MCSE 541: Web Computing and Mining

.Net Introduction

Prof. Dr. Shamim Akhter
shamimakhter@iubat.edu

@Text Book: Chapter 6

About Me

Experience	Time	Institute	
Bachelor (BSc)	1998-2001	American International University-Bangladesh	
Masters (MSc)	2003-2005	Asian Institute of Technology Thailand	
Doctorate (PhD)	2006-2009	Tokyo Institute of Technology Japan	
Post Doctoral Researcher	2009-2011	JSPS-NII Japan	
Assistant Prof.	2005-2014	American International University-Bangladesh	
Contact Asst. Prof.	2013-2014	Thompson Rivers University (TRU), Kamloops, BC, CANADA	
Associate Prof.	2014-2019	East West University	
Professor	2019-	IUBAT	

People and Places

Prof. Dr. Shamim Akhter,

Professor, CSE, Stamford University, Bangladesh

https://www.linkedin.com/in/prof-dr-md-shamim-akhter-a3a96666/

profshamimakhter@gmail.com

Please Call me : Prof. Shamim!, Professor, or Sir

Course Link:

https://classroom.google.com/c/NDk2NzE3Mzg5MTY4?cjc=jrdkuao

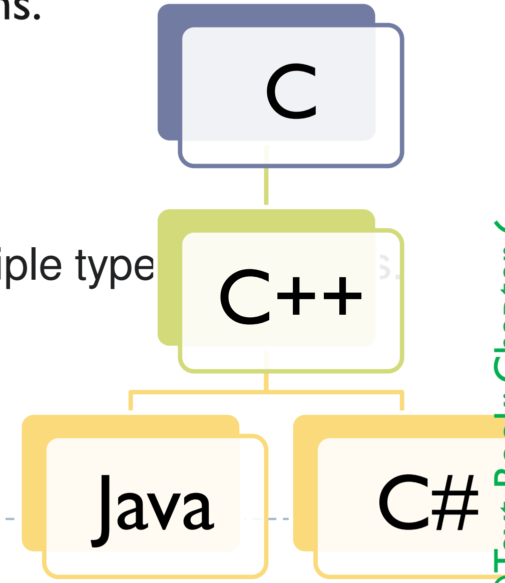
Class Schedule –Summer 2022

Day	9:00-10:00	10:00-11:00	11:00-12:30	12:30-14:00	14:00-15:30	15:30-17:00	17:00-18:30
Saturday	Web Data Mining		COA CSE233 Room: 503 Batch: 75A				
Sunday							
Monday		Class Meterials		Counsiling Hour	Research		
Tuesday							
Wednesday		Counsiling Hour				Class Meterials	COA CSE233 Room: 503 Batch: 75A
Thursday							

.Net and C#

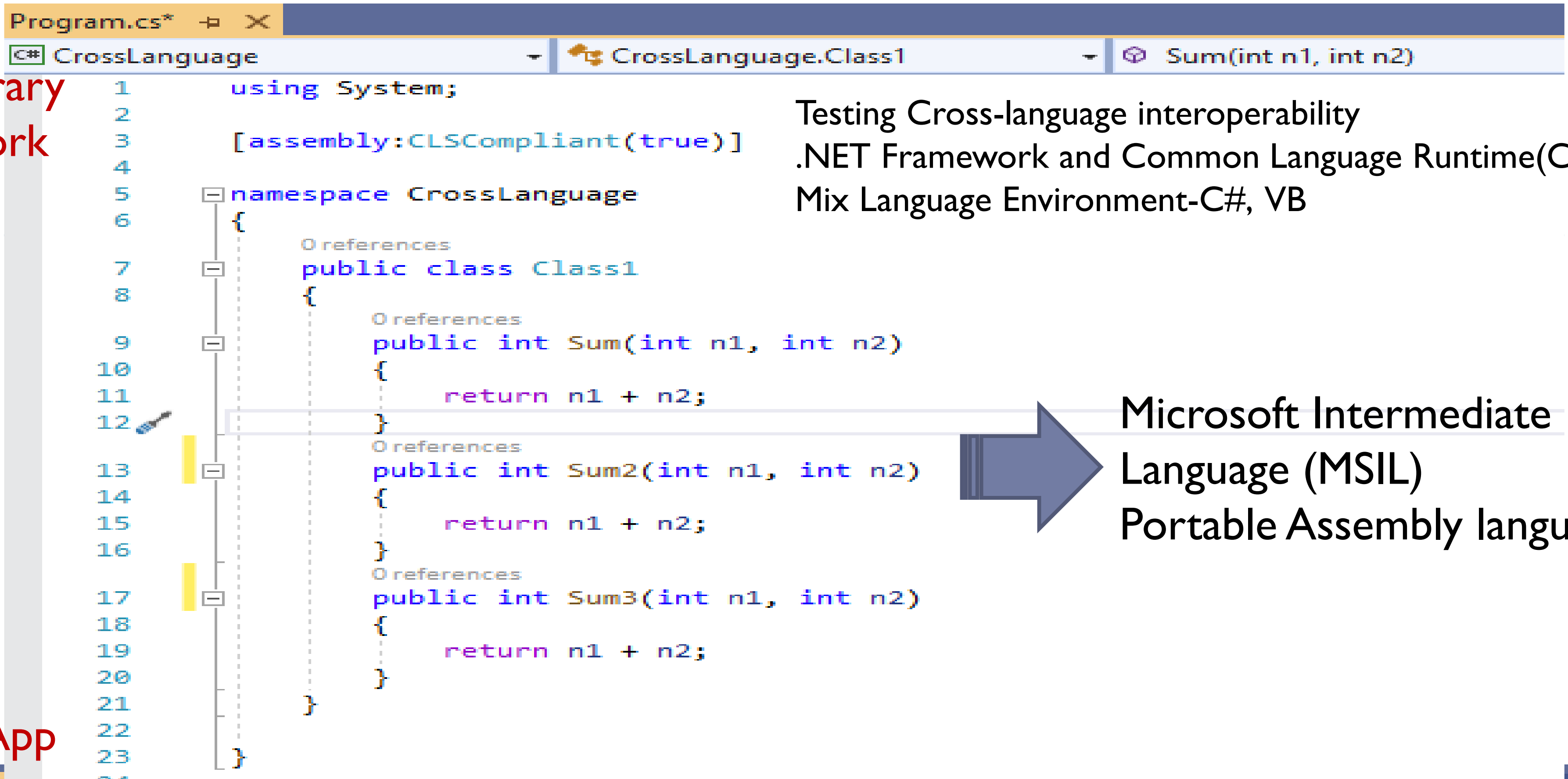
- .NET is a free, open-source development platform
  - for building many kinds of apps-Web apps, web APIs etc.
  - supports integrated development environments (IDEs), and other tools and supports three programming languages C#, F#, Visual Basic.

