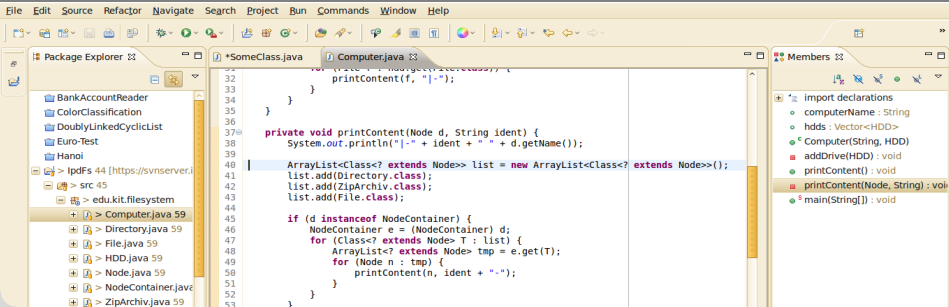


Programmieren-Tutorium Nr. 10 bei Martin Thoma

Five-in-A-Row, Multithreading

Martin Thoma | 12. Februar 2013

FAKULTÄT FÜR INFORMATIK



- 1 Einleitung
- 2 Abschlussaufgaben
- 3 Spiele
- 4 Multithreading
- 5 Abspann

Quiz

```
1 import java.util.ArrayList;
2 import java.util.List;
3
4 public class Main {
5     public static final int BIG_NR = 2000000;
6     public static long bigSum;
7
8     public static void main(String[] args) {
9         List<Thread> threads =
10             new ArrayList<Thread>();
11         for (int i = 0; i < 50; i++) {
12             Runnable task = new Sum(BIG_NR);
13             Thread worker = new Thread(task);
14             worker.start();
15             threads.add(worker);
16         }
17
18         int running = 0;
19         do {
20             running = 0;
21             for (Thread thread : threads) {
22                 if (thread.isAlive()) {
23                     running++;
24                 }
25             }
26             System.out.println("Remaining threads: " + running);
27         } while (running > 0);
28
29         System.out.println(Main.bigSum);
30
31     }
```

```
1 public class Sum implements Runnable {
2     private final int UpperEnd;
3
4     public Sum(int upperEnd) {
5         UpperEnd = upperEnd;
6     }
7
8     @Override
9     public void run() {
10         for (int i = 0; i < UpperEnd; i++) {
11             Main.bigSum++;
12         }
13     }
14 }
```

- `bigSum++;` ist nicht atomar:

3 Operationen: Wert holen, Wert erhöhen, Wert schreiben

⇒ Ergebnis ist zufällig

- Alles im Bereich $[BIG_NR, 50 \cdot BIG_NR]$ ist möglich:

- ① Thread 1 holt sich den Wert von `bigSum`
- ② Thread 2 - 50 laufen durch
- ③ Thread 1 erhöht den zuvor geholten Wert um 1
- ④ Thread 1 überschreibt `bigSum` mit 2
- ⑤ Thread 1 läuft durch

- Sind kleinere Werte möglich?

- `bigSum++;` ist nicht atomar:

3 Operationen: Wert holen, Wert erhöhen, Wert schreiben

⇒ Ergebnis ist zufällig

- Alles im Bereich $[BIG_NR, 50 \cdot BIG_NR]$ ist möglich:

- ① Thread 1 holt sich den Wert von `bigSum`
- ② Thread 2 - 50 laufen durch
- ③ Thread 1 erhöht den zuvor geholten Wert um 1
- ④ Thread 1 überschreibt `bigSum` mit 2
- ⑤ Thread 1 läuft durch

- Sind kleinere Werte möglich?

- `bigSum++;` ist nicht atomar:

3 Operationen: Wert holen, Wert erhöhen, Wert schreiben

⇒ Ergebnis ist zufällig

- Alles im Bereich $[BIG_NR, 50 \cdot BIG_NR]$ ist möglich:

- ① Thread 1 holt sich den Wert von `bigSum`
- ② Thread 2 - 50 laufen durch
- ③ Thread 1 erhöht den zuvor geholten Wert um 1
- ④ Thread 1 überschreibt `bigSum` mit 2
- ⑤ Thread 1 läuft durch

- Sind kleinere Werte möglich?

