Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

Programmiertechnik II
Übung 6
07.07.2016

Maximilian Jenders

Rückblick Hausaufgabe 5

Aufgabe 1 – Verlinkte Listen

- Generell noch Probleme mit Java Generics
  - Definieren von Knoten Klasse mit generischem Parameter, dann (teilweise) ohne (Resultat in der Praxis: *Node<Object>*)

- Für *DoublyLinkedSequence* keinen tail-Pointer benutzt (Dadurch O(n) statt O(1) für einige Methoden)

- Keine Definition einer Helpermethode wie *getNode(i)*, wodurch starke Duplikation zwischen *insert(i, x), get(i), set(i, x),* und *remove(i)* entsteht

Aufgabe 1 – Verlinkte Listen / ArrayListen

- Neudefinieren des Typs in inner class:
  - *class Sequence<T> { class Node<T> { } }*
  - T wird überschrieben
  - Lösung: static class Node<T> funktioniert

- *FixedArraySequence* häufig kein Ringbuffer!
- Komplexität nicht/falsch angegeben
- Edgecases bei Exception-handling nicht bedacht

- Schlechte Performance: Vergößern/Verkleinern des Arrays bei insert/remove

Aufgabe 1 – Verlinkte Listen

- Collection erstellen mit Array:
  - Wenn neues Objekt eingefügt wird und kein Platz im Array ist:
  - NICHT neues Array mit Größe +=1 (imperformant)
  - Mit „Puffer“ erhöhen, += log2, *= 2, …
  - „Amortisiertes Verhalten“ (https://de.wikipedia.org/wiki/Amortisierte_Laufzeitanalyse)

- Ähnlich auch bei HashMaps: Wann re-hash, neue Buckets?
  - Füllstatus, Bucketauslastung, …

Aufgabe 3 – Bäume

- Suchbaum bauen / verletzte Bediungen finden: sehr gut
- Balancieren durch Wurzeländerung: Oft falsch oder unvollständig (O und G möglich)
- Unmögliche Suchsequenzen nicht vollständig bearbeitet

- Praktischer Teil: *putIterative()* oft fehlerhaft
  - Auf falscher Seite eingefügt
  - Subtree nicht umgehängt
  - N nicht (richtig) gesetzt
  - Off-by-1-errors
  - …

Nachtrag Reguläre Ausdrücke

HPI Hasso Plattner Institut

THE VALIDATION REGEX

by Remy Porter in Representative Line on 2016-06-30

Regular expressions are a powerful tool for validating inputs, but what if your input is itself a regular expression? Is there a regular expression that can validate regular expressions?

Well, yes, if your regular expression engine supports recursion:

```
/^((?:(?:[^?+*{}()[\]\\|]+|\\.|\[(?:\^?\\.|\^[^\\]|[^\\^])(?:[^\]\\]+|\\.)*\]|\((?:\?[:=!]|\?<[=!]|\?>)?(?1)??\)|\(\?(?:R|[+-]?\d+)\))(?:(?:[?+*]|\{\d+(?:,\d*)?\})[?+]?)?|\|)*)$/.
```

- http://thedailywtf.com/articles/the-validation-regex

Hashing

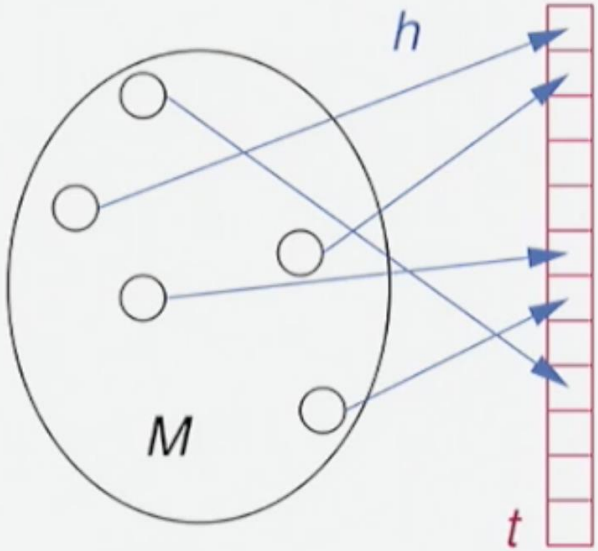
Hashfunktionen

- Gleiche Eingabe, gleiche Ausgabe
- Ziel: Ähnliche Eingabe, sehr verschiedene Ausgabe
- Ziel: Ausgabe möglichst gleichverteilt
- Kollisionen – unvermeidbar?
  - Hashfunktionsausgabe i.d.R. auf bestimme Länge definiert