- Java has portability but
  - lacks cross-language interoperability
    - requires for large and distributed software system.
    - Assemblies take the form of executable (*.exe*) or dynamic link library (*.dll*) files, and are the building blocks of .NET applications. They provide the common language runtime with the information it needs to be aware of type implementations.
  - Not Fully Integrates with windows platforms
    - Universal Windows Platform (UWP) API @ Windows 10
      - allows developers to create apps that will potentially run on multiple type
      - Development support by C#, JavaScript but not Core Java
    - Windows Forms written C#
      - Write rich client applications for Labtop, Desktop or Tablet PCs



@Text Book: Chapter 6

ClassLibrary Framework

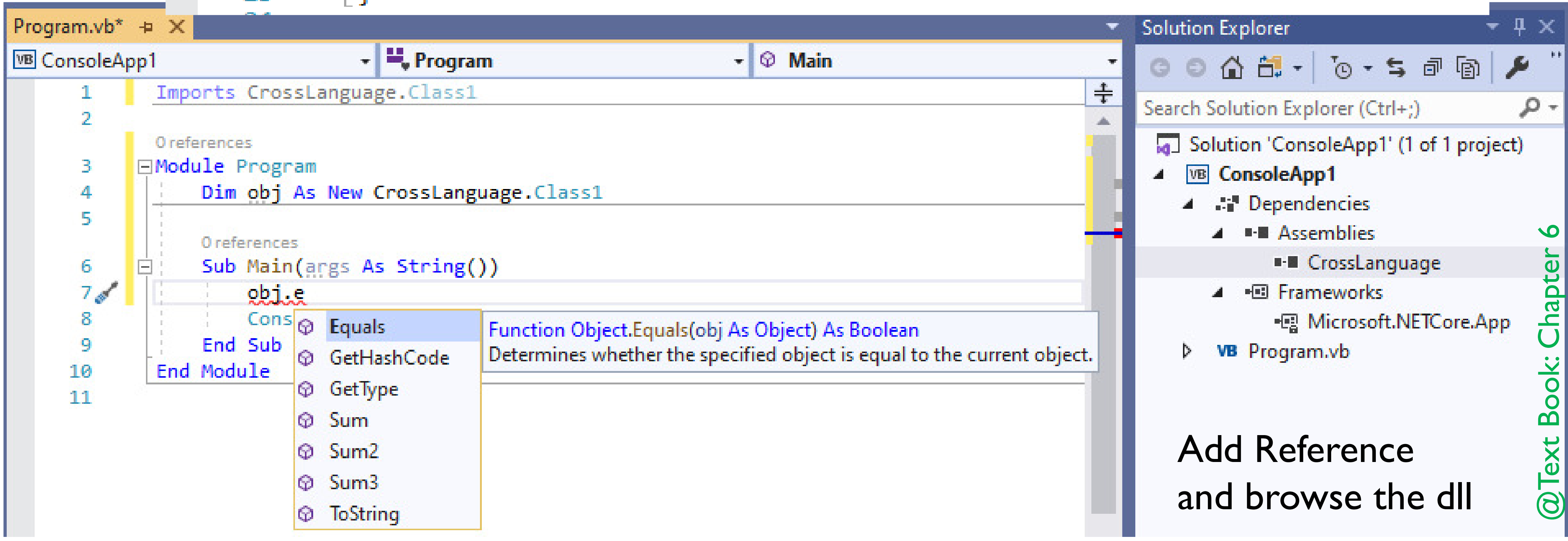


```
using System;

[assembly:CLSCompliant(true)]

namespace CrossLanguage
{
    public class Class1
    {
        public int Sum(int n1, int n2)
        {
            return n1 + n2;
        }
        public int Sum2(int n1, int n2)
        {
            return n1 + n2;
        }
        public int Sum3(int n1, int n2)
        {
            return n1 + n2;
        }
    }
}
```

Testing Cross-language interoperability
.NET Framework and Common Language Runtime(CLR)
Mix Language Environment-C#, VB