- `bigSum++;` ist nicht atomar:

3 Operationen: Wert holen, Wert erhöhen, Wert schreiben

⇒ Ergebnis ist zufällig

- Alles im Bereich $[BIG_NR, 50 \cdot BIG_NR]$ ist möglich:

- 1 Thread 1 holt sich den Wert von `bigSum`
- 2 Thread 2 - 50 laufen durch
- 3 Thread 1 erhöht den zuvor geholten Wert um 1
- 4 Thread 1 überschreibt `bigSum` mit 2
- 5 Thread 1 läuft durch

- Sind kleinere Werte möglich?

- `bigSum++;` ist nicht atomar:

3 Operationen: Wert holen, Wert erhöhen, Wert schreiben

⇒ Ergebnis ist zufällig

- Alles im Bereich $[BIG_NR, 50 \cdot BIG_NR]$ ist möglich:

- 1 Thread 1 holt sich den Wert von `bigSum`
- 2 Thread 2 - 50 laufen durch
- 3 Thread 1 erhöht den zuvor geholten Wert um 1
- 4 Thread 1 überschreibt `bigSum` mit 2
- 5 Thread 1 läuft durch

- Sind kleinere Werte möglich?

- `bigSum++;` ist nicht atomar:

3 Operationen: Wert holen, Wert erhöhen, Wert schreiben

⇒ Ergebnis ist zufällig

- Alles im Bereich $[BIG_NR, 50 \cdot BIG_NR]$ ist möglich:

- 1 Thread 1 holt sich den Wert von `bigSum`
- 2 Thread 2 - 50 laufen durch
- 3 Thread 1 erhöht den zuvor geholten Wert um 1
- 4 Thread 1 überschreibt `bigSum` mit 2
- 5 Thread 1 läuft durch

- Sind kleinere Werte möglich?

- `bigSum++;` ist nicht atomar:

3 Operationen: Wert holen, Wert erhöhen, Wert schreiben

⇒ Ergebnis ist zufällig

- Alles im Bereich $[BIG_NR, 50 \cdot BIG_NR]$ ist möglich:

- 1 Thread 1 holt sich den Wert von `bigSum`
- 2 Thread 2 - 50 laufen durch
- 3 Thread 1 erhöht den zuvor geholten Wert um 1
- 4 Thread 1 überschreibt `bigSum` mit 2
- 5 Thread 1 läuft durch

- Sind kleinere Werte möglich?

- `bigSum++;` ist nicht atomar:

3 Operationen: Wert holen, Wert erhöhen, Wert schreiben

⇒ Ergebnis ist zufällig

- Alles im Bereich $[BIG_NR, 50 \cdot BIG_NR]$ ist möglich:

- 1 Thread 1 holt sich den Wert von `bigSum`
- 2 Thread 2 - 50 laufen durch
- 3 Thread 1 erhöht den zuvor geholten Wert um 1
- 4 Thread 1 überschreibt `bigSum` mit 2
- 5 Thread 1 läuft durch

- Sind kleinere Werte möglich?

- `bigSum++;` ist nicht atomar:

3 Operationen: Wert holen, Wert erhöhen, Wert schreiben

⇒ Ergebnis ist zufällig

- Alles im Bereich $[BIG_NR, 50 \cdot BIG_NR]$ ist möglich:

- 1 Thread 1 holt sich den Wert von `bigSum`
- 2 Thread 2 - 50 laufen durch
- 3 Thread 1 erhöht den zuvor geholten Wert um 1
- 4 Thread 1 überschreibt `bigSum` mit 2
- 5 Thread 1 läuft durch

- Sind kleinere Werte möglich?

- `bigSum++;` ist nicht atomar:

3 Operationen: Wert holen, Wert erhöhen, Wert schreiben

⇒ Ergebnis ist zufällig

- Alles im Bereich $[BIG_NR, 50 \cdot BIG_NR]$ ist möglich:

- 1 Thread 1 holt sich den Wert von `bigSum`
- 2 Thread 2 - 50 laufen durch
- 3 Thread 1 erhöht den zuvor geholten Wert um 1
- 4 Thread 1 überschreibt `bigSum` mit 2
- 5 Thread 1 läuft durch

- Sind kleinere Werte möglich?

