.NET Framework 4.6 and C# 7.0

Lesson 02 : Introduction to C# 7.0

Capgemini

Lesson Objectives

- In this lesson, you will learn:
  - The features of C# language
  - Writing simple program in C#
  - Creating DLL in C# and using it

Briefly explain how C# as a language has evolved

2.1: Introduction to C#

Concept of C#

- For the past two decades, C and C++ have been the most widely used languages for developing commercial and business software.
- Due to the complexity and long cycle times associated with these languages, many C and C++ programmers have been searching for a language offering better balance between power and productivity.
- For the past two decades, C and C++ have been the most widely used languages for developing commercial and business software.
- Due to the complexity and long cycle times associated with these languages, many C and C++ programmers have been searching for a language offering better balance between power and productivity.

Introduction to C#:

C# is a strongly typed object-oriented language whose code visually resembles C++ (and Java). This decision by the C# language designers allows C++ developers to easily leverage their knowledge to quickly become productive in C#.

C# syntax differs from C++ in some ways, but most of the differences between these languages are semantic and behavioral, stemming from differences in the runtime environments in which they execute.

C# source code compiles into managed code. Managed code, as you may already know, is an intermediate language (IL) because it is halfway between the high-level language (C#) and the lowest-level language (assembly/machine code).

At run time, the Common Language Runtime (CLR) compiles the code on the fly by using Just In Time (JIT) compiling.

Concept of C#

- C# language was introduced by Microsoft in the middle of 2000.
- C# was developed by Anders Hejlsberg.
- It is an elegant and type-safe object-oriented language which enables the developers to build the secure and robust applications which runs on .NET Framework.

C# Evolution

Year	C#	Visual Studio	Features
2002	1.0	Visual Studio 2002	Classes, Structs, Interfaces, Events, Properties, Delegates, Operators and Expressions, Statements, Attributes
2003	1.2	Visual Studio 2003	Few small enhancement
2005	2.0	Visual Studio 2005	Generics, Partial Types, Anonymous Methods, Nullable Value Types, Iterators, Covariance and Contravariance
2007	3.0	Visual Studio 2008	Auto-implemented properties, Anonymous Types, Query Expressions, Lambda Expressions, Expression Trees, Extension Methods, Implicitly Typed Local Variables, Partial Methods, Object and Collection Initializers
2010	4.0	Visual Studio 2010	Dynamic Binding, Named/Optional Arguments, Generic Covariance and Contravariance, Embedded Interop Types

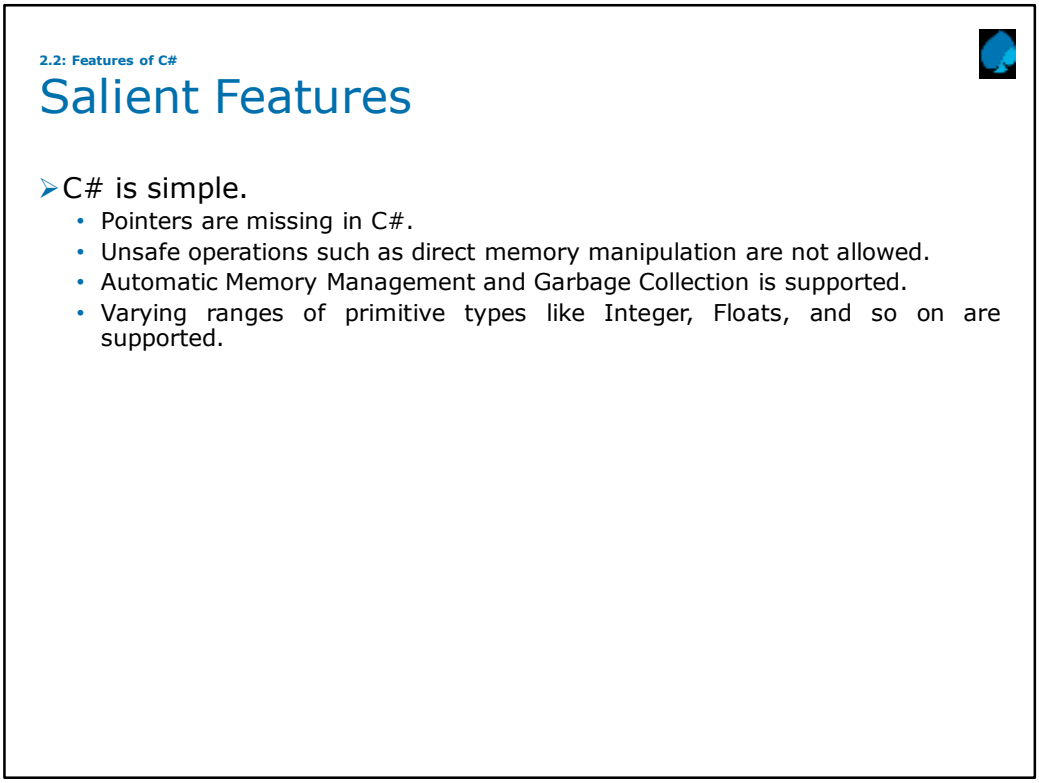
C# Evolution (Cont…)

