Multithreading

Übersicht

- **[illegible]ds**
  - [illegible]lauf
  - Prozesse
  - Threads

- **Probleme mit Threads**
  - Thread Kollision
  - Semaphoren

- **Threads in Java**
  - Die Klasse Thread
  - syncronized, wait und notify

Multithreading

Programmablauf

- … n arbeitet Befehle ab einen nach dem anderen.
- Dabei greift es auf Daten zu.

```
public static void main(String args[]){
    // Als absolute ausnahme (und um euch nicht die Arbeit abzunehmen)
    // gibt es hier eine Ausgabe mit System.out.println() :-)
    System.out.println("SWP Uebung1:");
    BasicConfigurator.configure();
    // Hier kommen die Aufrufe der einzelnen Uebungen rein
    Lars.Uebung1 larsSeinTest = new Lars.Uebung1();
    larsSeinTest.test();
    liwicki.log4j marcustest = new liwicki.log4j();
    marcustest.test();
    basti.log4jtest bastitest = new basti.log4jtest();
    bastitest.test();
    heiko.trojanhorse heikotest = new heiko.trojanhorse();
    heikotest.test();
    richard.Uebung1 xxx = new richard.Uebung1();
    xxx.test();
    raphi.LogTest logiproggy = new raphi.LogTest();
    logiproggy.test();

    anja.Uebung1 anjaslogger = new anja.Uebung1();
    anjaslogger.test();
}
```

Daten

Multithreading

Prozesse

Code und

- Mehrere Prozesse können parallel laufen.

Rechner

Mail-Prog. Prozess

```
public static void main(String args[]){
```

Daten

Browser Prozess

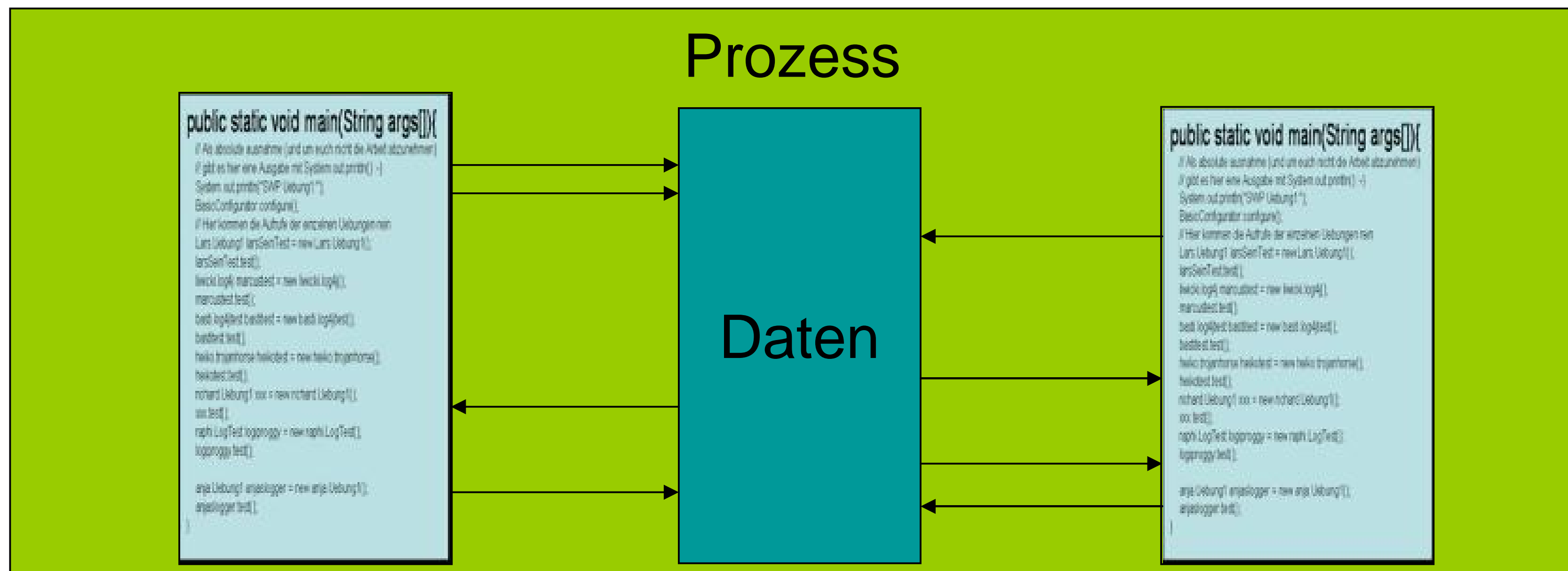
```
public static void main(String args[]){
```