Quiz: Race-Condition

Sind kleinere Werte als `BIG_NR` für `bigSum` möglich?

Ja:

- 1 Thread 1 holt sich den Wert von `bigSum`
- 2 Thread 2 holt sich den Wert von `bigSum`
- 3 Thread 3 - 50 laufen durch
- 4 Thread 2 läuft bis 1 vorm Ende durch
- 5 Thread 1 erhöht Wert im Register von 0 auf 1 und schreibt 1
- 6 Thread 2 holt sich die 1 aus `bigSum`
- 7 Thread 1 läuft durch
- 8 Thread 2 erhöht Wert im Register von 1 auf 2 und schreibt 2

⇒ Alle Werte in $[2, 50 \cdot \text{BIG_NR}]$ sind möglich!

Wie löst man das Problem?

Quiz: Race-Condition

Sind kleinere Werte als `BIG_NR` für `bigSum` möglich?

Ja:

- ① Thread 1 holt sich den Wert von `bigSum`
- ② Thread 2 holt sich den Wert von `bigSum`
- ③ Thread 3 - 50 laufen durch
- ④ Thread 2 läuft bis 1 vorm Ende durch
- ⑤ Thread 1 erhöht Wert im Register von 0 auf 1 und schreibt 1
- ⑥ Thread 2 holt sich die 1 aus `bigSum`
- ⑦ Thread 1 läuft durch
- ⑧ Thread 2 erhöht Wert im Register von 1 auf 2 und schreibt 2

⇒ Alle Werte in $[2, 50 \cdot \text{BIG_NR}]$ sind möglich!

Wie löst man das Problem?

Quiz: Race-Condition

Sind kleinere Werte als `BIG_NR` für `bigSum` möglich?

Ja:

- 1 Thread 1 holt sich den Wert von `bigSum`
- 2 Thread 2 holt sich den Wert von `bigSum`
- 3 Thread 3 - 50 laufen durch
- 4 Thread 2 läuft bis 1 vorm Ende durch
- 5 Thread 1 erhöht Wert im Register von 0 auf 1 und schreibt 1
- 6 Thread 2 holt sich die 1 aus `bigSum`
- 7 Thread 1 läuft durch
- 8 Thread 2 erhöht Wert im Register von 1 auf 2 und schreibt 2

⇒ Alle Werte in $[2, 50 \cdot \text{BIG_NR}]$ sind möglich!

Wie löst man das Problem?

Quiz: Race-Condition

Sind kleinere Werte als `BIG_NR` für `bigSum` möglich?

Ja:

- ① Thread 1 holt sich den Wert von `bigSum`
- ② Thread 2 holt sich den Wert von `bigSum`
- ③ Thread 3 - 50 laufen durch
- ④ Thread 2 läuft bis 1 vorm Ende durch
- ⑤ Thread 1 erhöht Wert im Register von 0 auf 1 und schreibt 1
- ⑥ Thread 2 holt sich die 1 aus `bigSum`
- ⑦ Thread 1 läuft durch
- ⑧ Thread 2 erhöht Wert im Register von 1 auf 2 und schreibt 2

⇒ Alle Werte in $[2, 50 \cdot \text{BIG_NR}]$ sind möglich!

Wie löst man das Problem?

Quiz: Race-Condition

Sind kleinere Werte als `BIG_NR` für `bigSum` möglich?

Ja:

- 1 Thread 1 holt sich den Wert von `bigSum`
 - 2 Thread 2 holt sich den Wert von `bigSum`
 - 3 Thread 3 - 50 laufen durch
 - 4 Thread 2 läuft bis 1 vorm Ende durch
 - 5 Thread 1 erhöht Wert im Register von 0 auf 1 und schreibt 1
 - 6 Thread 2 holt sich die 1 aus `bigSum`
 - 7 Thread 1 läuft durch
 - 8 Thread 2 erhöht Wert im Register von 1 auf 2 und schreibt 2
- ⇒ Alle Werte in $[2, 50 \cdot \text{BIG_NR}]$ sind möglich!

Wie löst man das Problem?