Microsoft Intermediate Language (MSIL)
Portable Assembly language

ConsoleApp

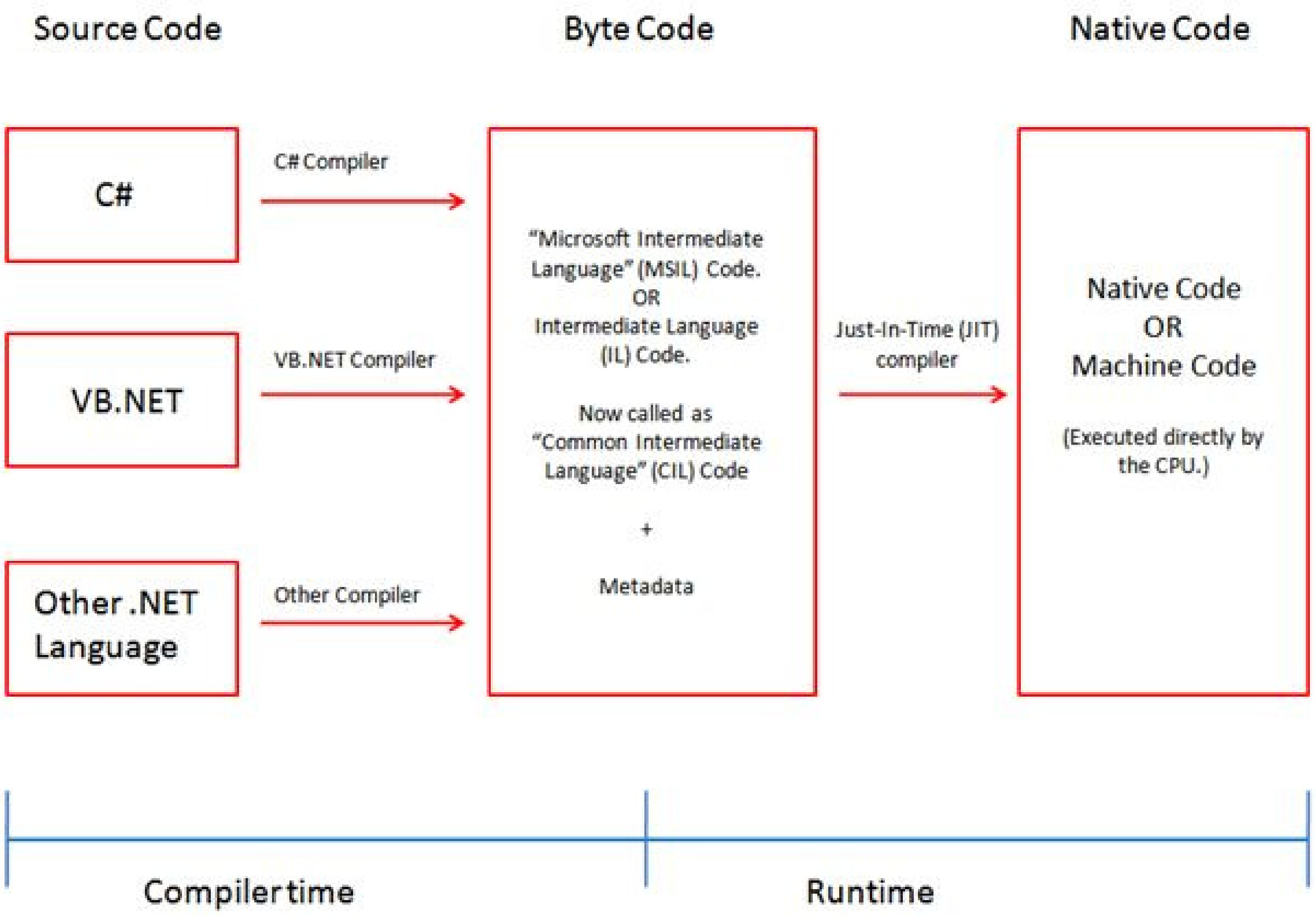


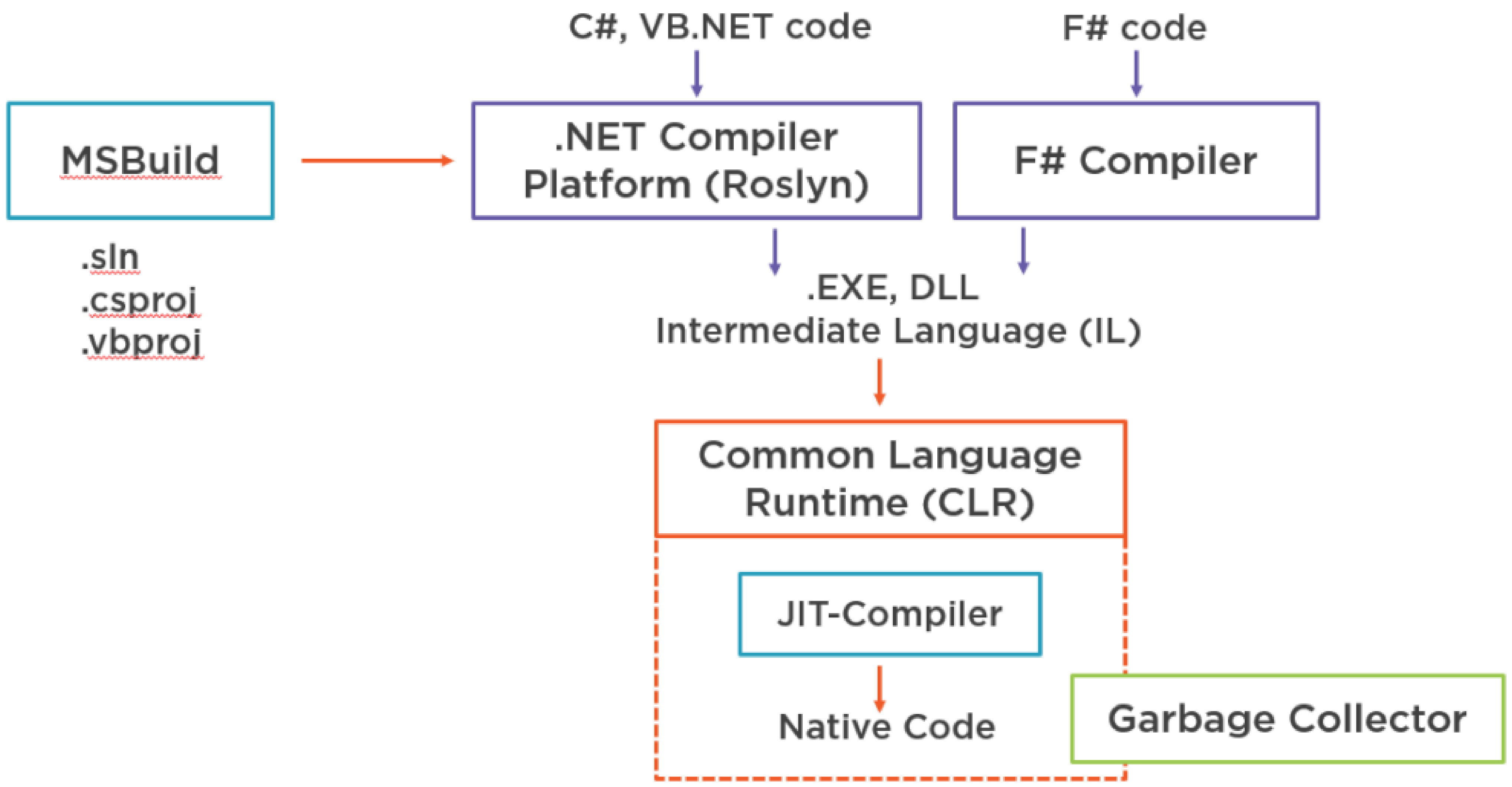
```
Imports CrossLanguage.Class1

Module Program
    Dim obj As New CrossLanguage.Class1

    Sub Main(args As String())
        obj.e
        Cons
    End Sub
End Module
```

Function Object.Equals(obj As Object) As Boolean
Determines whether the specified object is equal to the current object.

Add Reference and browse the dll

.NET Code Execution Process


Source Code
Byte Code
Native Code
C#
VB.NET
Other .NET Language
C# Compiler
VB.NET Compiler
Other Compiler
"Microsoft Intermediate Language" (MSIL) Code. OR Intermediate Language (IL) Code.
Now called as "Common Intermediate Language" (CIL) Code
+
Metadata
Just-In-Time (JIT) compiler
Native Code OR Machine Code
(Executed directly by the CPU.)
Compilertime
Runtime

