Profiler.cs

```csharp
using System;
using System.Diagnostics;
using System.Threading;

namespace Benchmarking {
  /// <summary>
  /// Make sure you compile in Release with optimizations enabled, and Run the
  /// tests outside of Visual Studio (This part is important because the JIT
  /// stints its optimizations with a debugger attached, even in Release mode).
  /// </summary>
  static public class Profiler {
    static public double Profile(string desc, uint iterations, Action func) {
      //Run at highest priority to minimize fluctuations caused by others
      Process.GetCurrentProcess().PriorityClass = ProcessPriorityClass.High;
      Thread.CurrentThread.Priority = ThreadPriority.Highest;

      // Warm up
      func();

      // Clean up
      GC.Collect();
      GC.WaitForPendingFinalizers();
      GC.Collect();//To make sure the "finalized" objects are also collected.

      var watch = Stopwatch.StartNew();
      for (uint i = 0; i < iterations; i++) {
        func();
      }
      watch.Stop();
      double elapsedTime = watch.Elapsed.TotalMilliseconds;
      if (desc != null) Console.WriteLine("{0,-40}\t{1,15:n} ms", desc,
        elapsedTime);
      return elapsedTime;
    }

    static public double Profile(string desc, uint iterations, Action func,
            out int gcCount) {
      Process.GetCurrentProcess().PriorityClass = ProcessPriorityClass.High;
      Thread.CurrentThread.Priority = ThreadPriority.Highest;

      func();

      GC.Collect();
      GC.WaitForPendingFinalizers();
      GC.Collect();

      gcCount = GC.CollectionCount(0);
      var watch = Stopwatch.StartNew();
      for (uint i = 0; i < iterations; i++) {
        func();
      }
      watch.Stop();
      gcCount = GC.CollectionCount(0) - gcCount;
      double elapsedTime = watch.Elapsed.TotalMilliseconds;
      if (desc != null) Console.WriteLine("{0,-40}\t{1,15:n} ms",
            desc, elapsedTime);
      return elapsedTime;
    }
  }
}
```