A Delegate is a type the represents a reference to a method with a specified signature. It is essentially a type-safe function pointer.

Delegates are useful when you want to pass a method as a parameter to another method, or when you want to define an event handler. They are also used to implement callbacks and to execute methods asynchronously.

A variable defined as a delegate is a reference-type variable that can hold a reference to a method. For a delegate to reference a particular method, the delegate must define parameters with types that match the parameter types contained in the relevant method. The delegate must also define a return type that matches the return type of the relevant method.

When a developer instantiates a delegate, the developer can associate its instance with any method with compatible parameters and return type. The developer can invoke or call the method through the delegate instance. Note that variance can be used in delegates, which means that the types defined for the parameter list and return type of the delegate do not have to match exactly with the relevant method’s parameter types and return type.



In this example, MyDelegate is a delegate type that takes a string parameter and returns void. Method1 and Method2 are two methods that match the signature of MyDelegate. We create an instance of MyClass, and then create three instances of MyDelegate that reference Method1, Method2 and both Method1 and Method2, respectively. Finally, we invoke each delegate instance with a string argument.