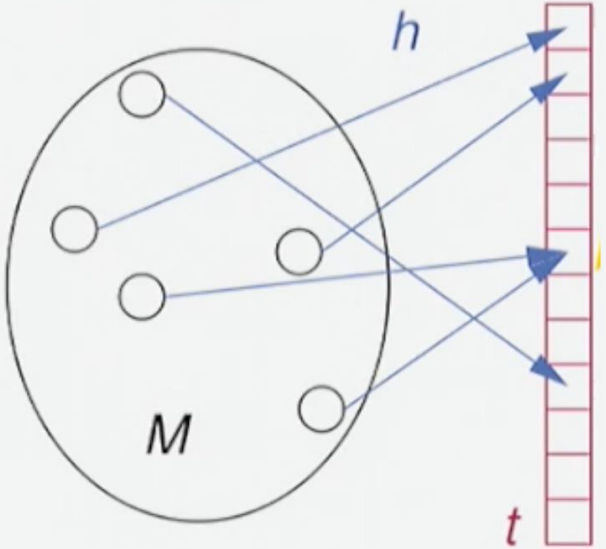
- Zeitverbrauch: Sollte eine Hashfunktion schnell sein?
  - Hash vs Checksum

Hashfunktionen

Do you think THIS is beautiful???



Or this?



REAL HASH FUNCTIONS HAVE COLLISIONS!

Stack Traces

Stack Traces

- Zeigt Aufruf-Hierarchie an
- Typischerweise geworfen bei Fehlern zur besseren Analyse / Behebung
- Auch manuell auslösbar: Thread.currentThread().getStackTrace();
- Wie lesen?
  - Von unten nach oben / von oben nach unten?
  - Ganz oben: Exception
  - Darunter: In welcher Zeile aufgetreten
  - Darunter: Von welcher Methode wurde die Methode dieser Methode gerufen?
  - Etc
- Gutes Beispiel: https://stackoverflow.com/questions/3988788/what-is-a-stack-trace-and-how-can-i-use-it-to-debug-my-application-errors

Lambda-Funktionen

Lambdas

- Seit Java 1.8 (also noch recht neu)
- Auch: Anonyme Funktionen

- Gut benutzbar in Verbindung mit Streams (auch ab 1.8)

- → Demo

Try-With-Resources

Try-With-Resources

- Ab 1.7

```
BufferedReader br = new BufferedReader(new FileReader("file.txt"));
try {
    StringBuilder sb = new StringBuilder();
    String line = br.readLine();

    while (line != null) {
        sb.append(line);
        sb.append(System.lineSeparator());
        line = br.readLine();
    }
    String everything = sb.toString();
} finally {
    br.close();
}
```

Try-With-Resources


```
try(BufferedReader br = new BufferedReader(new FileReader("file.txt"))) {
    StringBuilder sb = new StringBuilder();
    String line = br.readLine();

    while (line != null) {
        sb.append(line);
        sb.append(System.lineSeparator());
        line = br.readLine();
    }
    String everything = sb.toString();
}
```

Try-With-Resources

- Dateien einlesen in Java 1.8 übrigens noch einfacher:

```
//read file into stream, try-with-resources
try (Stream<String> stream = Files.lines(Paths.get(fileName))) {

    stream.forEach(System.out::println);

} catch (IOException e) {
    e.printStackTrace();
}
```

```
try (Stream<String> stream = Files.lines(Paths.get(fileName))) {

    //1. filter line 3
    //2. convert all content to upper case
    //3. convert it into a List
    list = stream
            .filter(line -> !line.startsWith("line3"))
            .map(String::toUpperCase)
            .collect(Collectors.toList());

} catch (IOException e) {
    e.printStackTrace();
}

list.forEach(System.out::println);
```

Hasso Plattner Institut

Feedback

Feedback

- Wichtige Lektion für das Studium: Fragt!
  - Ihr seid nicht der/die einzige, der etwas nicht verstanden hat
  - Wichtig auch für Lehrenden: Zu schnell / zu langsam? Verständlich?
  - Ohne Feedback keine Verbesserungen!

- openHPI-Survey: https://blog.openhpi.de/survey/index.php/survey/index/sid/668771/newtest/Y/lang/de

- Jetzt zu spät, daher nur Feedback über EvaP möglich
  - Bitte evaluiert ☺

Organisatorisches

- Klausur 02.08.2016 (Dienstag)
- Ich bin im Urlaub vom 15.07.2016 – 24.07.2016, **lese keine Emails**
- Danach / Davor beantworte ich gerne noch Fragen – aber bitte **frühzeitig** lernen / überlegen, nicht erst am Tag / Wochenende vor der Klausur

- Feedback zu HA 6 daher am 25.07.2016 (MO)