using System;

using System.Collections.Generic;

using System.Diagnostics.Contracts;

using System.IO;

using System.Linq;

using System.Reflection;

using System.Security;

using System.Security.Permissions;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace ConsoleApp2

{

#region Semaphore

//public class Program

//{

// static void Main(string[] args)

// {

// Semaphore semaphore = new Semaphore(3,15,"SEMAPHORE");

// for (int i = 0; i < 10; i++)

// {

// ThreadPool.QueueUserWorkItem(SomeMethod, semaphore);

// }

// Console.ReadLine();

// }

// private static void SomeMethod(object state)

// {

// var s = state as Semaphore;

// bool st = false;

// while (!st)

// {

// if (s.WaitOne(500))

// {

// try

// {

// Console.WriteLine($"{Thread.CurrentThread.ManagedThreadId} got the key");

// Thread.Sleep(2000);

// }

// finally

// {

// st = true;

// Console.WriteLine($"{Thread.CurrentThread.ManagedThreadId} returned key");

// s.Release();

// }

// }

// else

// {

// s.Release(1);

// Console.WriteLine($"{Thread.CurrentThread.ManagedThreadId} we do not have enought keys , please wait");

// }

// }

// }

//}

#endregion

#region SemaphoreSlim

class Program

{

static SemaphoreSlim \_semaSlim = new SemaphoreSlim(4);

static void Main(string[] args)

{

for (int i = 0; i < 6; i++)

{

var name = $"Thread {i}";

int seconds = 2 + 2 \* i;

var t = new Thread(() =>

{

AccessDatabase(name, seconds);

});

t.Start();

}

}

private static void AccessDatabase(string name, int seconds)

{

Console.WriteLine($"{name} waits for access");

\_semaSlim.Wait();

Console.WriteLine($"{name} is working on database");

Thread.Sleep(seconds\*1000);

Console.WriteLine($"{name} completed its work");

\_semaSlim.Release();

}

}

#endregion

}