Daten

Multithreading

Threads

- …aten innerhalb eines Prozesses.
- …
- Unabhängig von Objekten oder ähnlichen Datenstrukturen



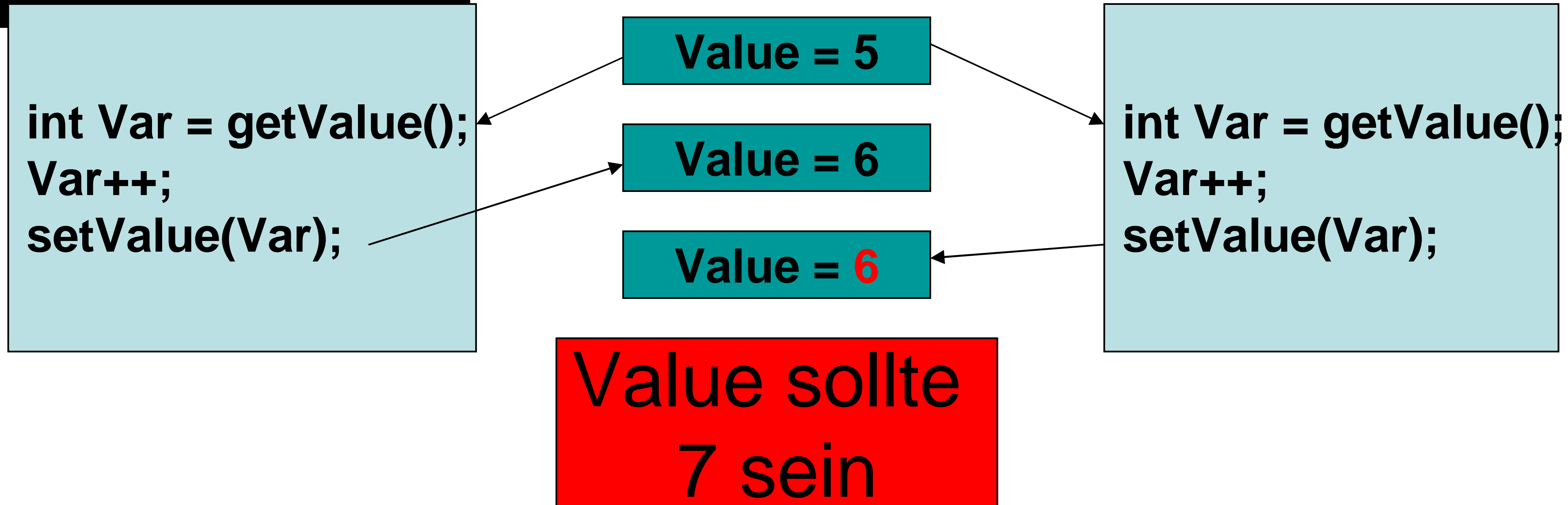
Threads(2)

- … wir Sie den nun ?
- Beispiel: KI mit Chatfunktion
  - Wir haben ein Programm welches einen Spielzug berechnet und dafür einige Minuten benötigt. Dieses bekommt eine Aufforderung und fängt an zu Rechnen. Zwischendurch kommen Chat Nachrichten an. Es gibt jetzt zwei Möglichkeiten.
    - 1. Wir haben in unserem KI Code mittendrin einen Aufruf, der nachkuckt ob Nachrichten da sind (nicht sehr schön).
    - 2. Wir haben einen Thread der immer nach neuen Nachrichten kuckt und diese bearbeitet.
  - Erweiterung: Wir lassen einen weiteren Thread auch einen Spielzug berechnen mit anderen Parametern (Vorteilhaft bei vielen Prozessoren. Einfach, wir rufen den Code nur zweimal auf).