Quiz: Race-Condition

Sind kleinere Werte als `BIG_NR` für `bigSum` möglich?

Ja:

- ① Thread 1 holt sich den Wert von `bigSum`
 - ② Thread 2 holt sich den Wert von `bigSum`
 - ③ Thread 3 - 50 laufen durch
 - ④ Thread 2 läuft bis 1 vorm Ende durch
 - ⑤ Thread 1 erhöht Wert im Register von 0 auf 1 und schreibt 1
 - ⑥ Thread 2 holt sich die 1 aus `bigSum`
 - ⑦ Thread 1 läuft durch
 - ⑧ Thread 2 erhöht Wert im Register von 1 auf 2 und schreibt 2
- ⇒ Alle Werte in $[2, 50 \cdot \text{BIG_NR}]$ sind möglich!

Wie löst man das Problem?

Quiz: Race-Condition

Sind kleinere Werte als `BIG_NR` für `bigSum` möglich?

Ja:

- ① Thread 1 holt sich den Wert von `bigSum`
 - ② Thread 2 holt sich den Wert von `bigSum`
 - ③ Thread 3 - 50 laufen durch
 - ④ Thread 2 läuft bis 1 vorm Ende durch
 - ⑤ Thread 1 erhöht Wert im Register von 0 auf 1 und schreibt 1
 - ⑥ Thread 2 holt sich die 1 aus `bigSum`
 - ⑦ Thread 1 läuft durch
 - ⑧ Thread 2 erhöht Wert im Register von 1 auf 2 und schreibt 2
- ⇒ Alle Werte in $[2, 50 \cdot \text{BIG_NR}]$ sind möglich!

Wie löst man das Problem?

Quiz: Race-Condition

Sind kleinere Werte als `BIG_NR` für `bigSum` möglich?

Ja:

- ① Thread 1 holt sich den Wert von `bigSum`
- ② Thread 2 holt sich den Wert von `bigSum`
- ③ Thread 3 - 50 laufen durch
- ④ Thread 2 läuft bis 1 vorm Ende durch
- ⑤ Thread 1 erhöht Wert im Register von 0 auf 1 und schreibt 1
- ⑥ Thread 2 holt sich die 1 aus `bigSum`
- ⑦ Thread 1 läuft durch
- ⑧ Thread 2 erhöht Wert im Register von 1 auf 2 und schreibt 2

⇒ Alle Werte in $[2, 50 \cdot \text{BIG_NR}]$ sind möglich!

Wie löst man das Problem?

Quiz: Race-Condition

Sind kleinere Werte als `BIG_NR` für `bigSum` möglich?

Ja:

- ① Thread 1 holt sich den Wert von `bigSum`
- ② Thread 2 holt sich den Wert von `bigSum`
- ③ Thread 3 - 50 laufen durch
- ④ Thread 2 läuft bis 1 vorm Ende durch
- ⑤ Thread 1 erhöht Wert im Register von 0 auf 1 und schreibt 1
- ⑥ Thread 2 holt sich die 1 aus `bigSum`
- ⑦ Thread 1 läuft durch
- ⑧ Thread 2 erhöht Wert im Register von 1 auf 2 und schreibt 2

⇒ Alle Werte in $[2, 50 \cdot \text{BIG_NR}]$ sind möglich!

Wie löst man das Problem?

Quiz: Race-Condition

Sind kleinere Werte als `BIG_NR` für `bigSum` möglich?

Ja:

- ① Thread 1 holt sich den Wert von `bigSum`
- ② Thread 2 holt sich den Wert von `bigSum`
- ③ Thread 3 - 50 laufen durch
- ④ Thread 2 läuft bis 1 vorm Ende durch
- ⑤ Thread 1 erhöht Wert im Register von 0 auf 1 und schreibt 1
- ⑥ Thread 2 holt sich die 1 aus `bigSum`
- ⑦ Thread 1 läuft durch
- ⑧ Thread 2 erhöht Wert im Register von 1 auf 2 und schreibt 2

⇒ Alle Werte in $[2, 50 \cdot \text{BIG_NR}]$ sind möglich!

Wie löst man das Problem?

Quiz: Race-Condition

Sind kleinere Werte als `BIG_NR` für `bigSum` möglich?

