NB Healthcare Technologies Pvt Ltd

Day 17 Morning Assignment (15 – Feb- 2022)

By
Vamsi Krishna Mandapati

1. Research and write what is assembly in C#

Assembly: An assembly can be a .exe file (or) dll file (or) an independent smallest unit of code is known as assembly.

An assembly is **a file that is automatically generated by the compiler upon successful compilation of every . NET application**. It can be either a Dynamic Link Library or an executable file. It is generated only once for an application and upon each subsequent compilation the assembly gets updated.

Assemblies form the fundamental units of deployment, version control, reuse, activation scoping, and security permissions for .NET-based applications. An assembly is a collection of types and resources that are built to work together and form a logical unit of functionality. 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.

The .NET assembly is the standard for components developed with the Microsoft.NET. Dot NET assemblies may or may not be executable, i.e., they might exist as the executable (.exe) file or dynamic link library (DLL) file. All the .NET assemblies contain the definition of types, versioning information for the type, meta-data, and manifest. The designers of .NET have worked a lot on the component (assembly) resolution.

An assembly can be a single file or it may consist of the multiple files. In the case of multi-file, there is one master module containing the manifest while other assemblies exist as non-manifest modules. A module in .NET is a subpart of a multi-file .NET assembly. Assembly is one of the most interesting and extremely useful areas of .NET architecture along with reflections and attributes.

.NET supports three kinds of assemblies:

1. private
2. shared

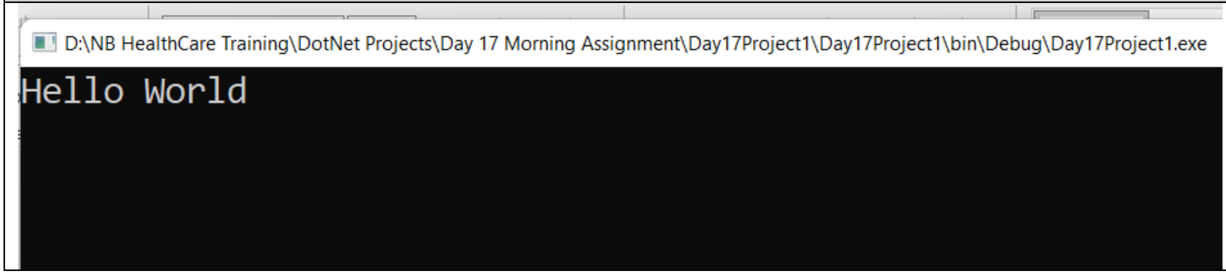
3. satellite

Example For Assembly/Independent Smallest unit of code snippet

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Day17Project1
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World");

            Console.ReadLine();
        }
    }
}
```

Output:



```
D:\NB HealthCare Training\DotNet Projects\Day 17 Morning Assignment\Day17Project1\Day17Project1\bin\Debug\Day17Project1.exe
Hello World
```

2. In a tabular format write the access modifiers and explain
(as I did in the class, create two assemblies with 3 classes in first assembly, 2 classes in other assembly)

Access Specifiers

	With in Assembly			Other Assembly	
	With in Class	Derived Class	Other Class	Derived Class	Other Class
Public	Yes	Yes	Yes	Yes	Yes
Private	Yes	No	No	No	No
Protected	Yes	Yes	No	Yes	No
Internal	Yes	Yes	Yes	No	No
Protected Internal	Yes	Yes	Yes	Yes	No
Public : All Classes					

Private: Only With in the Class

Protected: Base and Derived classes of Same assembly and other assemblies also

Internal : With in the assembly

VamsiLibrary

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace VamsiKrishnaLibrary
{
    public class BaseClass
    {
        public int a;
        private int b;
        protected int c;
        internal int d;
        protected internal int e;

        public void BaseClassMethod()
        {
            a = 10;
            b = 20;
            c = 30;
            d = 40;
            e = 50;
        }



    }
    public class DerivedClass : BaseClass
    {
        public void DerivedClassMethod()
        {
            a = 10;
            //b = 20;                 //Private variable can't access
here
            c = 30;
            d = 40;
            e = 50;
        }
    }
    public class OtherClass
    {
        public void OtherClassMethod()
        {
            BaseClass b = new BaseClass();
            b.a = 10;
            //b.b = 20;     Private variable can't access here
            //b.c = 30;     protected variable can't acess here
            b.d = 40;
            b.e = 50;
        }
    }
}
```

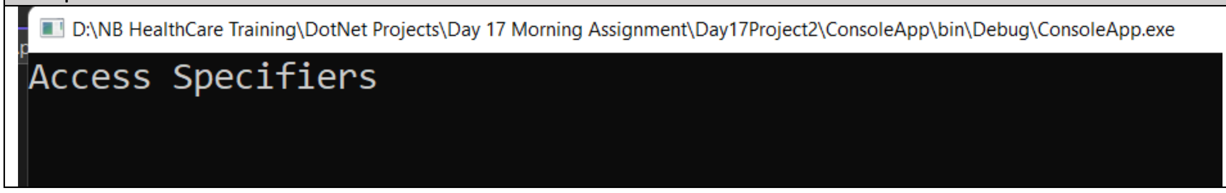
PublicLibrary:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using VamsiKrishnaLibrary;

namespace PublicLibrary
{
    public class PublicLibraryDerivedClass : BaseClass
    {
        public void PublicLibraryDerivedClassMethod()
        {
            a = 10;
            //b = 20;           private variable can't access here
            c = 30;
            //d = 40;           internal variable can't access here
            e = 50;
        }
    }
    public class PublicLibraryOtherClass
    {
        public void PublicLibraryOtherClassMethod()
        {
            BaseClass b = new BaseClass();
            b.a = 10;
            //b.b = 20;   private variable can't access here
            //b.c = 30;     Protected variable can't access here
            //b.d = 40;     internal variable can't access here
            //b.e = 50;        protected internal can't access here
        }
    }

}
```

Console App:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using VamsiKrishnaLibrary;
using PublicLibrary;

namespace ConsoleApp
{
    internal class Program
    {
        static void Main(string[] args)
        {

            Console.WriteLine("Access Specifiers");

            Console.ReadLine();

        }
    }
}
```

Output:



```
D:\NB HealthCare Training\DotNet Projects\Day 17 Morning Assignment\Day17Project2\ConsoleApp\bin\Debug\ConsoleApp.exe
Access Specifiers
```