Multithreading

Probleme

s arbeiten gleichzeitig auf den selben Daten

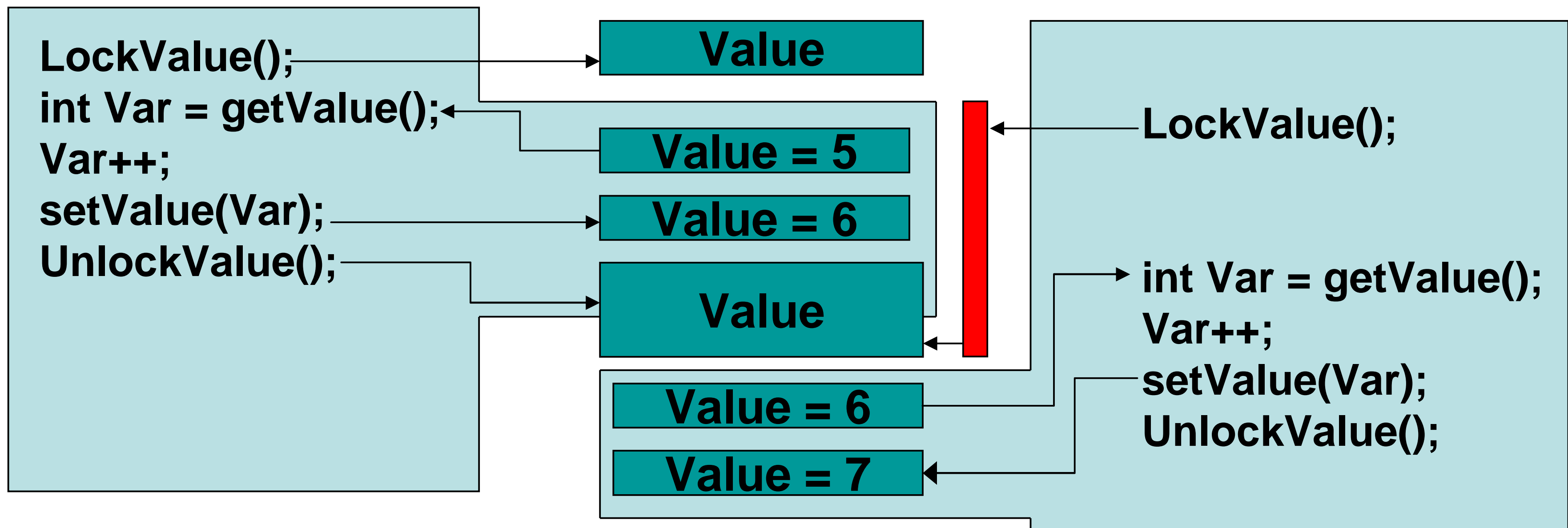


- Lösung : Ein Flag was zeigt ob jemand gerade was an einer Variablen ändert. Eine Semaphore...

Multithreading

Semaphore

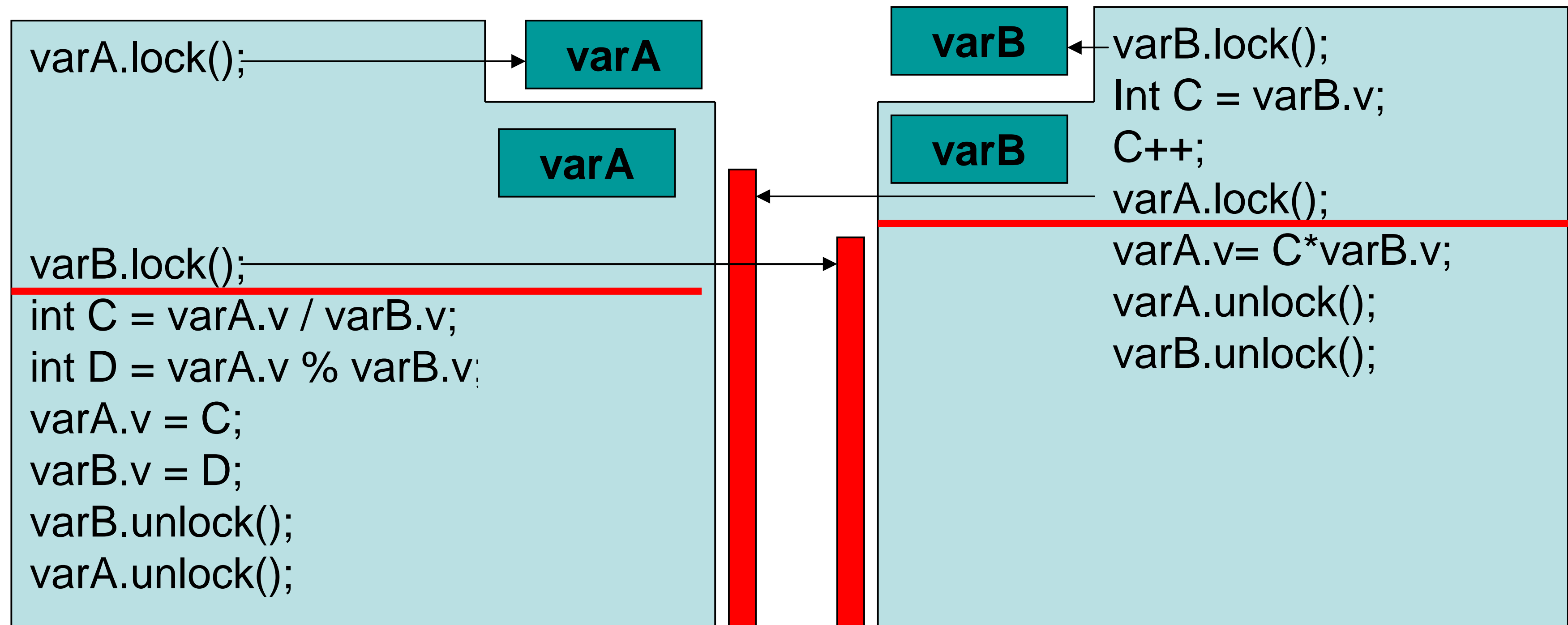
- … s die Semaphore setzten um auf die Daten … ck).
- Nach dem Zugriff, die Semaphore zurücksetzen, damit andere auf die Daten zugreifen können (Unlock).



