5. Delegates

Delegates

- A delegate is a type that represents references to methods with a particular parameter list and return type.
- When you instantiate a delegate, you can associate its instance with any method with a compatible signature and return type.
- Delegates are used to pass methods as arguments to other methods.
- You can invoke (or call) the method through the delegate instance.

Delegate Declaration

Syntax:
delegate <return type> <delegate-name> <parameter list>

Sample delegate declaration:
public delegate double Calculate(double x, double y);

- Once a delegate type is declared, a delegate object must be created with the **new** keyword and be associated with a particular method.
- When creating a delegate, the argument passed to the new expression is written similar to a method call, but without the arguments to the method.

```
public delegate void printFile(string s);
...
printFile pf1 = new printFile(PrintFileA);
printFile pf2 = new printFile(PrintFileB);
```

In this case, *PrintFileA* and *PrintFileB* are methods to be referenced by the delegate *printFile*.

```
using System;

delegate int NumberChanger(int n);
namespace DelegateAppl
{
   class TestDelegate
   {
      static int num = 10;
      public static int AddNum(int p)
      {
         num += p;
         return num;
```

```
public static int MultNum(int q)
      {
        num *= q;
        return num;
      }
      public static int getNum()
      {
        return num;
      }
```

```
static void Main(string[] args)
      {
        //create delegate instances
        NumberChanger nc1 = new
NumberChanger(AddNum);
        NumberChanger nc2 = new
NumberChanger(MultNum);
```

```
//calling the methods using the delegate objects
            nc1(25);
            Console.WriteLine("Value of Num: {0}", getNum());
            nc2(5);
            Console.WriteLine("Value of Num: {0}", getNum());
            Console.ReadKey();
        }
    }
}
```

Multicast Delegates

- Delegate objects can be composed using the "+" operator.
- A composed delegate calls the two delegates it was composed from.
- Only delegates of the same type can be composed.
- The "-" operator can be used to remove a component delegate from a composed delegate.
- Using this property of delegates, one can create an invocation list of methods that will be called when a delegate is invoked.
- This is what is referred to as **multicasting** of a delegate.

```
using System;

delegate int NumberChanger(int n);
namespace DelegateAppl
{
   class TestDelegate
   {
      static int num = 10;
      public static int AddNum(int p)
      {
         num += p;
         return num;
```

```
public static int MultNum(int q)
      {
        num *= q;
        return num;
      }

      public static int getNum()
      {
        return num;
      }
```

```
static void Main(string[] args)
      {
        //create delegate instances
        NumberChanger nc;
        NumberChanger nc1 = new
NumberChanger(AddNum);
        NumberChanger nc2 = new
NumberChanger(MultNum);
        nc = nc1;
        nc += nc2;
```

```
//calling multicast
        nc(5);
        Console.WriteLine("Value of Num: {0}", getNum());
        Console.ReadKey();
    }
  }
}
```