|  |
| --- |
| Day 17 Assignment  by Ramakrishna |

|  |
| --- |
| 1. Research and write what is assembly in C# |
| An Assembly is a basic building block of .Net Framework applications. It is basically a compiled code that can be executed by the CLR. An assembly is a collection of types and resources that are built to work together and form a logical unit of functionality. An Assembly can be a DLL or exe depending upon the project that we choose.    Assemblies are basically the following two types:   1. Private Assembly 2. Shared Assembly   **1. Private Assembly**    It is an assembly that is being used by a single application only. Suppose we have a project in which we refer to a DLL so when we build that project that DLL will be copied to the bin folder of our project. That DLL becomes a private assembly within our project. Generally, the DLLs that are meant for a specific project are private assemblies.    **2. Shared Assembly**    Assemblies that can be used in more than one project are known to be a shared assembly. Shared assemblies are generally installed in the GAC. Assemblies that are installed in the GAC are made available to all the .Net applications on that machine.  Assemblies have the following properties:   * Assemblies are implemented as .exe or .dll files. * For libraries that target .NET Framework, you can share assemblies between applications by putting them in the [global assembly cache (GAC)](https://docs.microsoft.com/en-us/dotnet/framework/app-domains/gac). You must strong-name assemblies before you can include them in the GAC. For more information, see [Strong-named assemblies](https://docs.microsoft.com/en-us/dotnet/standard/assembly/strong-named). * Assemblies are only loaded into memory if they are required. If they aren't used, they aren't loaded. This means that assemblies can be an efficient way to manage resources in larger projects. * You can programmatically obtain information about an assembly by using reflection. For more information, see [Reflection (C#)](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/reflection) or [Reflection (Visual Basic)](https://docs.microsoft.com/en-us/dotnet/visual-basic/programming-guide/concepts/reflection). * You can load an assembly just to inspect it by using the [MetadataLoadContext](https://docs.microsoft.com/en-us/dotnet/api/system.reflection.metadataloadcontext) class on .NET and .NET Framework. [MetadataLoadContext](https://docs.microsoft.com/en-us/dotnet/api/system.reflection.metadataloadcontext) replaces the [Assembly.ReflectionOnlyLoad](https://docs.microsoft.com/en-us/dotnet/api/system.reflection.assembly.reflectiononlyload) methods. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2. In a tabular format write the access modifiers and explain | | | | | |
|  | With Assembly | | | Other Assembly | |
| With  Class | Derived  Class | Other  Class | Derived  Class | Derived  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 |
| Internal protected | yes | yes | yes | yes | No |

|  |
| --- |
| myBaseClass |
| MyDerviedClass: |
| MyOtherClass |
| MyPublicDerviedClass: |
| MyPublicOtherClass: |