**The Main Method**

Every C# application must have a *Main* method defined as a method in one of its classes. In addition, this method must be defined as *public* and *static*. It doesn't matter to the C# compiler which class has the *Main* method defined, nor does the class you choose affect the order of compilation.

class Employee

{

private int employeeId;

}

class AppClass

{

**static public void Main()**

{

Employee emp = new Employee();

}

}

## Command-Line Arguments

You can access the command-line arguments to an application by declaring the *Main* method as taking a string array type as its only argument. At that point, the arguments can be processed as you would any array

using System;

class CommandLineApp

{

**public static void Main(string[] args)**

{

foreach (string arg in args)

{

Console.WriteLine("Argument: {0}", arg);

}

}

}

And here's an example of calling this application with a couple of randomly selected values:

e:>CommandLineApp 5 42

Argument: 5

Argument: 42

However, you can also define *Main* to return a value of type *int*. Although not common in GUI applications, this can be useful when you're writing console applications that are designed to be run in batch. The *return* statement terminates execution of the method and the returned value is used as an error level to the calling application or batch file to indicate user-defined success or failure. To do this, use the following prototype:

**public static int Main()**

{

// Return some value of type int

// that represents success or value.

return 0;

}

## Multiple Main Methods

The designers of C# included a mechanism by which you can define more than one class with a *Main* method. Why would you want to do that? One reason is to place test code in your classes. You can then use the */main:<* className *>* switch with the C# compiler to specify which class's *Main* method is to be used. Here's an example in which I have two classes containing *Main* methods:

using System;

class Main1

{

**public static void Main()**

{

Console.WriteLine("Main1");

}

}

class Main2

{

**public static void Main()**

{

Console.WriteLine("Main2");

}

}

To compile this application such that the *Main1.Main* method is used as the application's entry point, you'd use this switch:

csc MultipleMain.cs /main:Main1

Changing the switch to */main:Main2* would then use the *Main2.Main* method.