**Delegates:**

* A delegate is an object that points to a method and knows to call that method.
* It has a return type and parameter types.
  + Example : delegate int Transformer(int x)

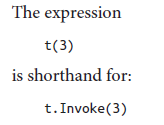
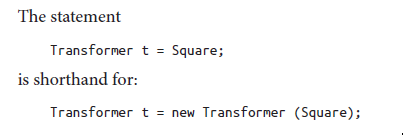
The above delegate takes int parameter and returns an int type. So it can point to any method that takes one int parameter and returns int type like this one 🡪

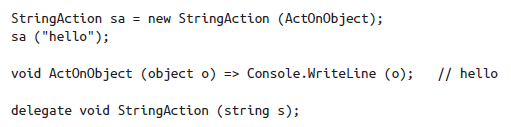
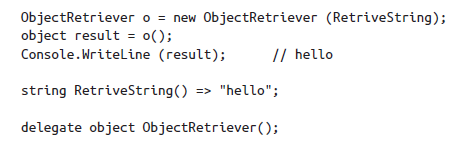
int square (int x) => x \* x;

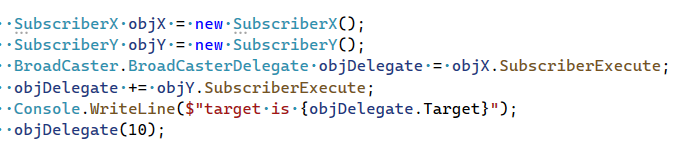
Transformer objDelegate = square ; // assigning a method to delegate variable creates delegate instance

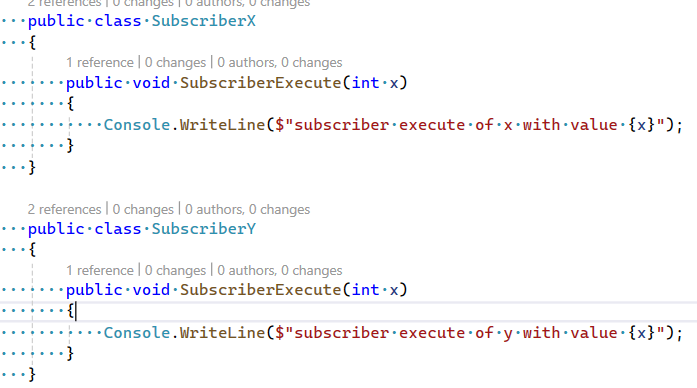
objDelegate(3); // invokes the instance.

A delegate instance acts as a delegate by calling the target method. In this way the caller is loosely coupled from the target method which the caller wants to call.



* *Contravariance*: When we are passing more specific types as argument than asked for to the parameters of the method. A delegate can have more specific parameter types than it’s target method. 
* *Covariance*: When we get more specific return type than we asked for. A delegate’s target method can return more specific return type than described by the delegate. 
* Broadcaster and Subscriber pattern:





The problem with above code is subscribers can do the following:

* Replace other subscribers by reassigning the ‘objDelegate’
* Clear all subscribers by assigning objDelegate to null
* Broadcast to all subscribers by invoking the delegate

**Events:**