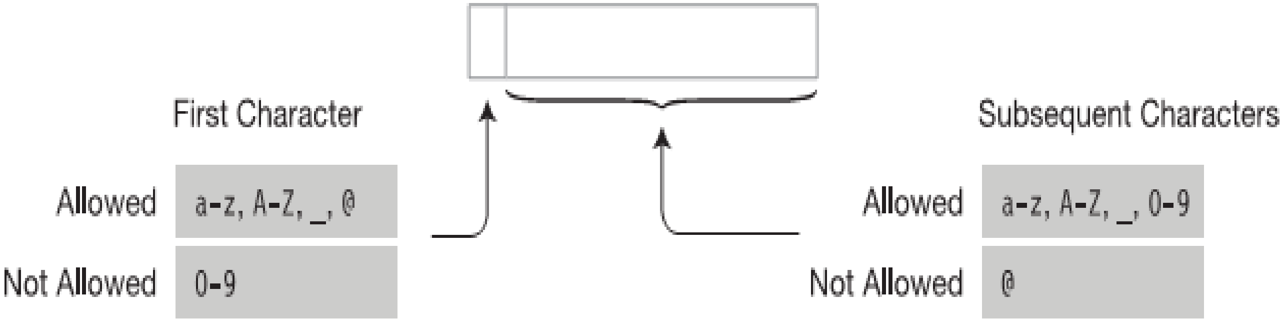
› Categorizing the C# Types



	Value Types			Reference Types
Predefined types	sbyte byte short int long bool	 ushort uint ulong	float double char decimal	object string dynamic
User-defined types	struct enum			class interface delegate array

Declare and use variables

- Syntax:
  <data type> <variable name> = <value>;
- Identifiers:



First Character		Subsequent Characters	
Allowed	a-z, A-Z, _, @	Allowed	a-z, A-Z, _, 0-9
Not Allowed	0-9	Not Allowed	@

- Must assign an initial value to variable before use.

Nullable type

- Value types never be assigned the value of null
- To define a nullable variable type, the question mark symbol (?) is suffixed to the underlying data
- Ex:

```
  int? nullableInt = 10;
  double? nullableDouble = 3.14;
  bool? nullableBool = null;
  char? nullableChar = 'a';
  int?[] arrayOfNullableInts = new int?[10];

// string? s = "oops"; // Error! Strings are reference types!
```

Output with Console

- Syntax:
  - Console.Write(“<data>” + variables);
  - Console.WriteLine(“<data>” + variables);
- Format Console
  - Use place holders {0}, {1},...
  - Use place holder and alignment: {0,10}, {1,-10}
  - Use Format characters: {0:f2}, {0,10:f2}, {0,10:c}
  - Use String.Format

Input with Console

- Syntax:
  - Console.Read() - Reads a single character
  - Console.ReadLine() - Reads a line of strings
- Input a string
- Input a numeric value
  - Parse
  - Convert
  - TryParse

C# programming constructs

- Selection construct:
  - If
  - If/Else
  - Ternary operator ? :
  - Switch
- Loop construct:
  - for
  - foreach…in
  - while
  - do/while

