**Delegates:**

* A delegate type is declared using the keyword delegate.
* The general form of a delegate declaration is shown here:

**delegate ret-type name(parameter-list);**

* Here, ret-type is the type of value returned by the methods that the delegate will be calling.
* The name of the delegate is specified by name. The parameters required by the methods called through the delegate are specified in the parameter-list.
* Once created, a delegate instance can refer to and call methods whose return type and parameter list match those specified by the delegate declaration.
* A key point to understand is that a delegate can be used to call any method that agrees with its signature and return type. Furthermore, the method can be either an instance method associated with an object or a static method associated with a class.
* All that matters is that the return type and signature of the method agree with those of the delegate.
* To see delegates in action, let’s begin with the simple example shown here:

// A simple delegate example.

using System;

// Declare a delegate type.

**delegate** **string** **StrMod(string str);**

**class** DelegateTest

{

// Replaces spaces with hyphens.

**static string ReplaceSpaces(string s)**

{

Console.WriteLine("Replacing spaces with hyphens.");

return s.Replace(' ', '-');

}

// Remove spaces.

**static string RemoveSpaces(string s)**

{

string temp = "";

int i;

Console.WriteLine("Removing spaces.");

for(i=0; i<s.Length; i++)

if(s[i] != ' ') temp += s[i];

return temp;

}

// Reverse a string.

**static string Reverse(string s)**

{

string temp = "";

int i, j;

Console.WriteLine("Reversing string.");

for(j=0, i=s.Length-1; i>= 0; i--, j++)

temp += s[i];

return temp;

}

}

**class y**

{

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

{

// Construct a delegate.

**StrMod strOp = new StrMod(ReplaceSpaces);**

string str;

// Call methods through the delegate.

str = **strOp**("This is a test.");

Console.WriteLine("Resulting string: " + str);

Console.WriteLine();

**strOp** = **new StrMod(RemoveSpaces);**

str = **strOp**("This is a test.");

Console.WriteLine("Resulting string: " + str);

Console.WriteLine();

**strOp = new StrMod(Reverse);**

str = **strOp**("This is a test.");

Console.WriteLine("Resulting string: " + str);

}

}

**The output from the program is shown here:**

Replacing spaces with hyphens.

Resulting string: This-is-a-test.

Removing spaces.

Resulting string: Thisisatest.

Reversing string.

Resulting string: .tset a sisihT

* Let’s examine this program closely. The program declares a delegate type called
* StrMod, shown here:

**delegate string StrMod(string str);**

Notice that StrMod takes one string parameter and returns a string.

* Next, in DelegateTest, three static methods are declared, each with a single parameter of type string and a return type of string. Thus, they match the StrMod delegate.
* These methods perform some type of string modification. Notice that ReplaceSpaces( ) uses one of string’s methods, called Replace( ), to replace spaces with hyphens.
* In Main( ), a StrMod reference called strOp is created and assigned a reference to ReplaceSpaces( ). Pay close attention to this line:

**StrModstrOp = new StrMod(ReplaceSpaces);**

Notice how the method ReplaceSpaces( ) is passed as a parameter. Only its name is used; no parameters are specified. This can be generalized.

* When instantiating a delegate, you specify only the name of the method to which you want the delegate to refer. Of course, the method’s signature must match that of the delegate’s declaration. If it doesn’t, a compile-time error will result.
* Next, ReplaceSpaces( ) is called through the delegate instance strOp, as shown here:

**str = strOp("This is a test.");**

Because strOp refers to ReplaceSpaces( ), ReplaceSpaces( ) is invoked.

* Next, strOp is assigned a reference to RemoveSpaces( ), and then strOp is called again. This time, RemoveSpaces( ) is invoked.
* Finally, strOp is assigned a reference to Reverse( ) and strOp is called. This results in Reverse( ) being called.
* The key point of the example is that the invocation of strOp results in a call to the method referred to by strOp at the time at which the invocation occurs.
* Thus, the method to call is resolved at runtime, not compile time.