Memory leak via event handlers

EventSink sink = new EventSink();

EventSource src = new EventSource();

src.SomeEvent += sink.EventHandler;

src.DoWork();

sink = null;

// Force collection

GC.Collect();

GC.WaitForPendingFinalizers();

For the above code if you have a finalizer for EventSink and add a breakpoint/Console.WriteLine in it, you’d see that it is never hit even though sink is clearly set to null. The reason is the 3rd line where we added sink to the invoke list of source via the += operator. So even after sink being set to null the original object is still reachable (and hence not garbage) from src.

+= is additive so as the code executes and new sinks are added the older objects still stick around resulting in working set to grow and finally lead to crash at some point.

The fix is to ensure you remove the sink with –= as in

EventSink sink = new EventSink();

EventSource src = new EventSource();

src.SomeEvent += sink.EventHandler;

src.DoWork();

src.SomeEvent -= sink.EventHandler;

sink = null;