Exercises

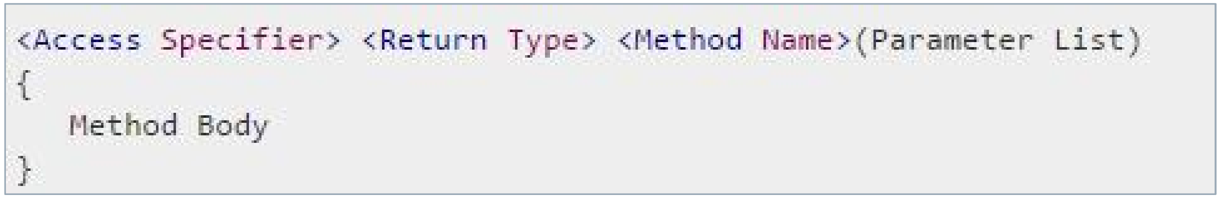
- Apply C# constructs

Lession 3: Methods

- Parameter
- Optional method
- Named method
- Overloading method

Method

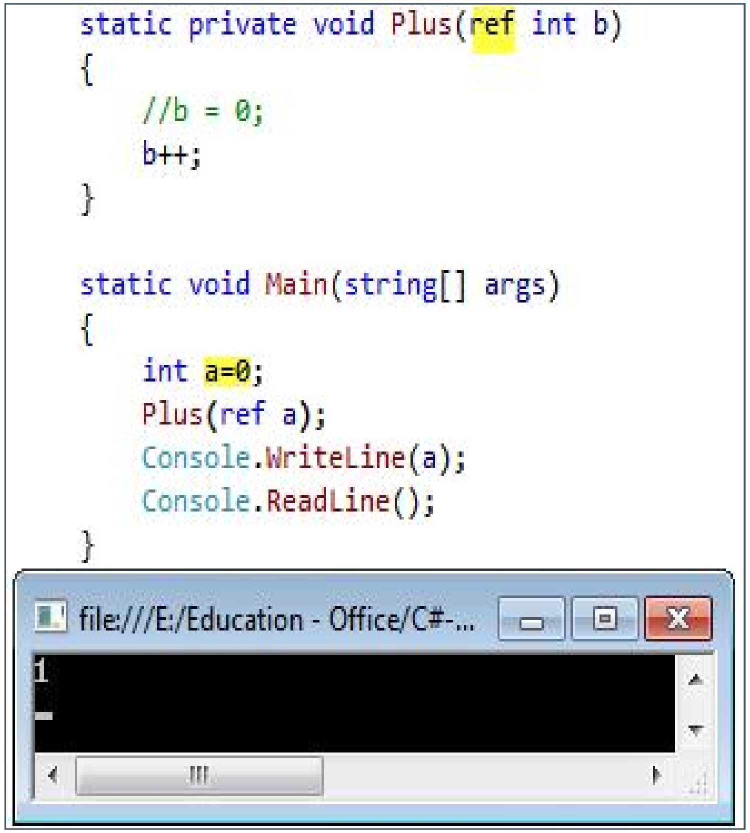
- Syntax:



```
<Access Specifier> <Return Type> <Method Name>(Parameter List)
{
   Method Body
}
```

- Parameters can be passed to a method by these following ways:
  - Value: passed by value, meaning the called method receives a copy of the original data.
  - Out: passed by reference
  - Ref:passed by reference
  - Params: end in a variable number of arguments as a single logical parameter.

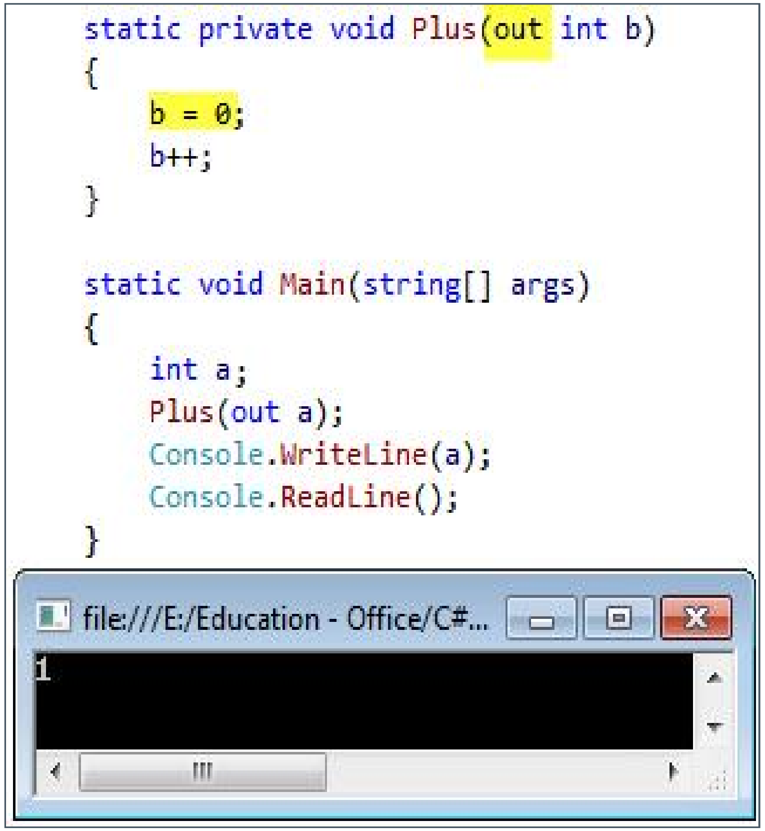
Passing parameters



```
static private void Plus(ref int b)
{
    //b = 0;
    b++;
}

static void Main(string[] args)
{
    int a=0;
    Plus(ref a);
    Console.WriteLine(a);
    Console.ReadLine();
}
```

```
1
_
```



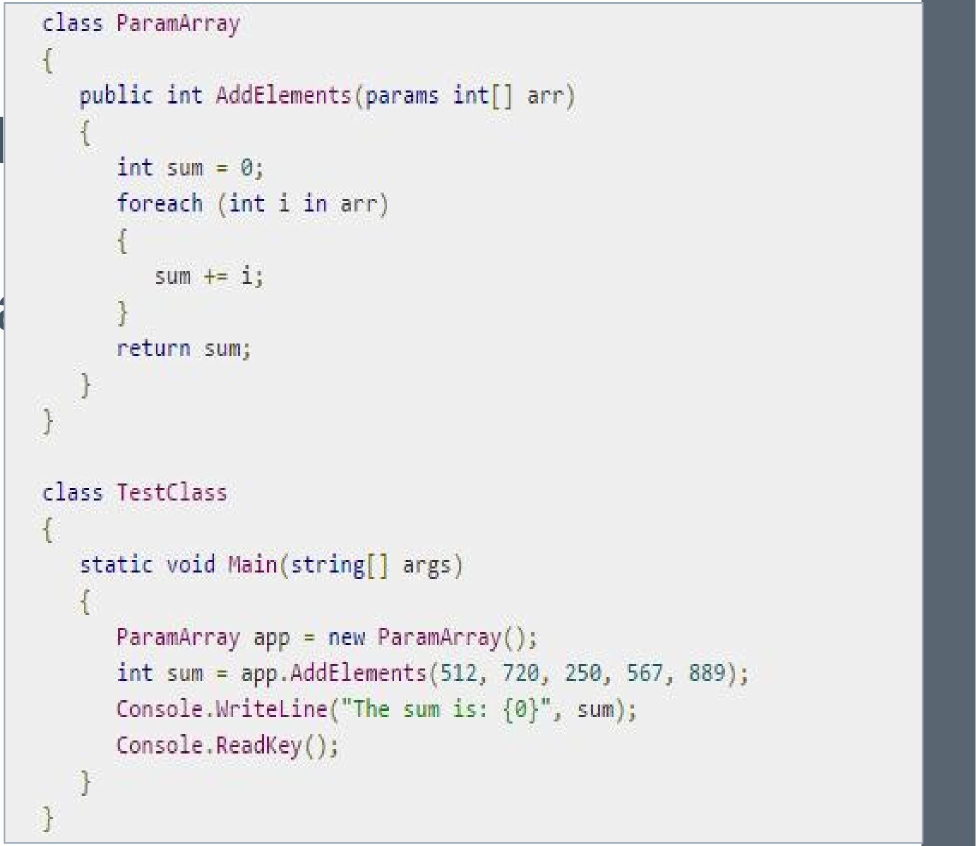
```
static private void Plus(out int b)
{
    b = 0;
    b++;
}

static void Main(string[] args)
{
    int a;
    Plus(out a);
    Console.WriteLine(a);
    Console.ReadLine();
}
```

```
1
```

Params modifier

- C# supports the use of paramete arrays using the params keyword
- Is used when you not sure of the number of arguments passed as a parameter

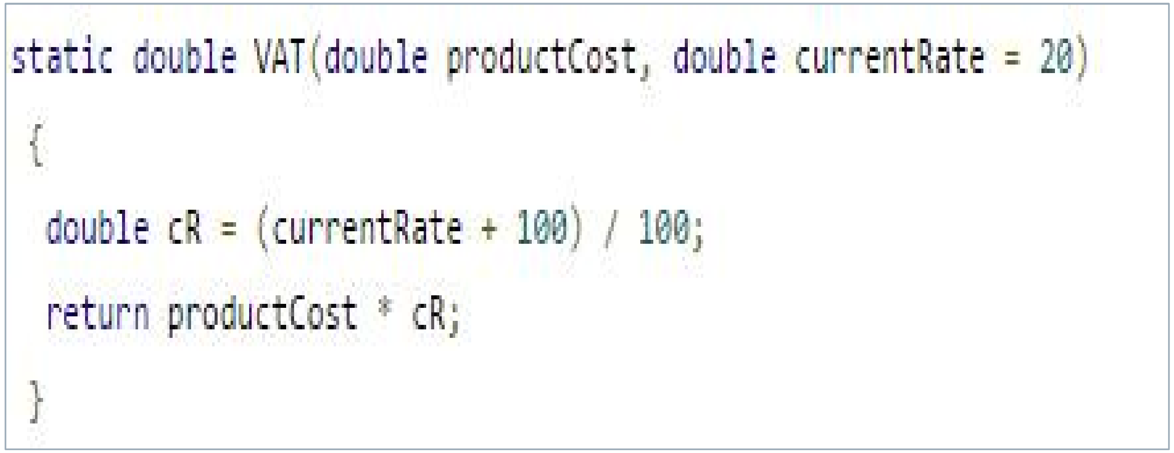


```
class ParamArray
{
    public int AddElements(params int[] arr)
    {
        int sum = 0;
        foreach (int i in arr)
        {
            sum += i;
        }
        return sum;
    }
}

class TestClass
{
    static void Main(string[] args)
    {
        ParamArray app = new ParamArray();
        int sum = app.AddElements(512, 720, 250, 567, 889);
        Console.WriteLine("The sum is: {0}", sum);
        Console.ReadKey();
    }
}
```