Year	C#	Visual Studio	Features
2012	5.0	Visual Studio 2012	Asynchronous Members, Caller Info Attributes
2015	6.0	Visual Studio 2015	Static Imports, Exception Filters, Auto-property Initializers, Expression Bodied Members, Null Propagator, String Interpolation, nameof Operator, Index Initializers
2017	7.X	Visual Studio 2017	Out Variables, Tuple and Deconstruction, Pattern Matching, Local Functions, Expanded Expression Bodied Members, Ref locals and returns, Discards, Binary Literals and Digit Separators, Throw expressions, async Main Method, default literal expressions, Inferred tuple element names

C# Evolution (Cont…)

Year	C#	Visual Studio	Features
2019	8.0	Visual Studio 2019	Readonly Members, Default Interface Methods, Pattern Matching Enhancements, Using declarations, Static local functions, Disposable ref structs, Nullable reference types, Asynchronous Streams, Indices and Ranges, Null-coalescing assignment, Unmanaged Constructed Types, Stackalloc in nested expressions

Briefly talk about managed code mechanism of C# and why pointers are not allowed



Features of C#:

C# is simple:

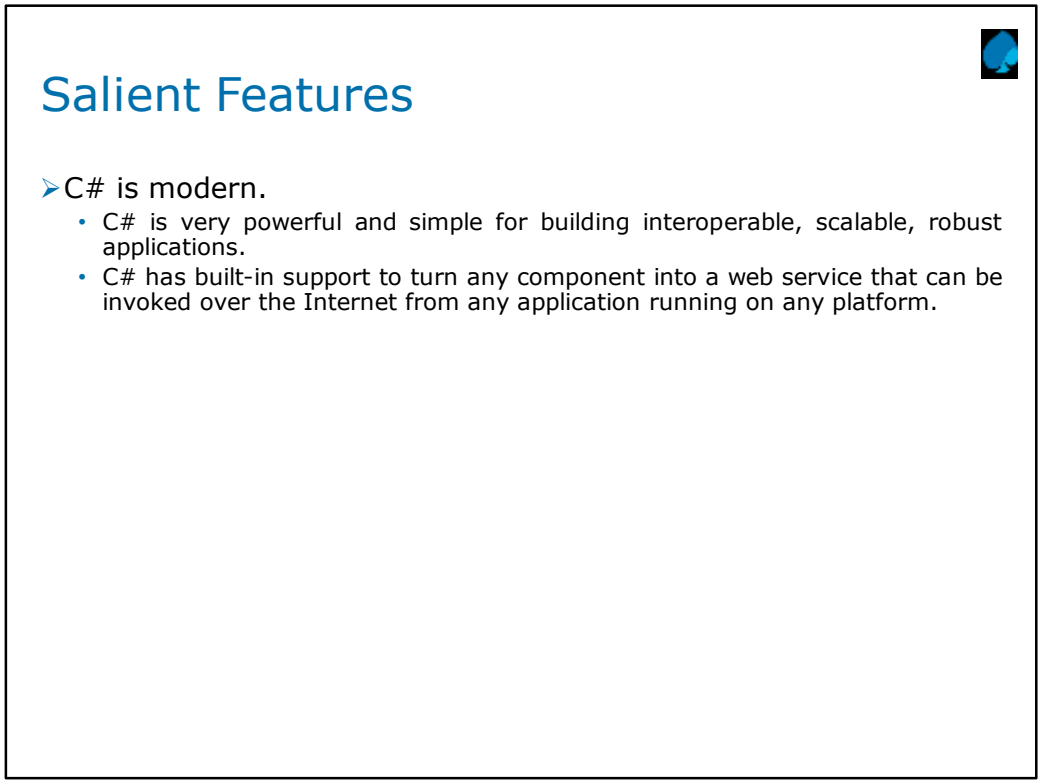
Pointers are missing in C#.