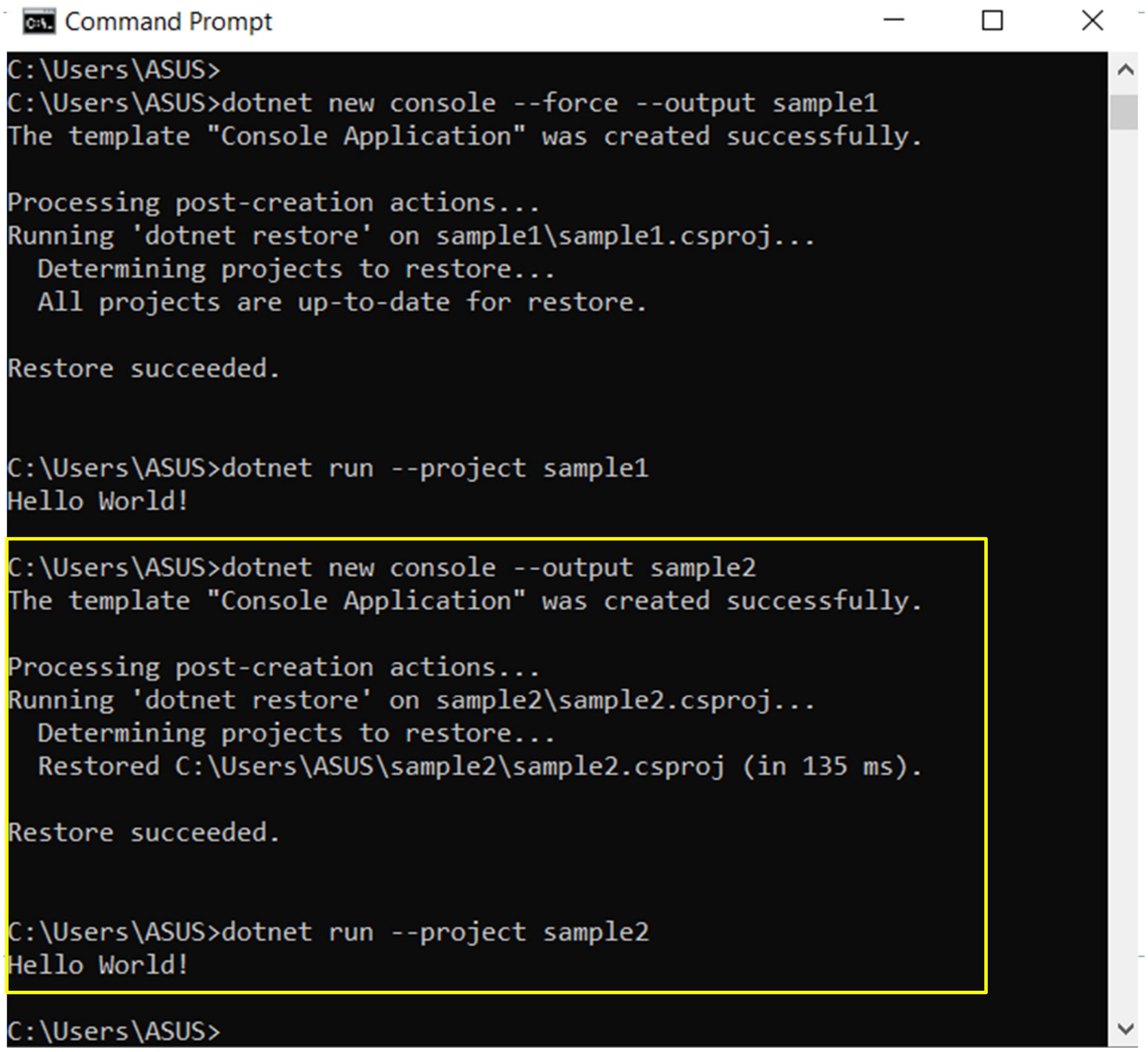

C#, VB.NET code
F# code
MSBuild
.NET Compiler Platform (Roslyn)
F# Compiler
.sln
.csproj
.vbproj
.EXE, DLL
Intermediate Language (IL)
Common Language Runtime (CLR)
JIT-Compiler
Native Code
Garbage Collector

SDK and Runtimes

- .NET SDK is a set of libraries and tools for developing and running .NET applications.

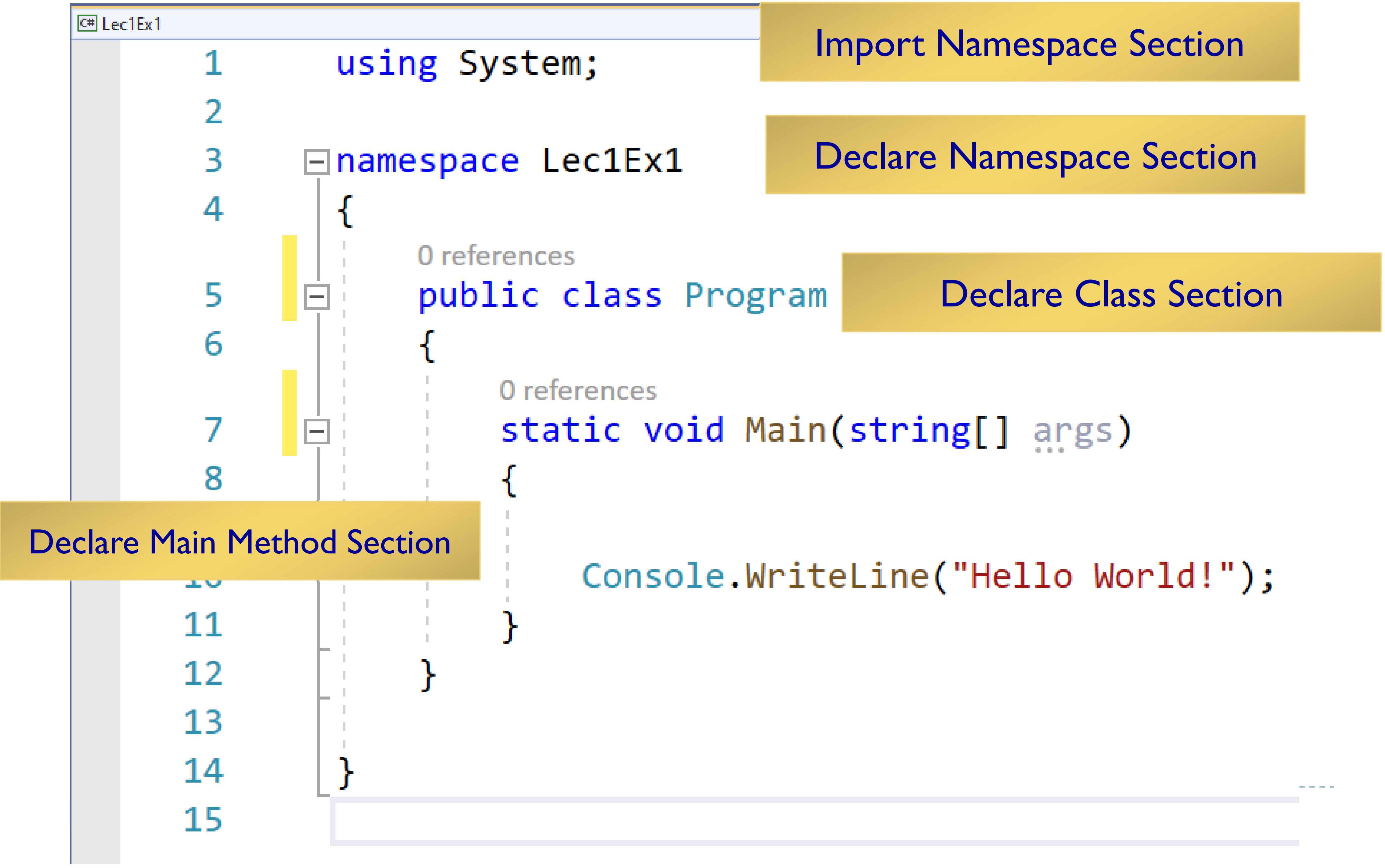
- SDK download includes the following components:
  - .NET CLI: Command-line tools that you can use for local development and continuous integration scripts.
  - dotnet driver: A CLI command that runs framework-dependent apps.
  - .NET runtime: Provides a type system, assembly loading, a garbage collector, native interop, and other basic services.
  - Runtime libraries. Provides primitive data types and fundamental utilities.

