LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR MATHEMATIK, INFORMATIK UND STATISTIK
INSTITUT FÜR INFORMATIK
LEHRSTUHL FÜR DATENBANKSYSTEME UND DATA MINING

Nebenläufigkeit

SEP

Emanuel Böse

23.10.2017

Wissenschaftlicher Betreuer:
Janina Sontheim, Daniel Kaltenthaler, Johannes Lohrer
Verantwortlicher Professor:
Prof. Dr. Peer Kröger

DBS

Einführung

- Prozesse und Threads
- Nebenläufigkeit
- Probleme bei nebenläufigen Prozessen/Threads
- Synchronisation mit Monitoren
- Nebenläufige Programmierung in Java

Prozesse und Threads

Prozesse

- Besitzen ein Programm welches ausgeführt wird
- Besitzen einen eigenen Speicherbereich
- Werden durch einen Scheduler gesteuert

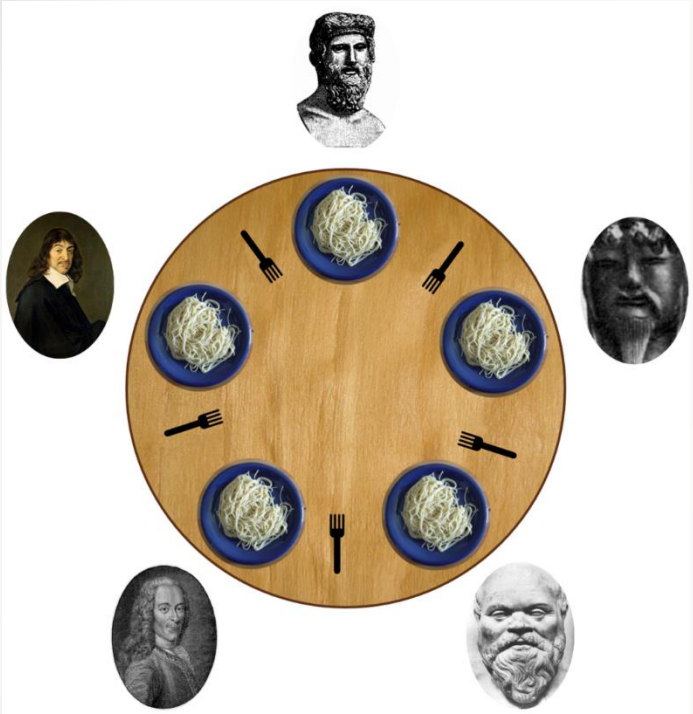
Threads

- Prozesse bestehen aus mindestens einem Thread
- Alle Threads eines Prozesses teilen sich dessen zugewiesenen Speicherbereich
- Sind eine Ausführungsreihenfolge

Nebenläufigkeit

- Eigenschaft eines Systems mehrere Berechnungen oder Befehle gleichzeitig auszuführen
- Kann unabhängige oder gemeinsame Aufgaben behandeln
- Operationen können **scheinbar** oder **echt** nebenläufig auf einem Prozessor ausgeführt werden (Einzel/Mehrkernprozessoren)

Probleme bei Nebenläufigkeit

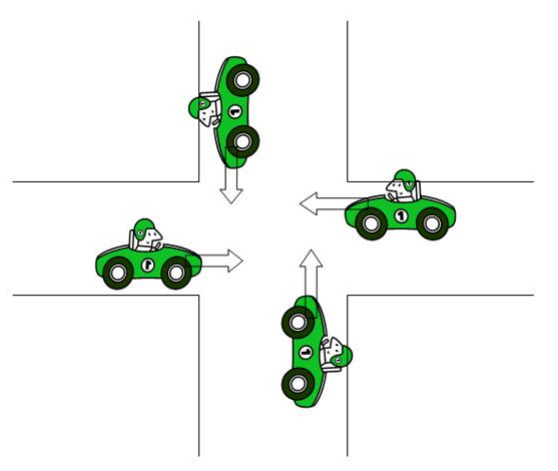


- „Philosophenproblem“
- Bei Hunger greift einer der Philosophen seine linke sowie rechte Gabel und legt diese zurück wenn gesättigt
- Problem: Wenn alle zur gleichen Zeit hungrig sind
- Philosophen verhungern