Optional Parameters

- Allows the caller to invoke a method while omitting arguments deemed unnecessary.
- Optional Parameter if not passed will take default value
- Optional Parameter must define at the end of the any required parameter



```
static double VAT(double productCost, double currentRate = 20)
 {
  double cR = (currentRate + 100) / 100;
  return productCost * cR;
 }
```

Named parameters

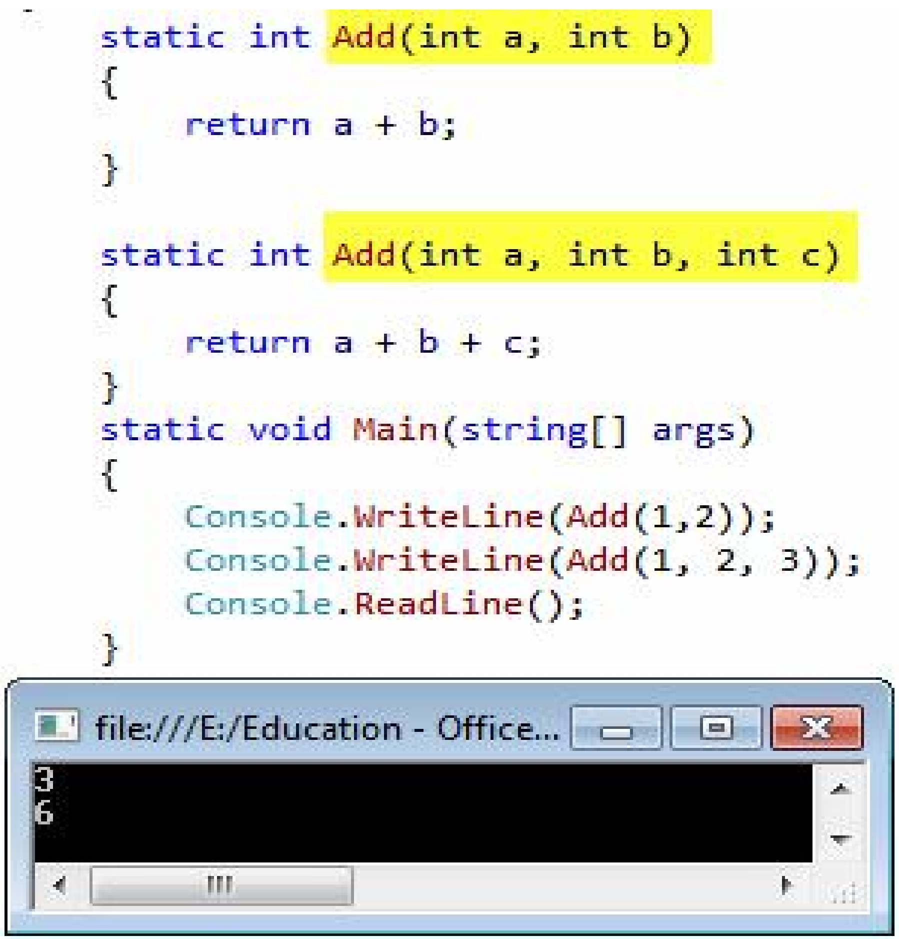
- Named arguments allow you to invoke a method by specifying parameter values rather than passing parameters by position.
- Ex:

```
Person person = new Person("John", "Smith", new DateTime(1970, 1, 1));
Person person = new Person(firstName: "John", lastName: "Smith",
dateOfBirth: new DateTime(1970, 1, 1));
```

- Invoke a method using positional parameters, they must be listed before any named parameters.
- If you have a method that defines optional arguments, this feature can actually be really helpful

Method overloading

› Define a set of identically named methods that differ by the number (or type) of parameters



```
static int Add(int a, int b)
{
    return a + b;
}

static int Add(int a, int b, int c)
{
    return a + b + c;
}
static void Main(string[] args)
{
    Console.WriteLine(Add(1,2));
    Console.WriteLine(Add(1, 2, 3));
    Console.ReadLine();
}
```

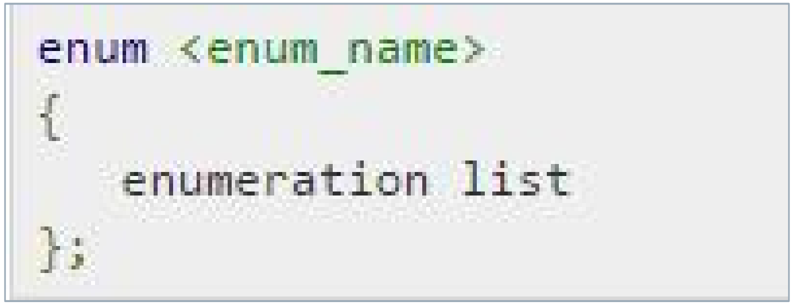
```
3
6
```

Lession 4: Enumeration type

- What is enumeration type?
- Syntax
- Comon methods

Enumeration type

- Enumeration is a value data type. The enum is used to declare a set of named integer constants.
- enum keyword allows you to define a custom set of name/value pairs.
- Syntax:

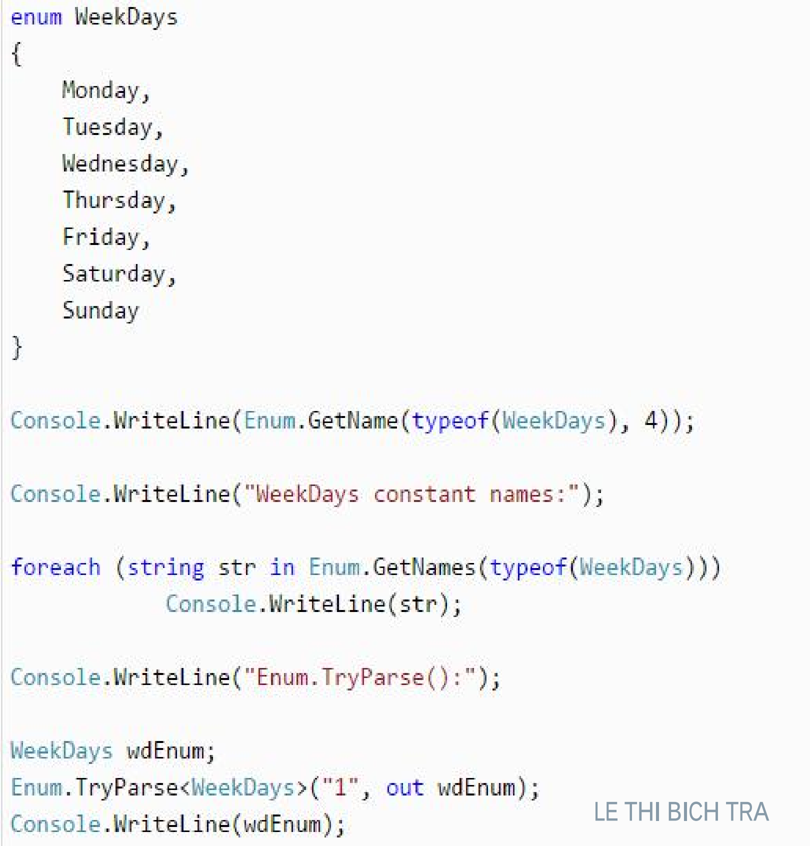


```
enum <enum_name>
{
    enumeration list
};
```

- The *enum_name* specifies the enumeration type name.
- The *enumeration list* is a comma-separated list of identifiers.

