

MAS: Betriebssysteme

Prozesse und Threads
Ergänzung: Threads in C#

T. Pospíšek

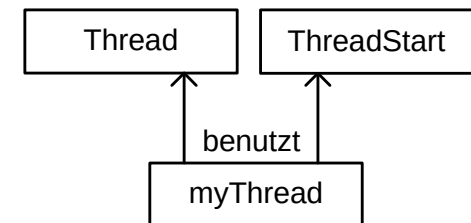
Threads in C#

Namespace System.Threading

- In diesem Namespace werden Basismechanismen für Threads bereitgestellt

namespace System.Threading

```
{  
    public delegate void ThreadStart();  
    public enum ThreadState  
        { Running=0, ..., Stopped=16, .., Suspended=64,..., Aborted=256}  
    ...  
    public sealed class Thread { ... }  
    public sealed class Monitor { ... }  
    public class ThreadStateException { ... }  
    public class ThreadAbortException { ... }  
    public class ThreadInterruptedException { ... }  
    public class SynchronizationLockException { ... }  
}
```



Threads in C#

Die Klasse Thread

■ Vorgegebene Thread-Klasse

```
public sealed class Thread {  
    public Thread(ThreadStart start);  
    public void Start();  
    public bool Join(int msec);  
    public static void Sleep(int msec);  
    public void Abort();  
  
    public void ResetAbort();  
    public void Interrupt()  
  
    public void Suspend();  
    public void Resume();  
    ...}
```

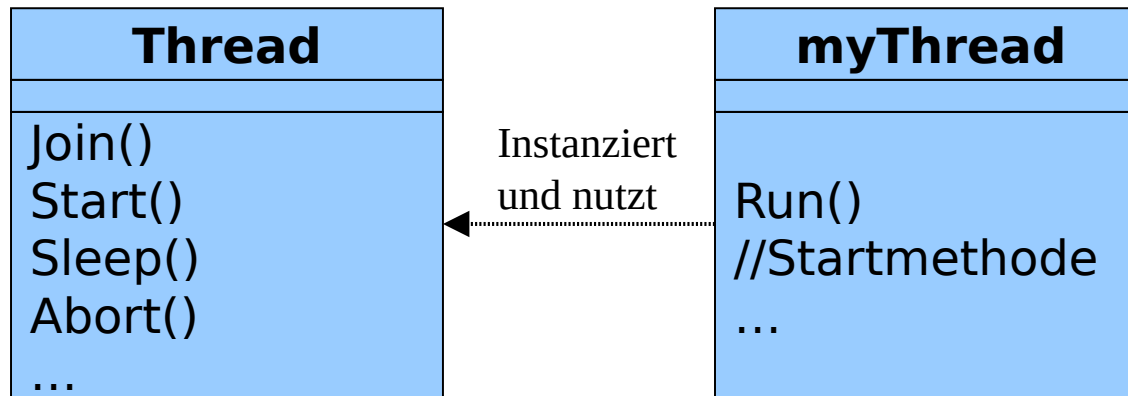
// Thread starten, Startmethode wird aktiviert
// Auf Ende des Threads warten
// Thread msec Millisekunden anhalten
// Auslösen einer Ausnahme vom Typ
// ThreadAbortException
// Abort zurücknehmen
// Thread unterbrechen, wenn eine Ausnahme
// vom Typ ThreadInterruptedException
// geworfen wird
// Thread suspendieren
// Thread wieder anstarten (nach einer Suspension)

Threads in C#

Nutzung von Threads

- Eigene Klasse nutzt Thread-Klasse
- Thread wird instanziiert
- Startmethode wird zugewiesen

Aus Namespace
System.Threading



Threads in C#

Beispielnutzung

- Keine Vererbung, Startmethode an Thread übergeben

```
using System.Threading;
class myThreadClass {
    public void myThreadClass() { .. }    // Konstruktor
    public static void Main() {
    {
        ThreadStart startMethod = new ThreadStart(Run);    // Startmethode festlegen
        Thread myThread = new Thread(startMethod);          // Neuen Thread erzeugen
        myThread.Name = ("myThread");                       // Thread erhält einen Namen
        myThread.Start();                                    // Neuer Thread wird gestartet
        ...                                                  // Erzeugender Thread macht etwas anderes
        myThread.Join();                                    // Warten, bis sich neuer Thread beendet hat
    }
    public void Run()                                       // Startmethode des Threads
    {
        // Aktionen des Threads müssen hier programmiert werden
    }
}
```

Gesamtüberblick

- ✓ Einführung in Computersysteme
- ✓ Entwicklung von Betriebssystemen
- ✓ Architekturansätze
- ✓ Interruptverarbeitung in Betriebssystemen
- ✓ **Prozesse und Threads**
- 5. CPU-Scheduling
- 6. Synchronisation und Kommunikation
- 7. Speicherverwaltung
- 8. Geräte- und Dateiverwaltung
- 9. Betriebssystemvirtualisierung