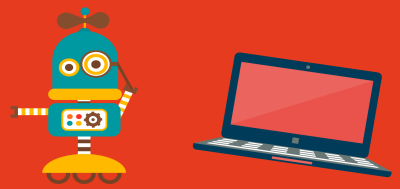
htl3 rennweg

Objekte vergleichen
Objekte sortieren

DI Reinhold Buchinger

Höhere Abteilung für Mechatronik
Höhere Abteilung für Informationstechnologie
Fachschule für Informationstechnik



Ziele

» Java Objekte vergleichen und sortieren können

Wann sind zwei Objekte gleich?

Vergleichen von Objekten

- Beispiel: Klasse, die eine Person mit Vornamen, Nachnamen und Sozialversicherungsnummer abbildet.
- Wenn ich mehrere Objekte vergleiche, wann handelt es sich um die gleiche Person?
  - Gleicher Nachname?
  - Gleicher Vorname und Nachname?
  - Gleiche Sozialversicherungsnummer?
- Das Programm kann das nicht automatisch wissen. Wir als Programmier*innen müssen es festlegen.
- Anwendungsfall: Wie stellt ein Set fest ob ein Objekt bereits Teil des Sets ist (keine Duplikate!)?

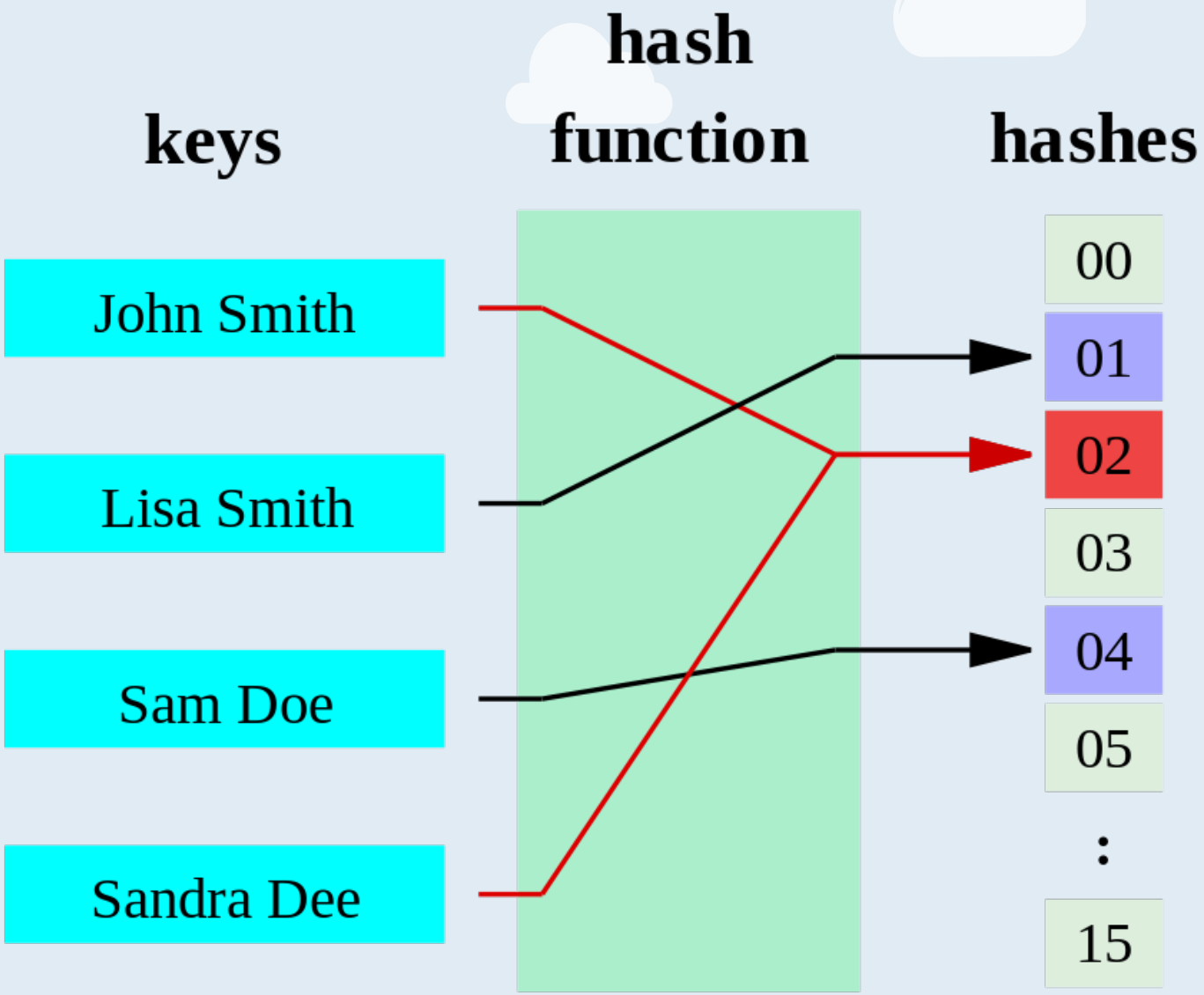
hashCode() und equals()

