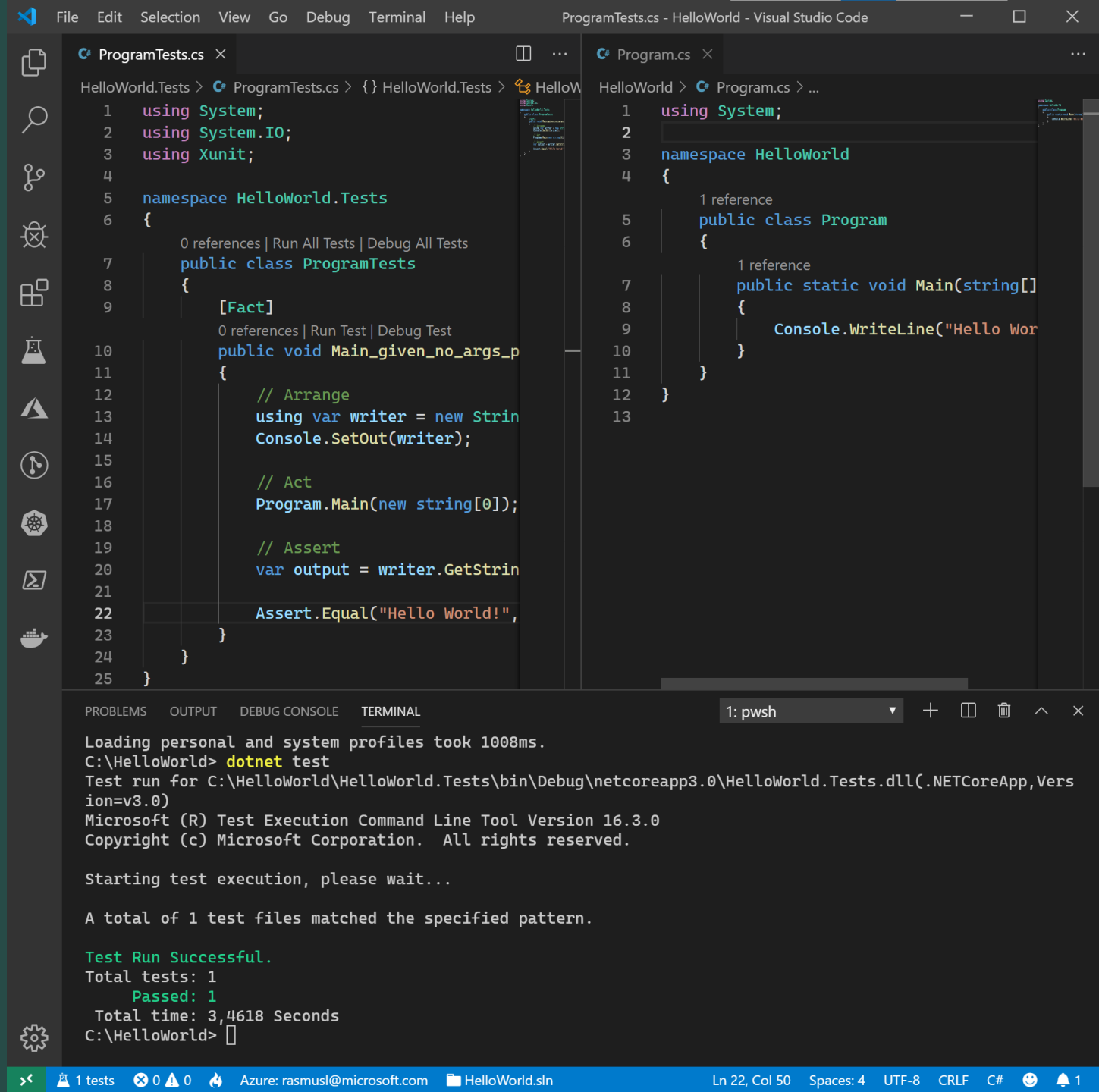
Asynchronous and Parallel Programming in C#

Rasmus Lystrøm
Associate Professor
ITU
rnie@itu.dk



Agenda

Multithreading

Concurrency

Threads

Task Parallel Library

Asynchronous Programming

Async ≠ Parallel ≠ Threads

Multithreading

Multithreading

Enables executing several pieces of code simultaneously

Leverage multicore CPUs

Speed

The operating system decides the order

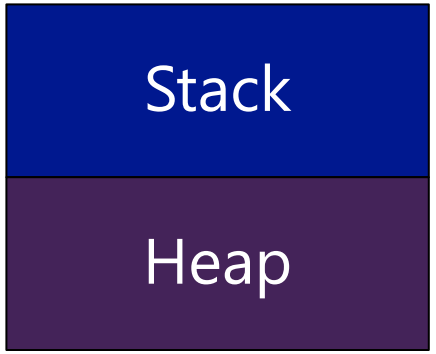
Concurrency

Concurrency

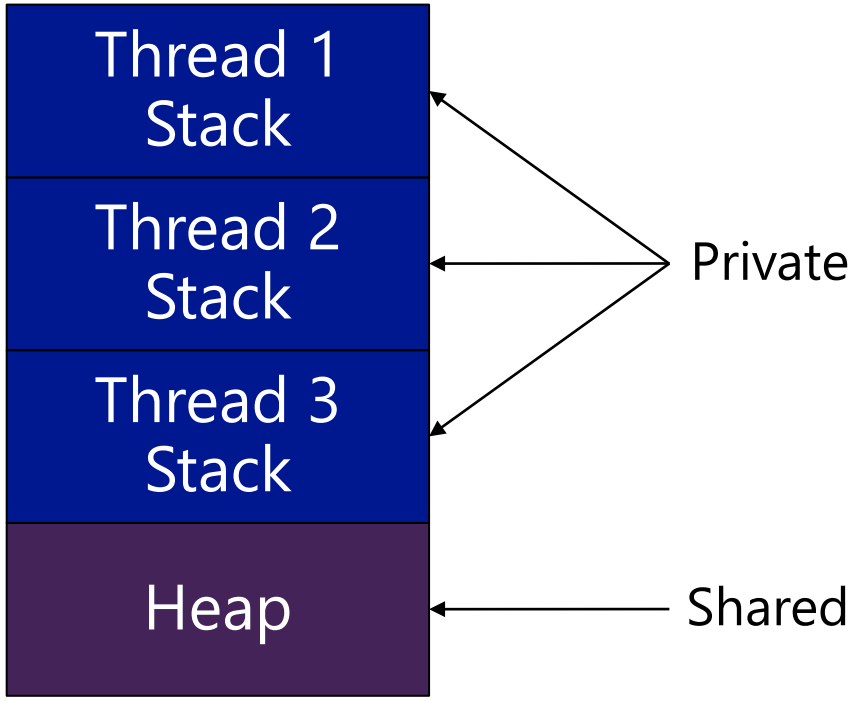
A property of systems in which several computations are executing **simultaneously**, and potentially interacting with each other. The computations may be executing on multiple cores in the same chip, preemptively time-shared threads on the same processor, or executed on physically separated processors.

Threads

Threads

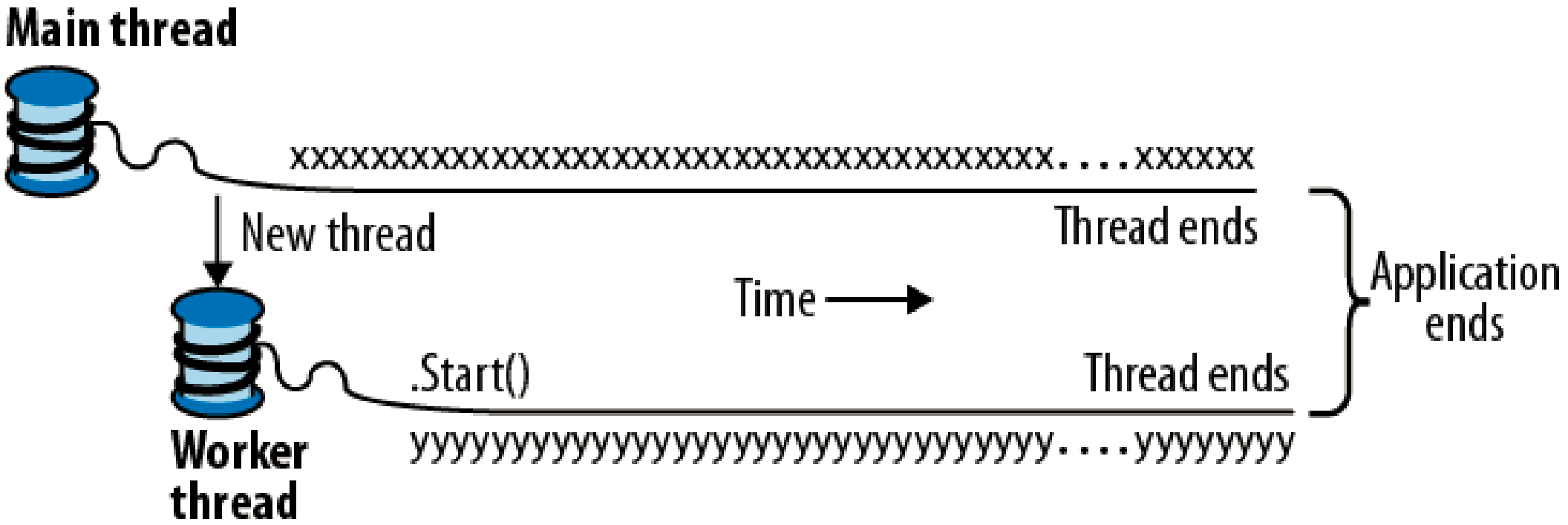


Single Threaded Program



Multithreaded Program

Threads Example



Threads

Demo

Race Condition

Race Condition

Behavior of a program where the output is **dependent** on the **sequence** or **timing** of other **uncontrollable** events.

→ Bug, when events do not happen in the order the programmer intended.

Race Condition

Demo

Deadlock

Deadlock

A situation in which two or more competing actions are each waiting for the other to finish, and thus neither ever does.

Deadlock

Demo

Task Parallel Library

Task.Run

Task.Factory…

Task.Delay

Parallel.For

Parallel.ForEach

Parallel.Invoke

Parallel Linq → .AsParallel()

Task Parallel Library

Demo

System.Collections.Concurrent

ConcurrentQueue<T>

ConcurrentStack<T>

BlockingCollection<T>

ConcurrentDictionary<TKey, TValue>

Asynchronous Programming

Asynchronous Programming

Asynchronous programming is a means of parallel **programming** in which a unit of work runs separately from the main application thread and notifies the calling thread of its completion, failure or progress.

async/await

`async` →

Method must return `void`, `Task`, `Task<T>`, or a task-like type.

Specifically: a type, which satisfy the `async` pattern, meaning a `GetAwaiter` method must be accessible.

`await` → Await task(s)…

Note: `Main` and *test* methods must return `Task`

async/await

Demo

Speed
Multiprocessor
Parallel execution

Async ≠ Parallel ≠ Threads

Non-blocking UI,
background tasks,
asynchronous

Low-level building
block
Do not use directly!