Ja:

- ① Thread 1 holt sich den Wert von `bigSum`
- ② Thread 2 holt sich den Wert von `bigSum`
- ③ Thread 3 - 50 laufen durch
- ④ Thread 2 läuft bis 1 vorm Ende durch
- ⑤ Thread 1 erhöht Wert im Register von 0 auf 1 und schreibt 1
- ⑥ Thread 2 holt sich die 1 aus `bigSum`
- ⑦ Thread 1 läuft durch
- ⑧ Thread 2 erhöht Wert im Register von 1 auf 2 und schreibt 2

⇒ Alle Werte in $[2, 50 \cdot \text{BIG_NR}]$ sind möglich!

Wie löst man das Problem?

Wie löst man das Problem:?

Mit `AtomicLong` aus `java.util.concurrent`

Quiz

```
1 import java.util.ArrayList;
2 import java.util.List;
3 import java.util.concurrent.atomic.AtomicLong;
4
5 public class Main {
6     public static final int BIG_NR = 2000000;
7     public static AtomicLong bigSum = new AtomicLong(0);
8
9     public static void main(String[] args) {
10         List<Thread> threads = new ArrayList<Thread>();
11         for (int i = 0; i < 50; i++) {
12             Runnable task = new Sum(BIG_NR);
13             Thread worker = new Thread(task);
14             worker.start();
15             threads.add(worker);
16         }
17
18         int running = 0;
19         do {
20             running = 0;
21             for (Thread thread : threads) {
22                 if (thread.isAlive()) {
23                     running++;
24                 }
25             }
26             System.out.println("Remaining threads: " + running);
27         } while (running > 0);
28
29         System.out.println(Main.bigSum);
30     }
31 }
```

```
1 public class Sum implements Runnable {
2     private final int UpperEnd;
3
4     public Sum(int upperEnd) {
5         UpperEnd = upperEnd;
6     }
7
8     @Override
9     public void run() {
10         for (int i = 0; i < UpperEnd; i++) {
11             Main.bigSum.incrementAndGet();
12         }
13     }
14 }
```

- Herangehensweise
 - Genau lesen
 - Frühzeitig lösen, viel Testen (Am besten schon heute!)
 - Bei Unklarheiten frühzeitig fragen!
- Ausgabe
 - Genau die erwartete Ausgabe liefern
 - An `toString()` denken
- Trennung von Logik und Shell
 - Siehe [Lösung zur Aufgabe 5.5](#)
- Code
 - `equals()` , `compareTo()` nutzen wenn sinnvoll
 - Exceptions einbauen, sollte der Nutzer aber nie sehen!
 - Gute JavaDoc!

Bepunktung:

- Punkte für Funktionalität
- Punkte für Programmier-Stil
- Nicht unbedingt gleich gewichtet

Anleitung für Snake, Tetris, Sokuban, Breakout ... ist [hier](#).

- Runnable
- Java Concurrency (Multithreading) - Tutorial
- Java - Multithreading

1. 28.01.2013: Abschlussprüfunsvorbereitung
0. 04.02.2013: Abschlussprüfunsvorbereitung
 - 10.02.2013: Ende der Vorlesungszeit des WS 2012/2013 ([Quelle](#))

Vielen Dank für eure Aufmerksamkeit!

Days 1 - 10

Teach yourself variables, constants, arrays, strings, expressions, statements, functions,...



Days 11 - 21

Teach yourself program flow, pointers, references, classes, objects, inheritance, polymorphism,



Days 22 - 697

Do a lot of recreational programming. Have fun hacking but remember to learn from your mistakes.



Days 698 - 3648

Interact with other programmers. Work on programming projects together. Learn from them.



Days 3649 - 7781

Teach yourself advanced theoretical physics and formulate a consistent theory of quantum gravity.



Days 7782 - 14611

Teach yourself biochemistry, molecular biology, genetics,...



Day 14611

Use knowledge of biology to make an age-reversing potion.



Day 14611

Use knowledge of physics to build flux capacitor and go back in time to day 21.



Day 21

Replace younger self.



As far as I know, this is the easiest way to "Teach Yourself C++ in 21 Days".