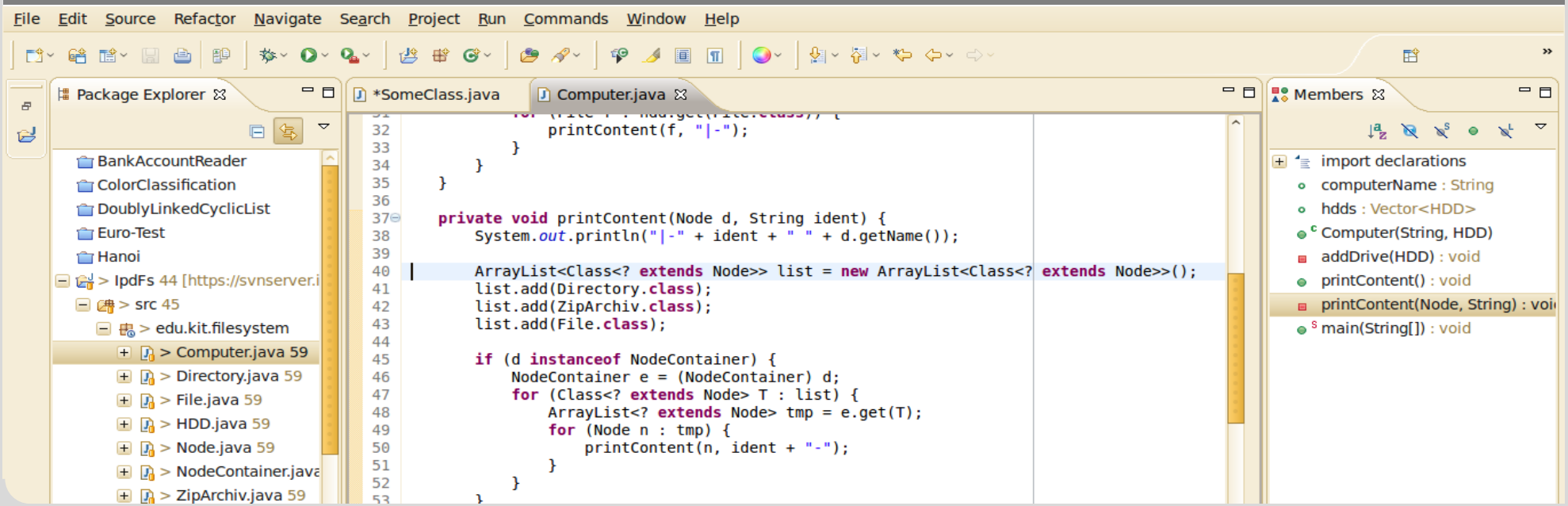
Karlsruher Institut für Technologie

Programmieren-Tutorium Nr. 10 bei Martin Thoma

Five-in-A-Row, Multithreading

Martin Thoma | 12. Februar 2013

FAKULTÄT FÜR INFORMATIK



Inhaltsverzeichnis

Quiz

```java
import java.util.ArrayList;
import java.util.List;

public class Main {
    public static final int BIG_NR = 2000000;
    public static long bigSum;

    public static void main(String[] args) {
        List<Thread> threads =
                new ArrayList<Thread>();
        for (int i = 0; i < 50; i++) {
            Runnable task = new Sum(BIG_NR);
            Thread worker = new Thread(task);
            worker.start();
            threads.add(worker);
        }

        int running = 0;
        do {
            running = 0;
            for (Thread thread : threads) {
                if (thread.isAlive()) {
                    running++;
                }
            }
            System.out.println("Remaining threads: " + running);
        } while (running > 0);

        System.out.println(Main.bigSum);

    }
}
```

```java
public class Sum implements Runnable {
    private final int UpperEnd;

    public Sum(int upperEnd) {
        UpperEnd = upperEnd;
    }

    @Override
    public void run() {
        for (int i = 0; i < UpperEnd; i++) {
            Main.bigSum++;
        }
    }
}
```

Quiz: Race-Condition

- `bigSum++;` ist nicht atomar:

  3 Operationen: Wert holen, Wert erhöhen, Wert schreiben

  $\Rightarrow$ 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

- `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

- `bigSum++;` ist nicht atomar:

  3 Operationen: Wert holen, Wert erhöhen, Wert schreiben

$\Rightarrow$ 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

- `bigSum++;` ist nicht atomar:

  3 Operationen: Wert holen, Wert erhöhen, Wert schreiben

$\Rightarrow$ 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

- `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

- `bigSum++;` ist nicht atomar:

  3 Operationen: Wert holen, Wert erhöhen, Wert schreiben

$\Rightarrow$ 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

- `bigSum++;` ist nicht atomar:

  3 Operationen: Wert holen, Wert erhöhen, Wert schreiben

$\Rightarrow$ 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

- `bigSum++;` ist nicht atomar:

  3 Operationen: Wert holen, Wert erhöhen, Wert schreiben

$\Rightarrow$ 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

- `bigSum++;` ist nicht atomar:

  3 Operationen: Wert holen, Wert erhöhen, Wert schreiben

$\Rightarrow$ 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

- `bigSum++;` ist nicht atomar:

  3 Operationen: Wert holen, Wert erhöhen, Wert schreiben

$\Rightarrow$ 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 hohlt sich die 1 aus `bigSum`
7. Thread 1 läuft durch
8. Thread 2 erhöht Wert im Register von 1 auf 2 und schreibt 2

$\Rightarrow$ Alle Werte in $[2, 50 \cdot BIG\_NR]$ 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 hohlt sich die 1 aus `bigSum`
7. Thread 1 läuft durch
8. Thread 2 erhöht Wert im Register von 1 auf 2 und schreibt 2

$\Rightarrow$ Alle Werte in $[2, 50 \cdot BIG\_NR]$ 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 hohlt sich die 1 aus `bigSum`
7. Thread 1 läuft durch
8. Thread 2 erhöht Wert im Register von 1 auf 2 und schreibt 2

$\Rightarrow$ Alle Werte in $[2, 50 \cdot BIG\_NR]$ 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 hohlt sich die 1 aus `bigSum`
7. Thread 1 läuft durch
8. Thread 2 erhöht Wert im Register von 1 auf 2 und schreibt 2

$\Rightarrow$ Alle Werte in $[2, 50 \cdot BIG\_NR]$ 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 hohlt sich die 1 aus `bigSum`
7. Thread 1 läuft durch
8. Thread 2 erhöht Wert im Register von 1 auf 2 und schreibt 2

$\Rightarrow$ Alle Werte in $[2, 50 \cdot BIG\_NR]$ 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 hohlt sich die 1 aus `bigSum`
7. Thread 1 läuft durch
8. Thread 2 erhöht Wert im Register von 1 auf 2 und schreibt 2

$\Rightarrow$ Alle Werte in $[2, 50 \cdot BIG\_NR]$ 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 hohlt sich die 1 aus `bigSum`
7. Thread 1 läuft durch
8. Thread 2 erhöht Wert im Register von 1 auf 2 und schreibt 2

$\Rightarrow$ Alle Werte in $[2, 50 \cdot BIG\_NR]$ 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 hohlt sich die 1 aus `bigSum`
7. Thread 1 läuft durch
8. Thread 2 erhöht Wert im Register von 1 auf 2 und schreibt 2

$\Rightarrow$ Alle Werte in $[2, 50 \cdot BIG\_NR]$ 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 hohlt sich die 1 aus `bigSum`
7. Thread 1 läuft durch
8. Thread 2 erhöht Wert im Register von 1 auf 2 und schreibt 2

$\Rightarrow$ Alle Werte in $[2, 50 \cdot BIG\_NR]$ 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 hohlt sich die 1 aus `bigSum`
7. Thread 1 läuft durch
8. Thread 2 erhöht Wert im Register von 1 auf 2 und schreibt 2

$\Rightarrow$ Alle Werte in $[2, 50 \cdot BIG\_NR]$ 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 hohlt sich die 1 aus `bigSum`
7. Thread 1 läuft durch
8. Thread 2 erhöht Wert im Register von 1 auf 2 und schreibt 2

$\Rightarrow$ Alle Werte in $[2, 50 \cdot BIG\_NR]$ sind möglich!

Wie löst man das Problem?

Quiz

Wie löst man das Problem:?
Mit AtomicLong aus java.util.concurrent

Quiz

```java
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.atomic.AtomicLong;

public class Main {
    public static final int BIG_NR = 2000000;
    public static AtomicLong bigSum = new AtomicLong();

    public static void main(String[] args) {
        List<Thread> threads = new ArrayList<Thread>();
        for (int i = 0; i < 50; i++) {
            Runnable task = new Sum(BIG_NR);
            Thread worker = new Thread(task);
            worker.start();
            threads.add(worker);
        }

        int running = 0;
        do {
            running = 0;
            for (Thread thread : threads) {
                if (thread.isAlive()) {
                    running++;
                }
            }
            System.out.println("Remaining threads: " + running);
        } while (running > 0);

        System.out.println(Main.bigSum);

    }
}
```

```java
public class Sum implements Runnable {
    private final int UpperEnd;

    public Sum(int upperEnd) {
        UpperEnd = upperEnd;
    }

    @Override
    public void run() {
        for (int i = 0; i < UpperEnd; i++) {
            Main.bigSum.incrementAndGet();
        }
    }
}
```

Abschlussaufgaben

- 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!

Abschlussaufgaben

Bepunktung:

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

Spiele

Anleitung für Snake, Tetris, Sokuban, Breakout . . . ist hier.

Multithreading

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

Kommende Tutorien

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!