Multithreading

Neue Probleme

Threads Festhängen weil sie aufeinander
an das einen Deadlock.



- Lösung: Aufpassen, das es nicht passiert (Debugger hilft)!

Multithreading

Threads in Java

- … t fest in Java integriert
- java.lang.Thread / java.lang.Runnable
- Erzeugung eines Threads:
  - Erzeugen einer Klasse die Runnable implementiert
  - Die run-Methode ist der Code den dieser Thread bearbeitet

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

  - Einen Thread mit dem Runnable erzeugen.

```
MyRunnable myRunnable = new MyRunnable();
Thread t = new Thread(myRunnable);
```

  - Den Thread starten

```
t.start();
```

Multithreading

Beispiel in Java

t Chat:

```
public class         nable implements Runnable {
    int p
    public KIRunnable(int p){param=p;}
    public void run(){
        rechneSpielzug(param)
}}
public class ChatRunnable implements Runnable {
    public void run(){
        while(true){handleMessage();}
}}
...
Thread t1 = new Thread(new KIRunnable(1));
Thread t2 = new Thread(new KIRunnable(2));
Thread tChat = new Thread(new ChatRunnable());
t1.start();
t2.start();
tChat.start();
```

Multithreading

Threads in Java(2)

- …nn alle Threads gestartet wurden?
- …äuft nach dem starten einfach weiter, aber wie wissen wir wann ein Thread fertig ist.
- Die Methode join() auf der Klasse Thread.

```
T1.join();
t2.join();
berechneErgebnisse();
```

- Unterbrechen eines Threads mit interrupt()
- Weiter Methoden finden sich in der API. Dort gibt es auch nochmal eine kleine Anleitung.

synchronized

- …ugriff auf Objekte.
- …ort für Funktionen.

```
public synchronized void foo(){...}
```

- Lockt das Objekt auf dem die Methode aufgerufen wird.
- Der Lock wird aufgehoben sobald die Methode verlassen wird.
- Ruft ein anderer Thread gleichzeitig eine synchronisierte Methode auf dem selben Objekt auf, so wartet dieser bis der erste fertig ist.
- ACHTUNG: kann zu Deadlocks führen, wenn der Thread mit Lock auf den anderen wartet, während der wartet in eine Methode reinzukommen (Siehe Folie „neue Probleme“).

Multithreading

wait / notify

- …glichkeiten von synchronized.
- Wait gibt den Lock frei und wartet auf ein Notify.
- Normale Verwendung:

  Thread 1

```
while(!nummer2istFertig){
    wait();
}
```

  Thread 2

```
nummer2istFertig=true;
notifyAll();
```

  Hier nix mehr auf dem Objekt machen !!!

Multithreading

Fazit

- nen vieles einfacher machen.
- Threads rufen neue Probleme hervor.
- Es gibt noch einiges was man Wissen kann. Eure Tutoren wissen das, also fragt sie bei Problemen!

FRAGEN ?