Unsafe operations such as direct memory manipulation are not allowed.

Since it is on .NET, C# inherits the features of automatic memory management and garbage collection.

Varying ranges of the primitive types like Integer, Floats, and so on are supported.

One language for variety of applications.



Features of C#:

C# is modern:

C# has been based according to the current trend and is very powerful and simple for building interoperable, scalable, robust applications.

C# includes built-in support to turn any component into a web service that can be invoked over the Internet from any application running on any platform.

Salient Features

- C# is object oriented.
  - C# supports Data Encapsulation, inheritance, polymorphism, interfaces.
  - C# introduces structures (structs) which enable the primitive types to become objects.

```
int i=1; string a=i.Tostring(); //conversion (or) Boxing
```

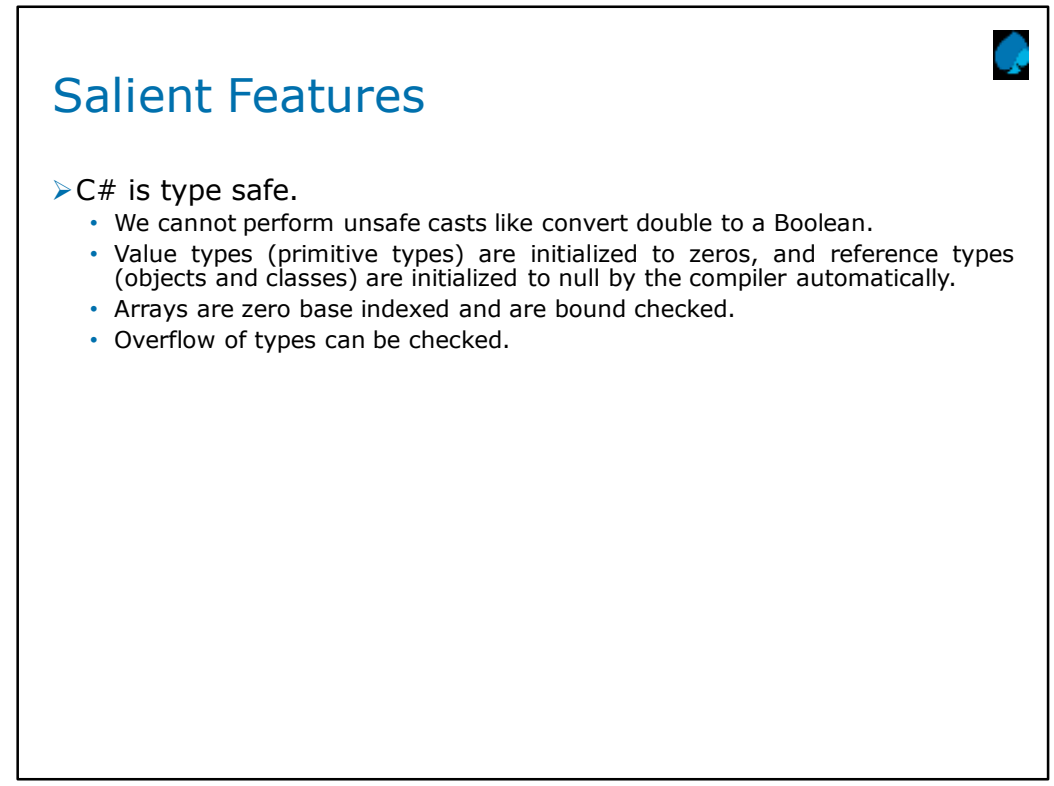
Features of C#:

C# is object oriented.

C# supports Data Encapsulation, inheritance, polymorphism, interfaces.

int, float, double are not objects in Java. However, C# has introduced structures (structs) which enable the primitive types to become objects. (An example is shown in the above slide.)

Briefly talk about the Type System of .NET



Features of C#:

C# is type safe.

In C#, we cannot perform unsafe casts like convert double to a Boolean.

Value types (primitive types) are initialized to zeros and reference types (objects and Classes) are initialized to null by the compiler automatically.

Arrays are zero base indexed and are bound checked.

Overflow of types can be checked.

Salient Features

- C# shows interoperability.
  - It includes native support for the COM and windows-based applications.
  - C# allows the users to use pointers as unsafe code blocks to manipulate your old code.
  - Components from VB NET and other managed code languages can directly be used in C#.

Features of C#:

C# shows interoperability.

C# includes native support for the COM and windows based applications.

C# allows restricted use of native pointers.

C# allows the users to use pointers as unsafe code blocks to manipulate your old code.