Command Line Tool



```
C:\Users\ASUS>
C:\Users\ASUS>dotnet new console --force --output sample1
The template "Console Application" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on sample1\sample1.csproj...
  Determining projects to restore...
  All projects are up-to-date for restore.

Restore succeeded.


C:\Users\ASUS>dotnet run --project sample1
Hello World!

C:\Users\ASUS>dotnet new console --output sample2
The template "Console Application" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on sample2\sample2.csproj...
  Determining projects to restore...
  Restored C:\Users\ASUS\sample2\sample2.csproj (in 135 ms).

Restore succeeded.


C:\Users\ASUS>dotnet run --project sample2
Hello World!

C:\Users\ASUS>
```

Basic Structure of A C# Program



```
using System;

namespace Lec1Ex1
{
    public class Program
    {
        static void Main(string[] args)
        {

            Console.WriteLine("Hello World!");
        }
    }

}
```

Namespace

- In C# namespace defines a declarative region
  - Remove the same name conflicts between user define class and .Net Library classes.
  - Within a namespace, we can declare another namespace, a class, an interface, a structure, an enumeration or a delegate.

Import Section:

Import statements are use to import the Base Class Libraries. This is similar to C #include statement.

Syntax: **using namespace;**

Example: **using System;**

```
System.Console.WriteLine("Hello World!");
```

If the required namespace is a member of another namespace we have to specify the parent and child namespaces separated by a dot.

- **using System.Data;**
- **using System.IO;**

a user-defined namespace is to be declared. In .NET applications, all classes related to the project should be declared inside one namespace.

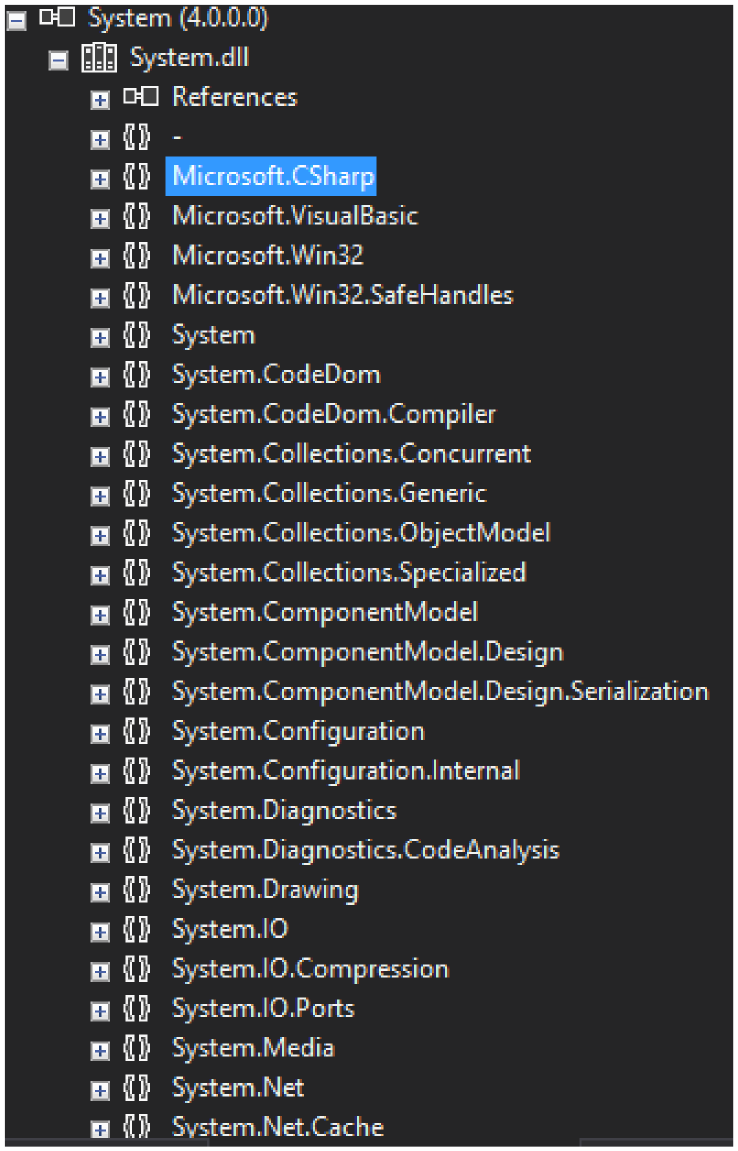
Syntax:

```
namespace NamespaceName
{
}
```

Generally, the namespace name will be the same as the project name.

