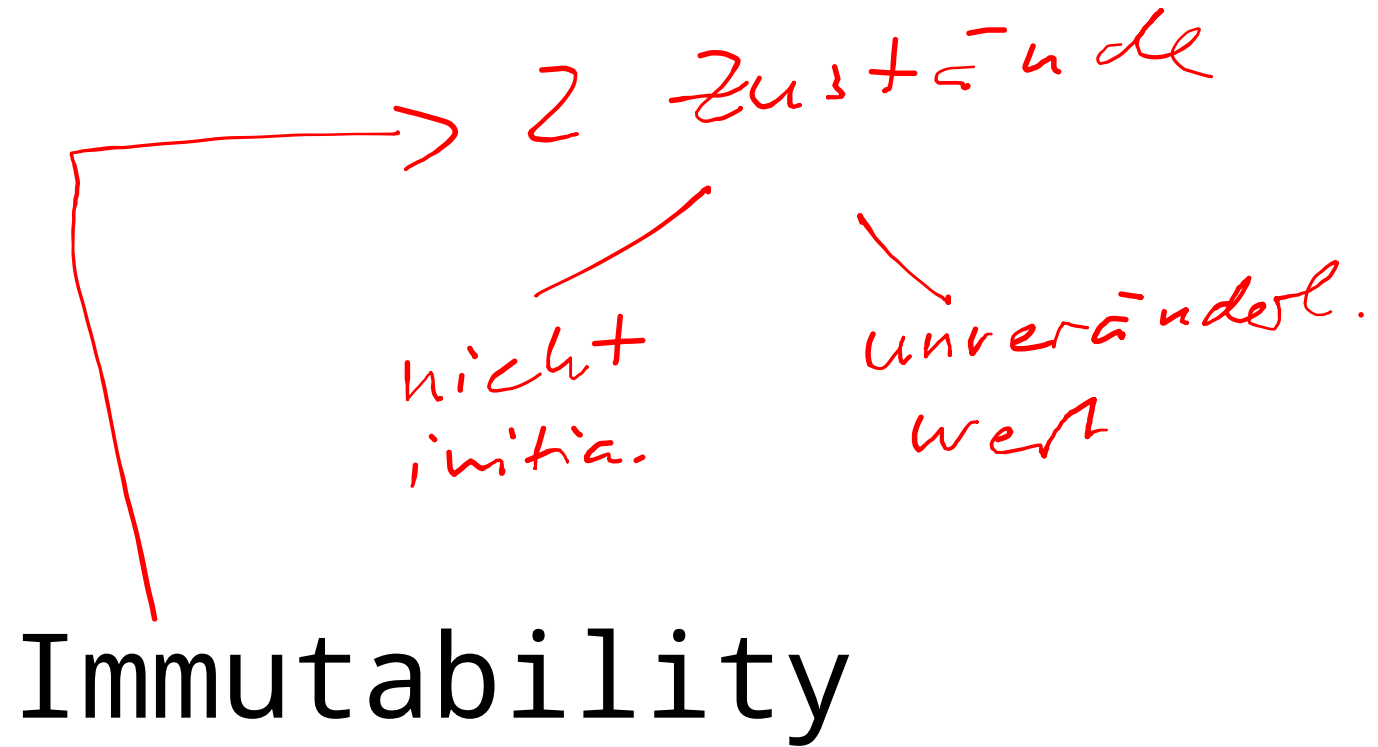


Mosbach – Java 2

VL 05

- todo



(Nicht verwechselt
mit Konstanten.)

- gut

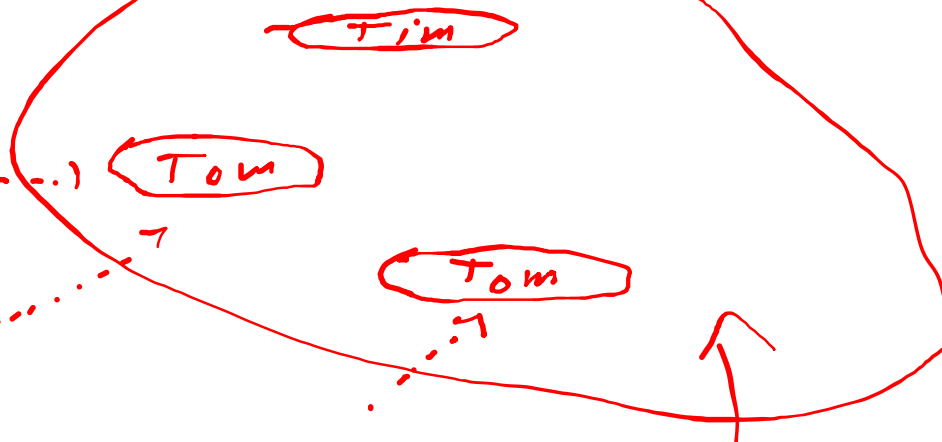
→ Performance
Korrektheit (formale) Beweise

- schlecht

wenn ich verändern muß

String Pool

String s1 = "Tim";
 String s2 = "Tom";
 String s3 = s1;
 String s4 = "Tom";
 String s5 = new String("Tom");



Pool aus
immut.
variable

Strings u.ä

Mischung aus prim. Typ und Klasse

String	StringBuffer	StringBuilder
immutable, aber Veränderung über neue Instanz	veränderbar	veränderbar
sehr schneller Zugriff sehr langsame Veränderung	bessere Performance bei Veränderungen	optimalste Performance bei Veränderungen
	Thread-safe	Nicht Thread-safe

- == vs equals
- wann bring == Gleichheit, wann nicht?

$s4 == s2$ true

$s4 == s5$ false

$s4.equals(s5)$
 $s4.equals(s2)$ } true

- Wettkampftest beim Erstellen eines Strings aus `int[]`-Array über `String`, `StringBuilder` und `StringBuffer`
- Einfach verschiedene Werte in einem `Random-int-array` miteinander verbinden und ausgeben.

Do you wanna have
it gift wrapped?

Wrapper

- **int-Wrapper**
 - Integer, BigInteger

-

- **Boxing/Unboxing**

- `int i = 8;`

- `Integer myNumber = new Integer(i);`

- `Integer myNumber = i;`

// bevorzugt

①

beiprintiven Daten -
+ Typen konvert. in
Objekte

z.B. ArrayList(*)

②

mehr Methoden

- Überlaufsicherheit
 - Weisen Sie nach, dass Addition, Multiplikation zu einem Überlauf führen können. (Aufgabenzettel)
 - Implementieren Sie daher die Durchschnittsberechnung von Zahlen über BigInteger und wandeln das Ergebnis zurück.
- Verwenden Sie dies in der Binären Suche.
 - Testen Sie, dass die binäre Suche immer noch funktioniert.
- Verändern Sie Binäre Suche zur Berechnung des Mittelwertes nach Erkenntnis des Bugs.
- Implementieren Sie eine Methode ggT in der Klasse Bruch über BigInteger-Operationen.

- Immutability
- Wrapper
- Boxing/Unboxing
- Erhöhe die Stelligkeit um 2
- = vs == vs equals
- String, StringBuffer, StringBuilder
- int => Integer, BigInteger

- slides ansehen