Components from VB NET and other managed code languages can be directly used in C#.

Briefly explain about Assemblies in C#

Salient Features

- C# is scalable and updateable.
  - .NET has introduced assemblies, which are self-describing by means of their manifest. Manifest establishes the assembly identity, version, culture and digital signature etc. Assemblies need not to be registered anywhere.
  - C# does not require registering of dynamic linking library.
  - C# supports versioning in the language.

Features of C#:

C# is scalable and updateable.

.NET has introduced assemblies, which are self-describing by means of their manifest. Manifest establishes the assembly identity, version, culture and digital signature etc. Assemblies need not to be registered anywhere.

To scale our application, we delete the old files and update them with new ones. C# does not require registering of dynamic linking library.

Updating software components is an error prone task. Revisions made to the code can effect the existing program. C# support versioning in the language.

This program can be demonstrated using Notepad so that participants understand the process of compilation

2.3: Hello World Program

Illustration

- Let us see an example in C# using Hello World program:

```
// A "Hello World!" program in C#  Hello.cs
public class Hello
{
     public static void Main()
     {
          System.Console.WriteLine("Hello World!");
      }
}
```

Hello World Program:

To compile the program from the command line, use the following syntax:

```
csc Hello.cs
```

If no compilation errors, then a **Hello.exe** file will be created. The exe generated is referred as Assembly. It contains **IL** code and **Manifest** file which contains meta data.

To run the program, enter the following command:

Hello

Demo

- Demo with Hello World Program using Notepad

Explain how the program can be compiled to .dll

Illustration

- Let us see an example on creating a DLL:

```
// File: Add.cs
namespace UtilityMethods
{
    public class AddClass
    {
            public static long Add(long i, long j)
            { return (i + j); }
    }
 }
```

Creating a DLL:

In the example in the above slide, two classes are created under a same namespace. We will compile these as DLLs.

Illustration

Explain how the program can be compiled to .dll

```
// File: Mult.cs
namespace UtilityMethods
{
    public class MultiplyClass
    {
            public static long Multiply(long x, long y)
            { return (x * y); }
    }
}
```

Use ILDASM tool and show the generated assembly using this tool.

Illustration

- Creating a DLL:
  - To build the file MathLibrary.DLL, compile the two files Add.cs and Mult.cs using the following command line:

```
csc /target:library /out:MathLibrary.DLL Add.cs Mult.cs
```

  - The **/target:library** compiler option tells the compiler to output a DLL instead of an EXE file.
  - The **/out** compiler option followed by a file name is used to specify the DLL file name. Otherwise, the compiler uses the first file **(Add.cs)as** the name of the DLL.

Explain how we can use the created .dll file into a .exe application by specifying a reference of .dll while compiling

2.5: Using a DLL

Illustration

Let us see an example on using a DLL:

```
class TestCode
{
static void Main(string[] args)
      {
Console.WriteLine("Calling
methods from MathLibrary.DLL:");
if (args.Length != 2)
      {
Console.WriteLine("Usage:
TestCode <num1> <num2>");

return;
      }
```

```
long num1 = long.Parse(args[0]);
long num2 = long.Parse(args[1]);
long sum = AddClass.Add(num1,
num2);
long product =
MultiplyClass.Multiply(num1, num2);
Console.WriteLine("{0} + {1} = {2}",
num1, num2, sum);
Console.WriteLine("{0} * {1} = {2}",
num1, num2, product);
      }
}
```

Illustration

- To build the file TestCode.exe:

```
csc /reference:MathLibrary.DLL TestCode.cs
```

Using a DLL:

The **/reference** compiler option specifies the DLL file or files that this program uses.

To run the program, enter the name of the EXE file, followed by two numbers, as follows:

```
TestCode 1234 5678
```

Demo

- Creating and Using a DLL

Summary

- In this lesson, you have learnt:
  - Different features of C#
  - The method to write Programs in Notepad and compile it through Visual Studio Command Prompt
  - The method to create and use DLL in C#

Review Questions

- Question 1: What are the different features C# offers?
- Question 2: How does the compilation take place for C# Program?
- Question 3: Can you create DLL in C#? How?

Answers for the Review Questions:

Answer 1:
Pointers are missing in C#
Automatic Memory Management
Varying ranges of data types
Object Oriented
Type Safe

Answer 2:
csc FileName.cs

Answer 3:
Yes.
Use the following command to create the DLL using C#:
csc /target:library /out:FileName.DLL FirstClassFileName.cs SecondClassFileName.cs