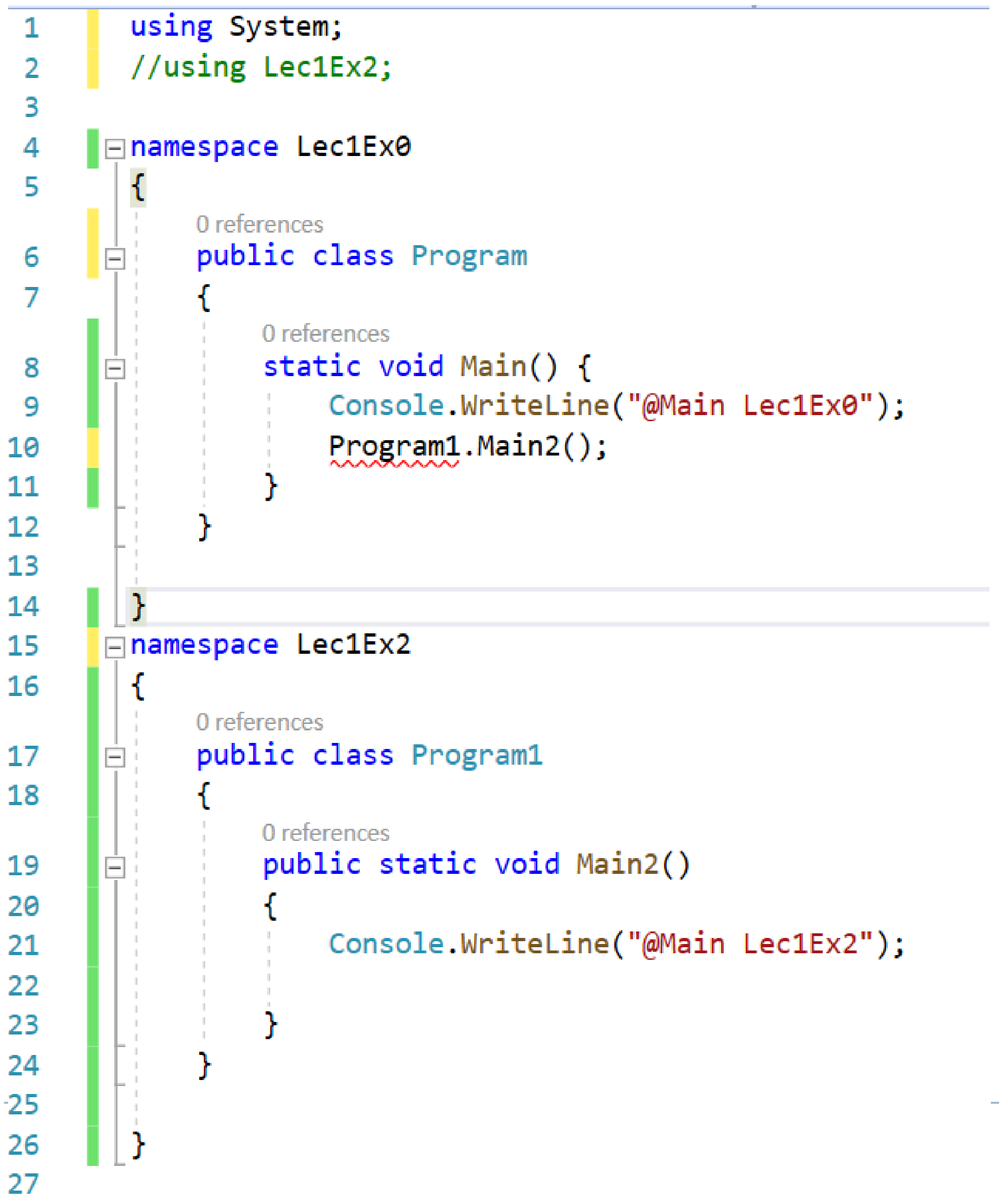
Namespace and Assembly

- Namespaces organize objects in an assembly.
- An assembly is a reusable, versionable and self-describing building block of a CLR application.
- Assemblies can contain multiple namespaces.
- Namespaces can contain other namespaces.
- An assembly provides a fundamental unit of physical code grouping.
- A namespace provides a fundamental unit of logical code grouping.



```
System (4.0.0.0)
  System.dll
    References
    {} -
    {} Microsoft.CSharp
    {} Microsoft.VisualBasic
    {} Microsoft.Win32
    {} Microsoft.Win32.SafeHandles
    {} System
    {} System.CodeDom
    {} System.CodeDom.Compiler
    {} System.Collections.Concurrent
    {} System.Collections.Generic
    {} System.Collections.ObjectModel
    {} System.Collections.Specialized
    {} System.ComponentModel
    {} System.ComponentModel.Design
    {} System.ComponentModel.Design.Serialization
    {} System.Configuration
    {} System.Configuration.Internal
    {} System.Diagnostics
    {} System.Diagnostics.CodeAnalysis
    {} System.Drawing
    {} System.IO
    {} System.IO.Compression
    {} System.IO.Ports
    {} System.Media
    {} System.Net
    {} System.Net.Cache
```

Using Namespace

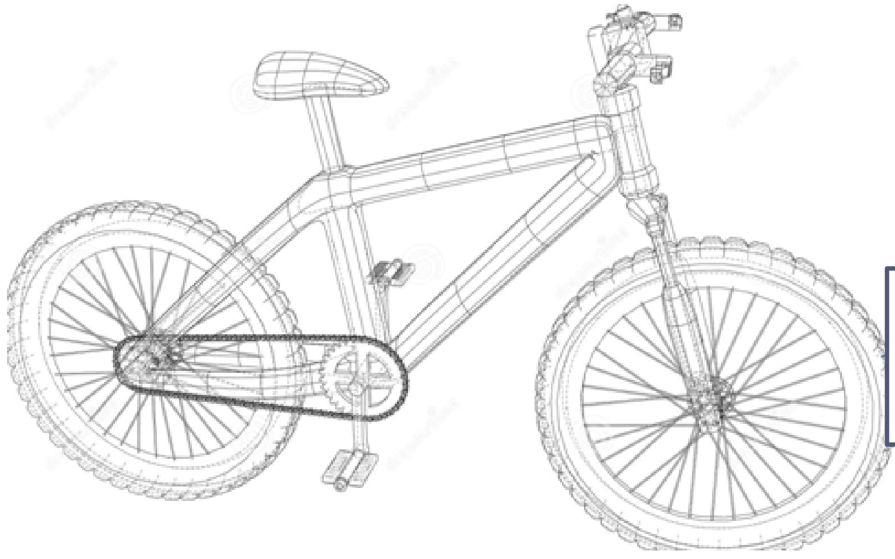


```
using System;
//using Lec1Ex2;

namespace Lec1Ex0
{
    0 references
    public class Program
    {
        0 references
        static void Main() {
            Console.WriteLine("@Main Lec1Ex0");
            Program1.Main2();
        }
    }

}
namespace Lec1Ex2
{
    0 references
    public class Program1
    {
        0 references
        public static void Main2()
        {
            Console.WriteLine("@Main Lec1Ex2");

        }
    }

}
```

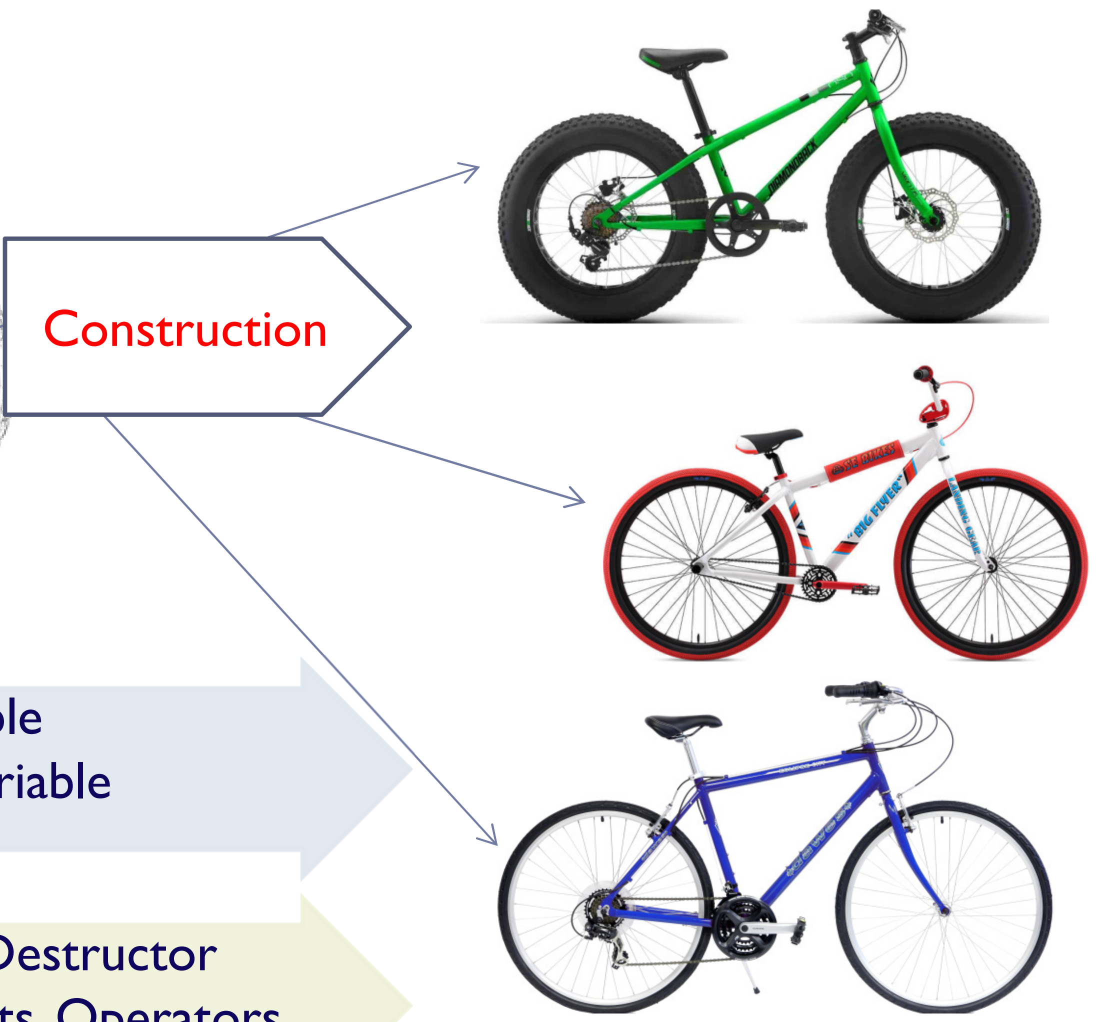
- Course Objectives and Outcomes.
  - Outline Pdf file.

