Access Modifiers in C#

Namespaces

1. A *Namespace* in C# is similar to a *package* in Java. It is a way to group classes and control access.

   - In Java the directory structure should match the package structure. No such restriction with C# namespaces.
   - In C# you can have multiple namespaces in one file. In Java you can only have one package in a file.
   - In Java you can import individual classes, whereas in C# the *using* directive imports a whole namespace.
   - In C# the *using* directive can also specify aliases, which Java doesn't support.

   Sources:
   http://stackoverflow.com/questions/9249357/difference-between-namespace-in-c-sharp-and-package-in-java
   http://stackoverflow.com/questions/20872223/is-namespace-in-c-sharp-is-something-similar-to-package-in-java

Assemblies

2. An a*ssembly* in .NET is similar to a *JAR* file in Java in the sense that both provide a way of packaging a distributable application; however, there are differences we will not discuss. What is an assembly?

   - An assembly is code compiled into MSIL (similar to Java bytecode) into either a DLL or EXE format. An assembly is used by the CLR (similar to JVM).
   - A .NET application consists of one or more assemblies.
   - Assemblies include both executable application files that you can run directly from Windows without the need for any other programs (these have a .exe file extension), and libraries (which have a .dll extension) for use by other applications.

   Sources:
   http://stackoverflow.com/questions/3721411/is-a-java-jar-file-similar-to-an-net-assembly
   http://stackoverflow.com/questions/2972732/what-is-net-assembly

Web Sites and Web Applications

This is out dated in VS 2019

3. "In Visual Studio you can create web application projects or web site projects. You create or open a web application project by choosing New Project or Open Project in the Visual Studio File menu. You create or open a web site project by choosing New Web Site or Open Web Site in the File menu." You can use either to create a "web site." We will discuss one difference next, but a thorough treatment of the differences is found here:

   http://msdn.microsoft.com/en-us/library/dd547590(v=vs.110).aspx

4. On difference is that with *web application* projects, the code is precompiled into a single (typically) *dll* and stored in a *bin* folder in the root of your web site. On the other hand, a *web site* is compiled on the fly (or retrieved from cache), as users request a page, into multiple *dll*'s, typically one per page and stored in a temporary location, typically:

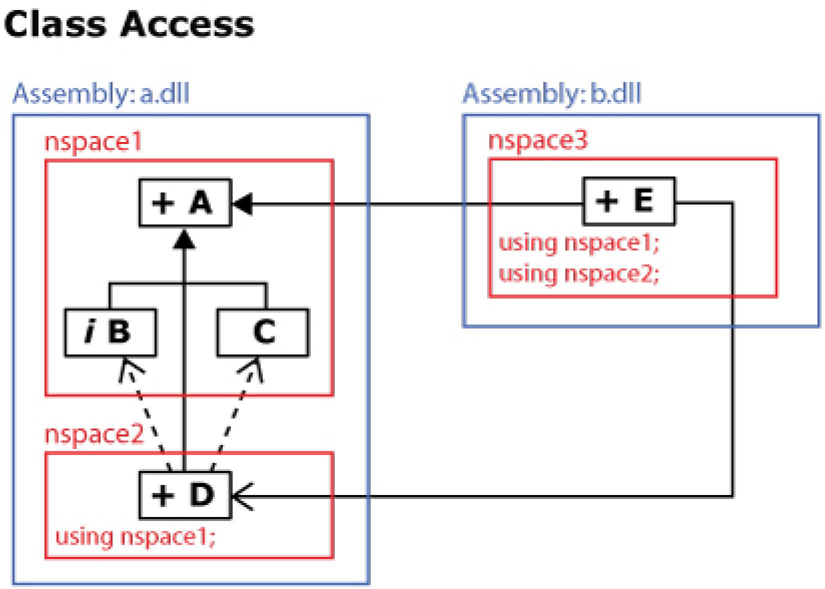
   C:\Windows\Microsoft.NET\Framework\v4.0.30319\Temporary ASP.NET Files

5. In this class we are creating/editing *web site projects*.

Class Access Modifiers

6. In .NET, classes can be declared *public*, *internal*, or left blank which defaults to internal. Thus, there are only two cases. *Internal* is similar to package level access in Java, except that *internal* classes can be referenced by any other class in the assembly, not just the namespace.

7. In the example below, class D can access class B (or C) even though it is in a different namespace. However, class E cannot access class B or C because they are in different assemblies.



