**Static Members and Instance Members**

you can define a member of a class as being a *static* *member* or an *instance member*. By default, each member is defined as an instance member, which means that a copy of that member is made for every instance of the class. When a member is declared as a static member, there is only one copy of the member. A static member is created when the application containing the class is loaded, and it exists for the life of the application. Therefore, you can access the member even before the class has been instantiated. But why would you do this?

One example involves the *Main* method. The Common Language Runtime (CLR) needs to have a common entry point to your application. So that the CLR doesn't have to instantiate one of your objects, the rule is to define a static method called *Main* in one of your classes. You'd also want to use static members when you have a method that, from an object-oriented perspective, belongs to a class in terms of semantics but doesn't need an actual object—for example, if you want to keep track of how many instances of a given object are created during the lifetime of an application. Because static members live across object instances, the following would work:

using System;

class InstCount

{

public InstCount()

{

**instanceCount++;**

}

**static public int instanceCount = 0;**

}

class AppClass

{

public static void Main()

{

Console.WriteLine(InstCount.instanceCount);

InstCount ic1 = new InstCount();

Console.WriteLine(InstCount.instanceCount);

InstCount ic2 = new InstCount();

Console.WriteLine(InstCount.instanceCount);

}

}

A last note on static members: a static member must have a valid value. You can specify this value when you define the member, as follows:

static public int instanceCount1 = 10;

If you do not initialize the variable, the CLR will do so upon application startup by using a default value of 0.