```
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading;

namespace PubSub.Implementation
{
    #region Publisher

    public class Publisher
    {
        const bool HasBug = false;
        Random random = new Random();
        static List<Thread> workers = new List<Thread>();
        static int instanceCount; //for reference counting of publishers
        int count;

        internal List<Subscriber> subscribers;

        public Publisher()
        {
            subscribers = new List<Subscriber>();
            count = instanceCount++;
        }

        public override string ToString()
        {
            return "pub#" + count;
        }

        public void Publish(string data)
        {
            Console.WriteLine(this + " publishing " + data);
            foreach (Subscriber sub in subscribers)
                if (!HasBug)
                    sub.RaiseReceived(data);
                else
                    new Sender(sub, data);
        }
    }

    class Sender  //introduces a bug
    {
        static List<Thread> senderThreads = new List<Thread>();
        static Random randomGen = new Random();
        Subscriber destination;
        string data;

        internal Sender(Subscriber destination, string data)
        {
            this.destination = destination;
            this.data = data;
            Thread thread = new Thread(new ThreadStart(Run));
            senderThreads.Add(thread);
            thread.Start();
        }

        internal void Run()
        {
            Console.WriteLine("sender for " + destination + " started");
            Thread.Sleep(randomGen.Next(500));
            destination.RaiseReceived(data);
        }

        static void Reset()
        {
```

```
            Console.WriteLine("reset");
            foreach (Thread t in senderThreads)
            {
                try
                {
                    t.Abort();
                    t.Join();
                }
                catch { }
            }
            senderThreads.Clear();
        }
    }

    #endregion

    #region Subscriber

    public delegate void ReceivedEventHandler(string data);

    public class Subscriber
    {
        static int instanceCount; //ref counting of subscribers
        int count;

        public event ReceivedEventHandler Received;

        public override string ToString()
        {
            return "sub#" + count;
        }

        public Subscriber(Publisher pub)
        {
            count = instanceCount++;
            pub.subscribers.Add(this);
        }

        internal void RaiseReceived(string data)
        {
            if (Received != null)
                Received(data);
        }
    }

    #endregion
}
```