**Creating Static Classes and Members**  
  
In some cases, you may want to create a class purely to encapsulate some useful functionality, rather than to represent an instance of anything. For example, suppose you wanted to create a set of methods that convert imperial weights and measures to metric weights and measures, and vice versa. It would not make sense if you had to instantiate a class in order to use these methods, because you do not need to store or retrieve any instance-specific data. In fact, the concept of an instance is meaningless in this case.

In scenarios like this, you can create a static class. A static class is a class that cannot be instantiated. To create a static class, you use the static keyword. Any members within the class must also use the static keyword, as shown in the following example:

// Static Classes  
public static class Conversions  
{  
   public static double PoundsToKilos(double pounds)  
   {  
      // Convert argument from pounds to kilograms  
      double kilos = pounds \* 0.4536;  
      return kilos;  
   }  
   public static double KilosToPounds(double kilos)  
   {  
      // Convert argument from kilograms to pounds  
      double pounds = kilos \* 2.205;  
      return pounds;  
   }  
}

To call a method on a static class, you call the method on the class name itself instead of on an instance name, as shown by the following example:

//Calling Methods on a Static Class  
double weightInKilos = 80;  
double weightInPounds = Conversions.KilosToPounds(weightInKilos);

**Static Members**  
Non-static classes can include static members. This is useful when some behaviors and characteristics relate to the instance (instance members), while some behaviors and characteristics relate to the type itself (static members). Methods, fields, properties, and events can all be declared static. Static properties are often used to return data that is common to all instances, or to keep track of how many instances of a class have been created. Static methods are often used to provide utilities that relate to the type in some way, such as comparison functions.

To declare a static member you use the static keyword before the return type of the member, as shown by the following example:

// Static Members in Non-static Classes  
public class DrinksMachine  
{  
   public int Age { get; set; }  
   public string Make { get; set; }  
   public string Model { get; set; }  
   public static int CountDrinksMachines()  
   {  
      // Add method logic here.  
   }  
}

Regardless of how many instances of your class exist, there is only ever one instance of a static member. You do not need to instantiate the class in order to use static members. You access static members through the class name rather than the instance name, as shown by the following example:

// Access Static Members  
int drinksMachineCount = DrinksMachine.CountDrinksMachines();