Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

Programmiertechnik II
Übung 3
26.05.2016

Maximilian Jenders

Hausaufgabe 2
Rückblick

Generelles

- Korrektur noch nicht komplett fertig, wahrscheinlich heute abend / morgen

- Progammierer sind meistens faul….
  - Nutzt Funktionalität, die es schon gibt
  - String.compareTo()
  - Collections.sort()

Exceptions

- ```
  try {
          foo();
  }
  catch (Exception e) {
          e.printStackTrace();
  } catch (FileNotFoundException e2) {
  e2.printStackTrace();
  }
  ```

- ```
  try {
          foo();
  }
  catch (FileNotFoundException e) {
          e.printStackTrace();
  } catch (Exception e2) {
  e2.printStackTrace();
  }
  ```

- Gibt es einen Unterschied?

Aufgabe 1

- Umgang mit Exceptions: NullPointerException
- Unschön: try {…} catch (NullPointerException e)
- Sehr schlecht: try {…} catch (Exception e)
  - Warum?
- Schön: if (var == null) return;
- Wer hat mit Lamda experimentiert?

Aufgabe 1

- 1d) wenig oder falsch bearbeitet
  - Frage bzgl „Stabilität“ -> Welche Attribute werden bei der Sortierung verwendet?

sort

```
public static <T extends Comparable<? super T>> void sort(List<T> list)
```

Sorts the specified list into ascending order, according to the *natural ordering* of its elements. All e `ClassCastException` for any elements `e1` and `e2` in the list).

This sort is guaranteed to be *stable*: equal elements will not be reordered as a result of the sort.

- Welche Punkte sollte man generell mitnehmen?
  - Validate your input
  - Objektgleichheit != Wertegleichheit

Aufgabe 2 / 3

- Laufzeitbeobachtung kaum gemacht (auch A3) – warum?
  - Probleme?

- Bubble/Quicksort/Countingsort meist kein Problem
- Warum kein CountingSort für Collections.sort() ?
  - Nimmt Objektgleichheit an bei gleichem Sortierschlüssel
  - Höherer Speicherbedarf (viel bei großen Intervallen)

- Vergleiche: Guckt euch noch einmal O()-Notationen an
  - Auch in HA 3

Zusatzaufgabe

- Interessante Ansätze
  - Raster rotieren, um in andere Richtung zu suchen
  - All Zeilen/Spalten in langen String, dann String-Matching

- Sonstige Optimierungen:
  - StringBuffer.reverse()
  - Nur Zeilen/Spalten mit Vokal
  - Abbruch nach 1 Wort pro Zeile / Spaölte
  - Gefundenes Wort wird nicht noch woanders gesucht

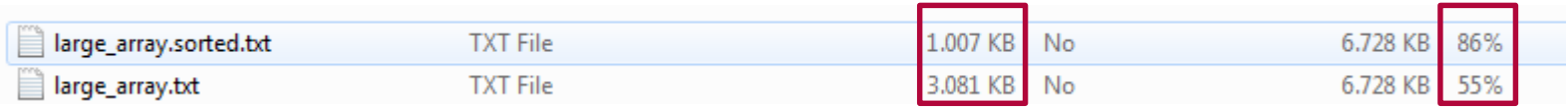
- Teils keine Kommentare bei 200-Zeilen-Methoden ☹

Zusatzaufgabe

- Generell:
  - Daumenregel: Methode max. 50 Zeilen
  - Einrückungen ab 5 Tiefenebenen sind doppelplusungut, Indikator für schlechte Struktur
  - Kommentare, Kommentare, Kommentare!
  - Macht Code lesbarer und wartbar

Übrigens…

- Large_arrays.zip: Dateigröße / Kompressionsgrad nach zippen



large_array.sorted.txt	TXT File	1.007 KB	No	6.728 KB	86%
large_array.txt	TXT File	3.081 KB	No	6.728 KB	55%

- Erklärансätze?
  - Komprimierung: Dictionary vs Delta-Encoding

Übrigens…

- Abstrakte Klassen, Interfaces…
  - Beide geben Methodensignaturen vor…
  - Was genau ist noch einmal der Unterschied?

Hasso Plattner Institut

Hausaufgabe 3

HA 3

- Wer hat sich das Übungsblatt schon angeguckt?
- Wer hat schon angefangen?
- Diesmal weniger Tests von unserer Seite!

TESTING

Tests

- A tester walks into a bar and orders 1 beer
  - And orders 5 beers
  - And orders 9999 beers
  - And orders 0 beers
  - And orders -1 beers
  - And orders foobar beers
  - And orders null beers
  - And orders NaN beers
  - And orders INT.MAX_INT beers
  - And orders  beers
  - And orders '); DROP TABLE BEERS; -- beers

Tests

- Wie stelle ich Korrektheit meines Programms sicher?

- Tests sollten
  - Standardfälle abdecken
  - Illegale Eingaben abdecken
  - Grenzfälle abdecken

Tests

- Welche Metriken gibt es noch?

- Testabdeckung
  - Zeilenübderdeckung: Wie viele Zeilen meines Programms wurden durch Tests überprüft?
  - Anweisungsüberdeckung: $C_0 = \frac{\text{Anzahl der überdeckten Anweisungen}}{\text{Gesamtanzahl der Anweisungen}}$
  - Zweigüberdeckung: $C_{\text{primitiv}} = \frac{\text{Anzahl der ausgefuehrten primitiven Zweige}}{\text{Anzahl aller primitiven Zweige}}$
  - Pfadüberdeckung

- Immer schwerer zu erreichen, aber immer aussagekräftiger!

Hasso Plattner Institut

Laufzeitmessungen

Laufzeitmessungen

- Vergleich zweier Algorithmen, start/stopzeit: System.currentTimeMillis()
- Ausführung auf PC – nebenbei noch:
  - Betriebssystem
  - IDE
  - Brower
  - Messenger

- Kann alles Messung beeinflussen

Laufzeitmessungen

- Was tun?
  - Überflüssige Programme beenden
  - Aber: Immer noch OS, Hardware-Interrupts, …
- Beispiel: Sort-Vergleich auf kleiner Eingabe
  - Laufzeit: 4 ms vs 6 ms
  - Aussagekräftig?
- For (int i = 0; i <= 1000; i++) {sortA(); sortB();}
  - Algorithmus mehrfach ausführen → externe Einflüsse werden unwichtiger

Laufzeitmessungen

- Aber: Messen beeinflusst das Experiment
  - Kostet Zeit & Platz
  - Berücksichtigen

- Profiling-Apps: Welche Methode wird am häufigsten gerufen / verbraucht am meisten Zeit
  - Haben Nutzen, sollte man aber nicht blind vertrauen

</Vortrag>

- Fragen?
- Schon erste Probleme?
- Ansonsten: Jetzt Poolraum.