.NET Assemblies

Private Assemblies

Assemblies are files that contain compiled code targeted at the .NET Framework. These are basically physical packages meant for distributing code. The .NET classes are actually contained in several assemblies (Harwani, 2015, p. 621).

An assembly has two (2) sections:

- **Metadata** – It includes information about the data types of the program that are being used.
- **Manifest** – It holds the information of the assembly, which consists of the name, version number, and the type of mapping information.

An assembly also has two (2) file extensions:

- **.exe** – for standalone applications
- **.dll** – for reusable components.

.NET Frameworks' core assemblies location can be found in `C:\Windows\assembly`.

Private Assembly

- Simplest type of assembly
- Can only be used within a software package that is intended to be used.
- Two (2) private assemblies with the same class name are not a problem since the application can only see the classes that are mentioned in its private assemblies
- Does not require registry entries

Shared Assemblies

These are libraries that other applications can commonly use. Security precautions are necessary when using a shared assembly since any other software can access a shared assembly.

Name Collision

- It is a common problem in shared assembly wherein other classes or variables have the same name that matches with the other shared assembly.
- A different form of the same assembly might overwrite the shared assembly, and the new version might be incompatible with the existing code.
- Shared assemblies are giving a strong, unique name based on the private key cryptography to avoid name collision.

Global Assembly Cache (GAC)

- It enables several applications to share shared assembly.
- It is required to add the assembly to the special directory.
- It is a centralized storage location for .NET Assemblies.

Creating a Shared Assembly

These are the steps to creating a shared assembly:

1. Create a project containing a class file. The class file contains the methods and properties that you want other applications to access.
2. Generate a strong name for the project. The strong name is saved in a strong key filename (e.g., **ShareAssemblyMessageKeyFile**). The strong key file is created named **ShareAssemblyMessageKeyFile.snk** and located at the solutions explorer.
3. Specify the key filename in the project by indicating its strong key filename in the `AssemblyInfo.cs` file.
4. Compile the project to generate an assembly. The assembly is generated with the extension `.dll`.

REFERENCES:
Deitel, P. & Deitel, H. (2015). *Visual C# 2012 how to program* (5th ed.). USA: Pearson Education, Inc.
Gaddis, T. (2016). *Starting out with Visual C#* (4th ed.). USA: Pearson Education, Inc.
Harwani, B. (2015). *Learning object-oriented programming in C# 5.0.* USA: Cengage Learning PTR.
Miles, R. (2016). *Begin to code with C#.* Redmond Washington: Microsoft Press.
Doyle, B. (2015). *C# programming: from problem analysis to program design* (5th ed.). Boston, MA: Cengage Learning.