Classes and Objects

- A class is a template that define the forms of an object



Blueprint/Logical State of a Bike



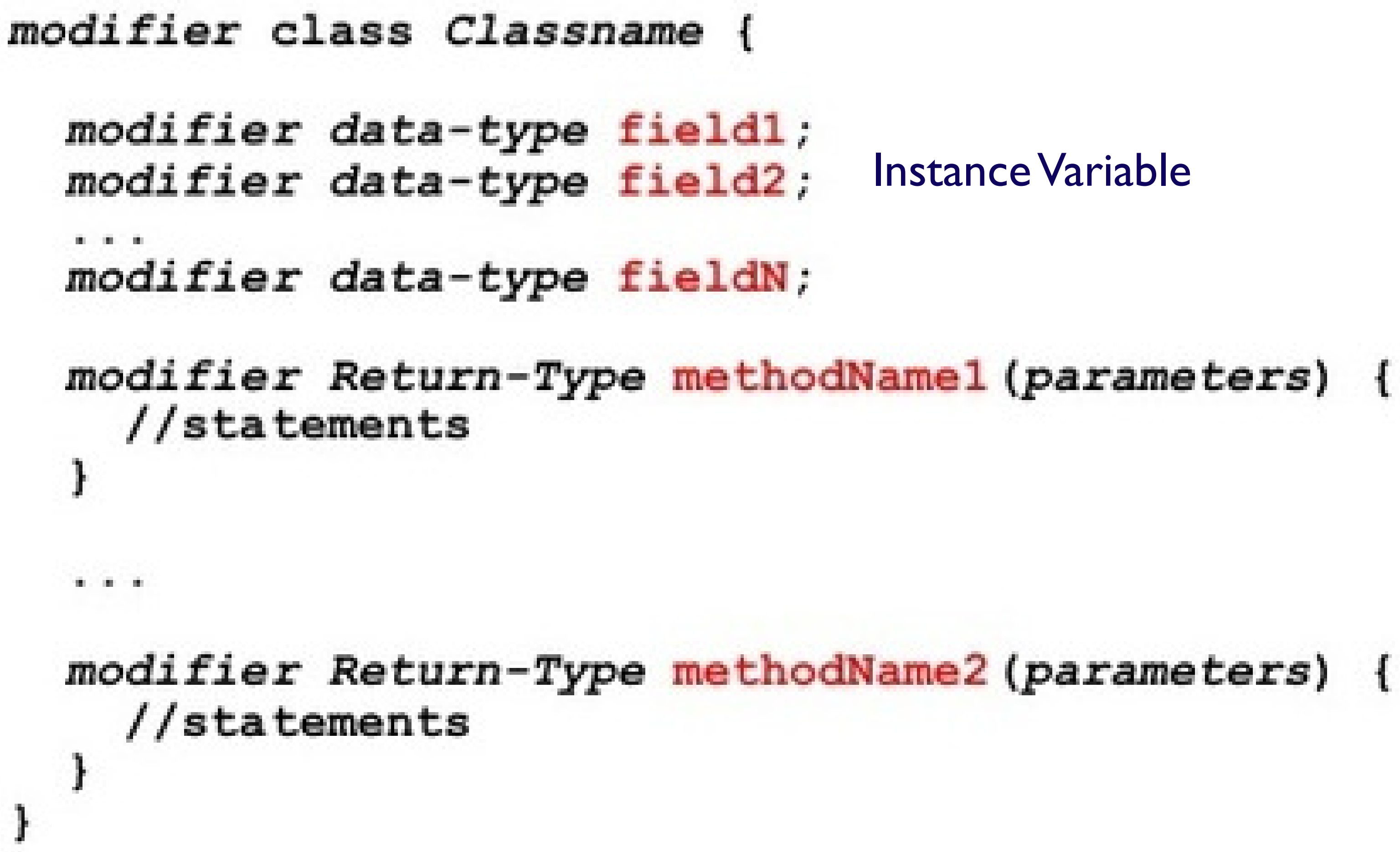
Physical State of Bikes

Field Members
- Instance Variable
- Static/Class Variable

Function Members
- Constructor, Destructor
- Indexers, Events, Operators and Properties

A Simple Class

- Class is created by use of keyword class.



```
modifier class Classname {

   modifier data-type field1;
   modifier data-type field2;
   ...
   modifier data-type fieldN;

   modifier Return-Type methodName1(parameters) {
      //statements
   }

   ...

   modifier Return-Type methodName2(parameters) {
      //statements
   }
}
```