Enumeration type

› Enum is an abstract class that includes static helper methods to work with enums.
  - – Format
  - – GetName
  - – GetNames
  - – GetValues
  - – Object Parse(type,string)
  - – Bool TryParse(string,out Tenum)



```
enum WeekDays
{
    Monday,
    Tuesday,
    Wednesday,
    Thursday,
    Friday,
    Saturday,
    Sunday
}

Console.WriteLine(Enum.GetName(typeof(WeekDays), 4));

Console.WriteLine("WeekDays constant names:");

foreach (string str in Enum.GetNames(typeof(WeekDays)))
            Console.WriteLine(str);

Console.WriteLine("Enum.TryParse():");

WeekDays wdEnum;
Enum.TryParse<WeekDays>("1", out wdEnum);
Console.WriteLine(wdEnum);
```

Exercises:

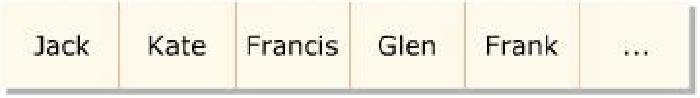
- Exercises about methods, enumeration type.

Lession 5: Array

- What is enumeration type?
- Introduction
- Declare and initialize an array
- Characteristics
- Array Class

Introduction

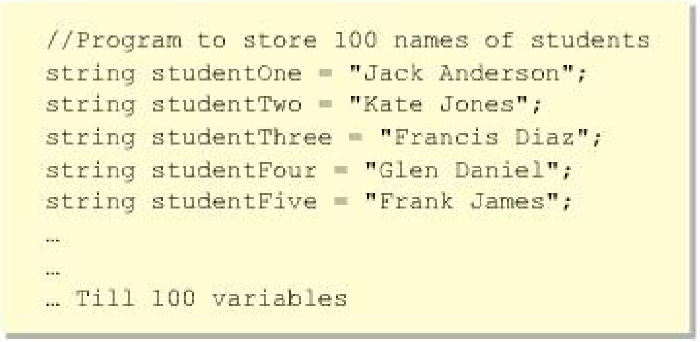
- An array always stores values of a single data type.
- Each value is referred to as an element.



Jack	Kate	Francis	Glen	Frank	...

Array of 100 Names

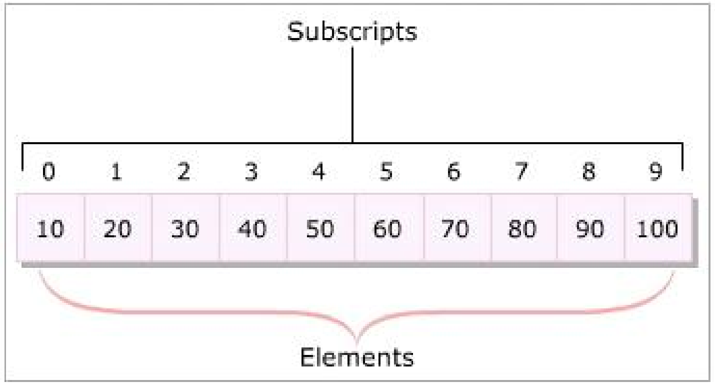
Efficient Memory Utilization



```
//Program to store 100 names of students
string studentOne = "Jack Anderson";
string studentTwo = "Kate Jones";
string studentThree = "Francis Diaz";
string studentFour = "Glen Daniel";
string studentFive = "Frank James";
…
…
… Till 100 variables
```

100 Variables Storing Names

Inefficient Memory Utilization



Subscripts

0	1	2	3	4	5	6	7	8	9
10	20	30	40	50	60	70	80	90	100

Elements

Arrays

Declare an array

› Syntax:

```
<data type>[ ] arrayName= new <data type>[size]
```

› Or:

```
<data type>[ ] arrayName={value1,value2,...,valueN}
```

ü Defaut values:

ü Characteristics:

Iterate through arrays

- Use for or foreach loop

```
for (int i = 0; i < arr.Length; i++)
```

```
foreach (KiểuDL s in arr)
```

Array Class

- The Array class is the base class for all the arrays in C#. It is defined in the System namespace
- The Array class provides various properties and methods to work with arrays.
- Properties of the Array class:
  - Length
  - IsReadOnly
  - IsFixedSized
  - Rank

Array Class

- Methods of the Array class:
  - Clear():
  - Copy(Array, Array, Int32)
  - IndexOf(Array,Object):
  - Reverse(Array):
  - SetValue(Int32)
  - GetValue(Int32):
  - Sort(Array):

Exercises:

- Use arrays

Thank You !