VALID COMBINATION

Private Protected:

- Access is limited to the containing class or types derived from the containing class within the current assembly
- The private protected keyword combination serves as a modifier for accessing members. This grants access to a private protected member solely to types derived from the class it belongs to, but its reach is limited to within the assembly it is contained in.
- A private protected member of a base class is accessible from derived types in its containing assembly only if the static type of the variable is the derived class type.

Sample:

```
public class BaseClass
{
   private protected int myValue = 0;
}

public class DerivedClass1 : BaseClass
{
   void Access()
   {
      var baseObject = new BaseClass();

      // Error CS1540, because myValue can only
be accessed by
      // classes derived from BaseClass.
      // baseObject.myValue = 5;

      // OK, accessed through the current derived
class instance
      myValue = 5;
   }
}
```

```
// Assembly2.cs
// Compile with: /reference:Assembly1.dll
class DerivedClass2 : BaseClass
{
   void Access()
   {
      // Error CS0122, because myValue can only
be
      // accessed by types in Assembly1
      // myValue = 10;
   }
}
```

This example showcases the relationship between two files, Assembly1.cs and Assembly2.cs.
In Assembly1.cs, we find a public base class named BaseClass and its derived class, DerivedClass1. BaseClass possesses a private protected member, myValue, which is accessed by DerivedClass1 in two ways. While the first attempt to access myValue through an instance of BaseClass will result in an error, using it as an inherited member in DerivedClass1 will prove to be successful. On the other hand, in Assembly2.cs, we see an attempt to access myValue as an inherited member of DerivedClass2. However, this will result in an error as myValue is only accessible by derived types

in Assembly1. To overcome this problem in accessibility, we can use an InternalsVisibleToAttribute in Assembly1.cs to name Assembly2. This will grant the derived class DerivedClass2 access to the private member, myValue. This allows for better organization and seamless communication between the two.

Protected Internal:

- Access is limited to the current assembly or types derived from the containing class.
- The protected internal keyword combination is a member access modifier. A protected internal member is accessible from the current assembly or from types that are derived from the containing class.
- A protected internal member of a base class is accessible from any type within its containing assembly. It is also accessible in a derived class located in another assembly only if the access occurs through a variable of the derived class type.

Sample:

```
// Assembly1.cs
// Compile with: /target:library
public class BaseClass
{
  protected internal int myValue = 0;
}

class TestAccess
{
   void Access()
   {
      var baseObject = new BaseClass();
      baseObject.myValue = 5;
   }
}
```

```
// Assembly2.cs
// Compile with: /reference:Assembly1.dll
class DerivedClass : BaseClass
{
   static void Main()
   {
      var baseObject = new BaseClass();
      var derivedObject = new DerivedClass();

      // Error CS1540, because myValue can only
be accessed by
      // classes derived from BaseClass.
      // baseObject.myValue = 10;

      // OK, because this class derives from
BaseClass.
      derivedObject.myValue = 10;
   }
}
```

In this example, we have two files: Assembly1.cs and Assembly2.cs. Inside Assembly1.cs, there's a public base class called BaseClass and another class named TestAccess. BaseClass has a special part called myValue, which TestAccess uses. But in Assembly2.cs, if we try to use myValue with a BaseClass, it won't work and will cause an error. However, if we use myValue with a derived class like DerivedClass, it will work just fine.

INVALID COMBINATIONS :

Internal Public:

- Cannot be combined together, since they contradict each other's function. Internal makes a member accessible within the same assembly but not from outside restricting access to the assembly in which the member is declared. While, public makes a member accessible from any part of the program, including outside the assembly where it is declared.

Invalid: private public:

- Private is the most restrictive access level, and public is the least restrictive. Combining them is not possible.

Invalid: protected private:

- Protected allows access to derived classes, and private restricts access to the declaring class.
  Combining them is contradictory.

https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/protected-internal

https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/private-protected