Method Access Modifiers

8. In .NET methods can have one of these access modifiers:

Method Access Modifier	Meaning
public	Can access anywhere
internal	Can access anywhere in the same assembly
protected internal	Can access from anywhere in the assembly or from a class derived from that class
protected	Can access from same class or class derived from that class
private	Can access only from that class

http://msdn.microsoft.com/en-us/library/ms173121.aspx

9. Similarities between Java and C#

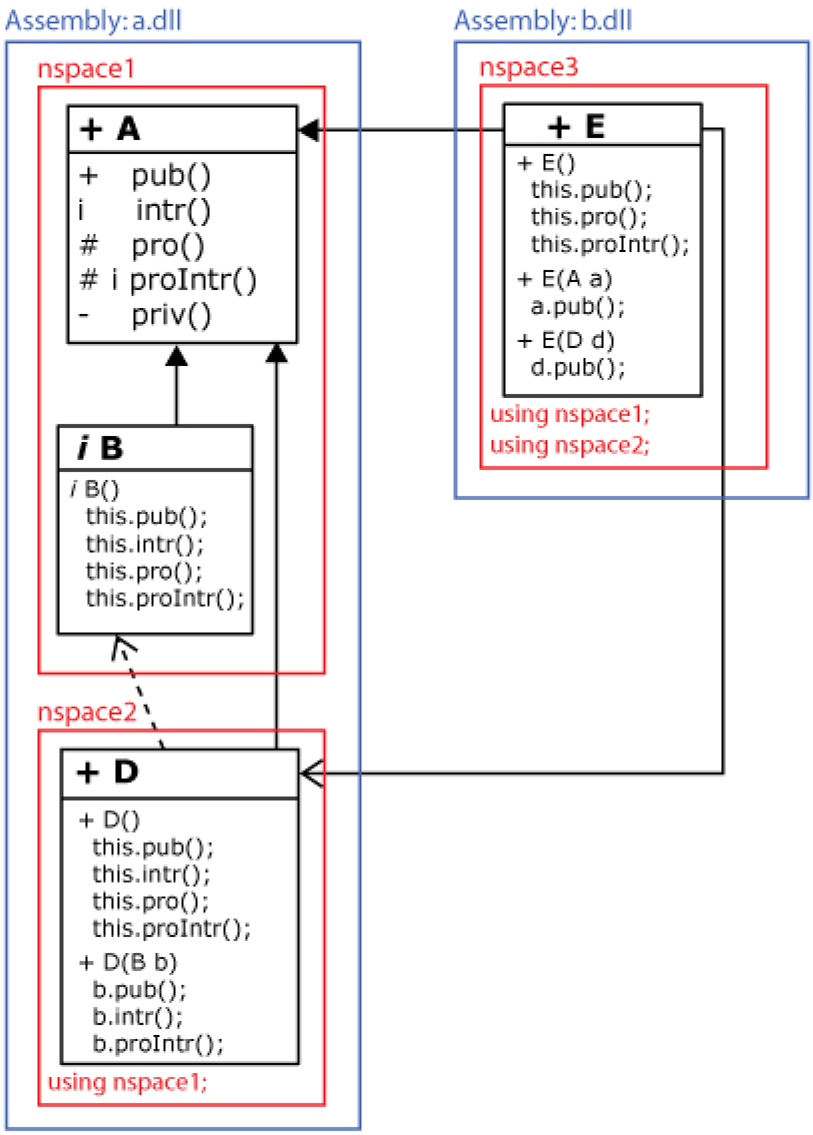
- C#'s internal is equivalent to Java's default scope (package level).
- C#'s internal protected is equivalent to Java's protected.

http://stackoverflow.com/questions/1295966/access-modifier-best-practice-in-c-sharp-vs-java

10. In the figure below, we see that:

- D which is a subclass of A, can access these methods in A: pub, intr, pro, proIntr
- D which accepts a B object in its constructor, can access these methods in B: pub, intr, proIntr
- E's default constructor, which is a subclass of A, can access these methods is A: pub, pro, proIntr
- E's constructor that takes an A object, can access these methods in that A instance: pub

Method Access


Assembly: a.dll
nspace1
+ A
+ pub()
i intr()
# pro()
# i proIntr()
- priv()
i B
i B()
this.pub();
this.intr();
this.pro();
this.proIntr();
nspace2
+ D
+ D()
this.pub();
this.intr();
this.pro();
this.proIntr();
+ D(B b)
b.pub();
b.intr();
b.proIntr();
using nspace1;
Assembly: b.dll
nspace3
+ E
+ E()
this.pub();
this.pro();
this.proIntr();
+ E(A a)
a.pub();
+ E(D d)
d.pub();
using nspace1;
using nspace2;