.NET Framework 4.7 and C# 8.0

Lesson 02 : Introduction to C# 8.0

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

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.

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

Salient Features

- C# is simple.
  - Pointers are missing in C#.
  - Unsafe operations such as direct memory manipulation are not allowed.
  - Automatic Memory Management and Garbage Collection is supported.
  - Varying ranges of primitive types like Integer, Floats, and so on are supported.

Salient Features

- C# is modern.
  - C# is very powerful and simple for building interoperable, scalable, robust applications.
  - C# has 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
```

Salient Features

- C# is type safe.
  - 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#.

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.

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!");
      }
}
```

Demo

- Demo with Hello World Program using Notepad

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); }
    }
 }
```

Illustration

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

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.

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
```

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?