Probleme bei Nebenläufigkeit

Verklemmung

- 2 Prozesse benötigen ein und dasselbe Betriebsmittel um weiterarbeiten zu können



Aushungern

- Prozess B wartet darauf, das Prozess A ein Betriebsmittel freigibt
- Prozess A ist bereits terminiert
- Prozess B wartet ewig und „verhungert“

Monitore

- Konzept zur Synchronisation zeitlich verschränkt oder parallel laufender Prozess/Threads auf gemeinsam genutzte Ressourcen

```
Name n;

//-------------

// thread 1:
  ...
  n.setName("Charlie","Chaplin");
  ...
  n.setName("Jerry","Lewis");
  ...

//-------------

// thread 2:

  ... n.getName() ...
```



```
"Charlie Chaplin"
"Jerry Lewis"
"Jerry Chaplin"
"Charlie Lewis"
```

Monitore

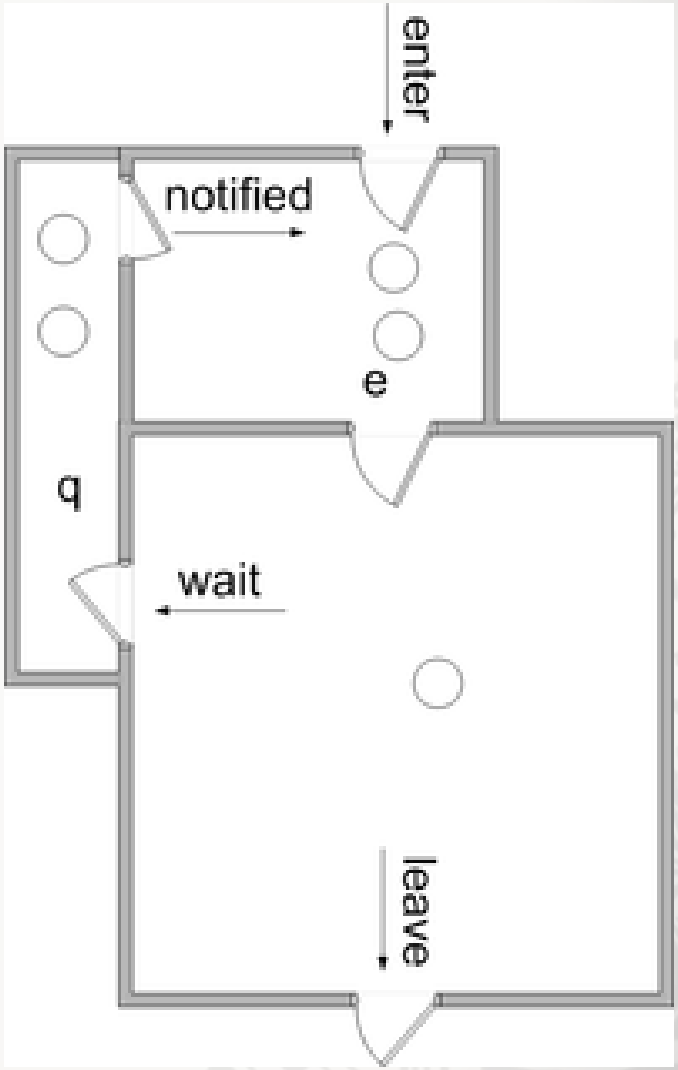
```
class SyncName {
 private String firstname;
 private String surname;

    public synchronized void setName(String firstname,String surname) {
      this.firstname = firstname;
      this.surname   = surname;
  }

    public synchronized String getName() {
      return firstname + " " + surname;
  }
}
```

Monitore

- **wait()** teilt dem Thread mit zu warten bis er ein **notify()** bekommt
- **notify()** weckt einen **beliebigen** Thread auf, welcher an diesem an diesem Objekt wartet
- **notifyAll()** weckt alle wartenden Threads auf, welche auf dieses Objekt warten



Klasse Thread / Interface Runnable

Vererbung

```
public class MyThread extends
Thread {
    public void run() {
        //Code
    }
}
```

Schnittstelle

```
public class MyRunnable
implements Runnable {
    public void run() {
        //Code
    }
}
```

Es ist nicht mehr möglich von einer anderen Klasse zu erben

Beispiele: