

# Parallelism

## Synchronous mechanism

Peter Levinsky IT, Roskilde

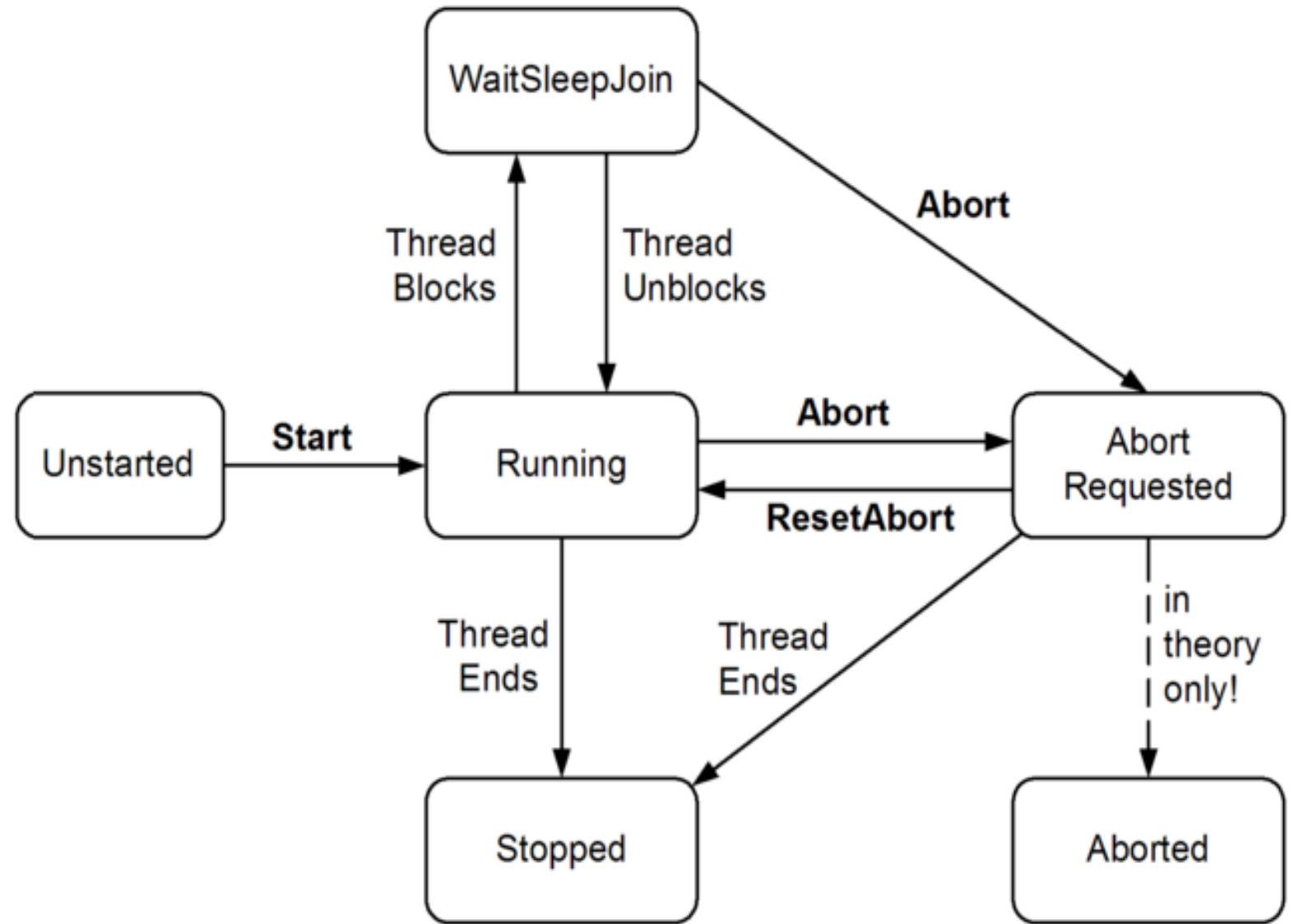
28.03.2023

# Time consuming operations

## Two categories

- CPU-bound operations
- I/O-bound operations

# Thread Life cycle



# Thread in C#

```
Thread t = new Thread (-- delegate Method --);  
t.Start();
```

...

```
t.Join(); // wait here until t is completed
```

? Delegate Method

# Thread in C# - executing

```
class ThreadTest
{
    static bool done;    // Static fields are shared between all threads

    static void Main()
    {
        new Thread (Go).Start();
        Go();
    }

    static void Go()
    {
        if (!done) { done = true; Console.WriteLine ("Done"); }
    }
}
```

# Parallelism in C# - An Overview

## Levels of parallelism:

- Thread -- Basic structure for parallelism (in most programming languages)
- Task -- C# smooth variant i.e. `Task.Run(<<delegate method>>)`
- `Parallel.Invoke` -- Can start several threads (continues after all thread is completed)
- `Parallel.For/ForEach` -- Can start several threads in a loop (continues after all thread is completed)
- `Plinq` -- Can execute a Linq expression in parallel

# High End Parallelisme **async / await**

- Use of built in features **async / await**

Do not create a new thread but make use of a coroutine i.e. program continue and 'jumps' back to the await call when it is ready.

- Where to use

- I/O-bound operations – Like network, accessing files etc.

- How to use

- Method is async – like public **async Task<int>** DoSomethingAsync()

- In method body ... somewhere

**await** ..... return anInteger;

**Good Practice**



# What is Async / Await ?

- The use of Async / Await is **not** directly the same as a **thread** / task !
- But the program will wait at 'await' until this job is done
- And you can continue do other stuff in between  
e.g. show information about 'work in progress' (Jacob Nielsen – System status)

```
Task<List<Picture>> pictures = await ReadPicturesFromFile("somefile.pic");  
Status = "Getting pictures ..."; // set system status  
foreach(var pic in pictures.Result){  
    ...  
}
```



# Demo

Opgaver C#Exercises Prog.3.6+3.7