htl3 rennweg
IT & MECHATRONIK

- Die beiden Methoden hashCode() und equals() sind in der Klasse Object definiert, von der jede Klasse in Java automatisch erbt.
- Beide Methoden besitzen eine Default-Implementierung, die in einer Klassen überschreiben werden kann und auch sollte.
- IntelliJ unterstützt bei der automatischen Erstellung von passenden Implementierungen (Rechts-Klick -> Generate)

Einschub Hashfunktion

- Eine Hashfunktion bildet eine große Eingabemenge auf eine kleinere Zielmenge (=die Hashwerte) ab.
- Hashwerte haben meist eine fixe Länge.
- Hashwerte bestehen meist auf natürlichen Zahlen.
- Kollision = unterschiedlichen Eingabedaten erhalten denselben Hashwert
  - Gute Hashfunktion => möglichst wenig Kollisionen für die erwarteten Eingaben
- Hashwerte werden angewendet für Prüfsummen, in der Kryptologie (Nachrichten signieren, Passwörter speichern), Daten effizient in großen Datenmengen zu suchen (mittels Hashtabellen)


keys
hash function
hashes
John Smith
Lisa Smith
Sam Doe
Sandra Dee
00
01
02
03
04
05
:
15

hashCode()

- Methode der Klasse Object
  - besitzt jede Klasse in Java automatisch
  - kann (und soll) überschrieben werden
- liefert für das Objekt eine möglichst eindeutige Integerzahl (=den Hashwert), die sowohl positiv als auch negativ sein kann.
- Nötig für Datenstrukturen mit Hashing-Algorithmen
  - Eine Zahl zu vergleichen ist einfacher als ganze Objekte
- Default-Implementierung: interne Speicheradresse des Objekts

equals()

- Ebenfalls Methode der Klasse Object
- Definiert die **inhaltliche** Gleichheit von Objekten
  - kennen den Unterschied bereits von Strings
  - s1 == s2 => testet **Identität** (das selbe Objekt)
  - s1.equals(s2) => testet inhaltliche Gleichheit
- Standardimplementierung testet ob die Objekte gleich sind
  - Ohne Überschreiben liefert == und .equals() das selbe Ergebnis
- Wenn equals() überschrieben wird muss auch hashCode() überschrieben werden
  - Wenn equals() true liefert, müssen auch die Hashwerte gleich sein

Sortierung

Sortierung

- Genauso wie Java nicht wissen kann wann zwei Objekte inhaltlich gleich sind, kann es nicht wissen wie wir Objekte sortieren wollen.
- Implementieren dafür eines der folgendes Interfaces
  - *Comparator (vergleicht zwei Objekte)*
  - *Comparable (vergleicht anderes Objekt mit sich selbst)*
- Viele Klassen der Java API implementieren Comparable (z.B. String, Integer,...)
  - -> Objekte dieser Klassen können sortiert werden.
- Beispiele Anwendung
  - Treeset – ein sortiertes Set
  - Collections.sort() und Arrays.sort()
- **Siehe Codebeispiele!**

Comparable vs Comparator

htl3 rennweg
IT & MECHATRONIK

Comparable<T>

int compareTo(T o);

Vergleicht this mit dem Parameter o

Comparator<T>

int compare(T o1, T o2);

Vergleicht o1 mit o2

		Rückgabewert:	
< 0	this	ist kleiner	als das übergebene Obj. o
==0	o1	ist gleich	o2
>0		ist größer	

Hinweis: Viele Klassen (Integer, String,...) implementieren das Comparable Interface. Du kannst das bei der Implementierung für deine eigenen Klassen nutzen!

Fragen?
Anregungen?
Bemerkungen?