Instance Variable

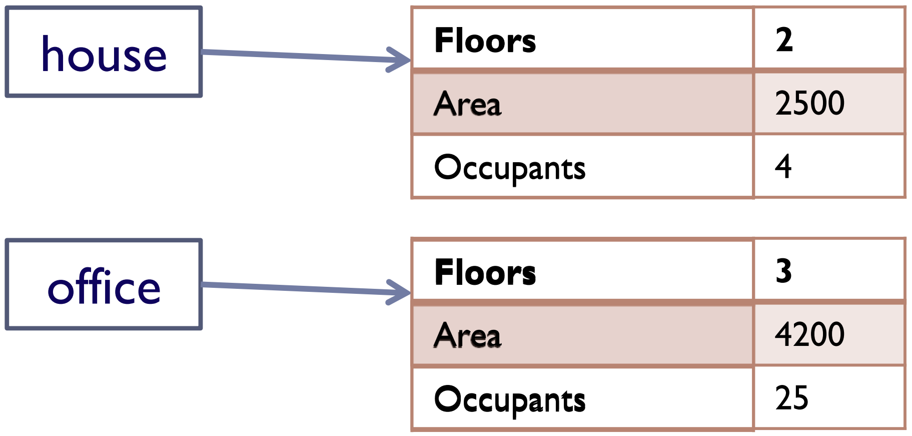
Non-static variable cannot be referenced without creating class instance

```
public class VariableExample
{
   int myVariable;
   static int data = 30;
   public static void main(String args[])
   {
        VariableExample obj = new VariableExample();
        System.out.println("Value of instance variable: "+obj.myVariable);
        System.out.println("Value of static variable: "+VariableExample.data);
   }
}
```

JAVA Program
Same works for C# also

Reference Variable and Assignment

```
using System;

class Building{
            public int Floors;
            public int Area;
            public int Occupants;

            public void areaPerPerson(){
                        Console.WriteLine(" "+ Area/Occupants + " area per person");
            }
}
public class Demo{
    static void Main(){
            Building house  = new Building();
            Building office = new Building();
            house.Occupants=4;   house.Area=2500;  house.Floors=2;
            office.Occupants=25;  office.Area=4200;   office.Floors=3;
            house.areaPerPerson();
            office.areaPerPerson();
     }
}
```



Floors	2
Area	2500
Occupants	4

Floors	3
Area	4200
Occupants	25

```
Building house1= new Building();
Building house2= house1
```

Constructor

- Constructors
  - are special methods called when a class is instantiated.
  - will not return anything.
  - name is same as class name.
  - By default C# will create default constructor internally.
  - with no arguments and no body is called default constructor.
  - with arguments is called parameterized constructor.
  - by default public.
  - We can create private constructors.
  - Constructor allocates memory for all instance variables of its class.

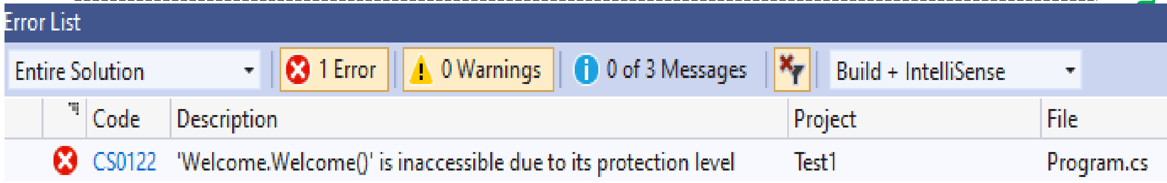
Private Constructor

I

```
using System;
namespace ConstructorSample
{
    public class Welcome
    {
        private Welcome()   // // Default private constructor
        {
          Console.WriteLine(“Default Private Constructor...");
        }

        static void Main(string[] args)
        {
            Welcome obj = new Welcome();
            Console.Read();
        }
    }
}
```

II)

```
public class demo {
        static void Main(string[] args)
        {
            Welcome obj = new Welcome();
            Console.Read();
        }
    }
```



Error List

Entire Solution | 1 Error | 0 Warnings | 0 of 3 Messages | Build + IntelliSense

Code	Description	Project	File
CS0122	'Welcome.Welcome()' is inaccessible due to its protection level	Test1	Program.cs

Parameterized Constructor

```
using System;
 public class MyClass
 {
     public int x;

     public MyClass (int i)
     {
         x=i;
     }
 }
  public class Demo{
     static void Main( )
     {
        MyClass t1=new MyClass (10);
        MyClass t2=new MyClass (88);
        Console.WriteLine(t1.x + " " + t2.x);
     }
 }
```

Garbage Collection and Destructors

- Recovery of free memory from unused object
  - C++ delete operator is used to free allocated memory
  - C# and Java: Garbage Collection- automatically
    - Can’t know or make assumptions about the timing of garbage collection
    - Non-deterministic

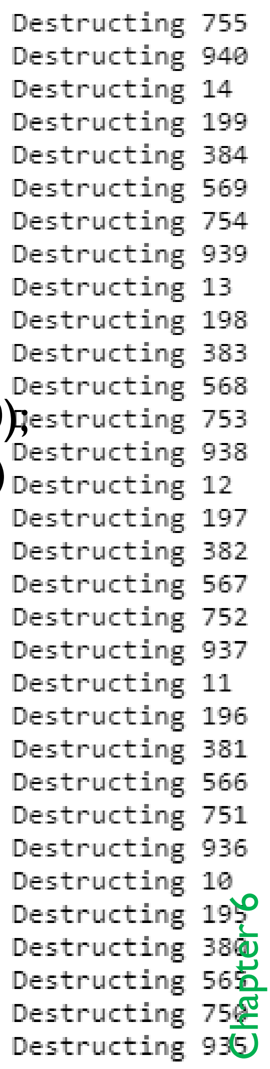
- Destructor
  - Another method to clean object allocated memory.
  - It ensures that a system resource owned by an objet is released.
  - Is declared like constructor except proceed with ~(tilde) -tilda
  - No return type and takes no arguments.

Destructor Example

```
using System;

  public class Destructor
  {
     public int x;

     public Destructor (int i)
     {
        x=i;
     }
     ~Destructor(){
        Console.WriteLine("Destructing " + x);
     }

     public void Generator(int i)
     {
        Destructor o = new Destructor(i);
     }
  }
```

```
public class Demo{
      static void Main( )
      {
         int count;
         Destructor ob = new Destructor(0);
         for(count=1; count<1000; count++)
             ob.Generator(count);

          Console.WriteLine("Done");
      }
   }
```

- No serialization
- Non deterministic



```
Destructing 755
Destructing 940
Destructing 14
Destructing 199
Destructing 384
Destructing 569
Destructing 754
Destructing 939
Destructing 13
Destructing 198
Destructing 383
Destructing 568
Destructing 753
Destructing 938
Destructing 12
Destructing 197
Destructing 382
Destructing 567
Destructing 752
Destructing 937
Destructing 11
Destructing 196
Destructing 381
Destructing 566
Destructing 751
Destructing 936
Destructing 10
Destructing 195
Destructing 380
Destructing 565
Destructing 750
Destructing 935
```

this Keyword

```
using System;
class Rect{
     public int Width;
     public int Height;
    //public Rect(){ this(3, 2); }
    public Rect(int Width, int Height){
                this.Width=Width;
                this.Height=Height;
     }
     public int Area(){
                return this.Width * this.Height;
     }
}                  class UseRect{
                        static void Main(){
                              Rect r1= new Rect(4,5);
                              Rect r2= new Rect(7, 9);
                              Console.WriteLine("Area of r1:" +r1.Area());
                              Console.WriteLine("Area of r1:" +r2.Area());
                        }
                   }
```