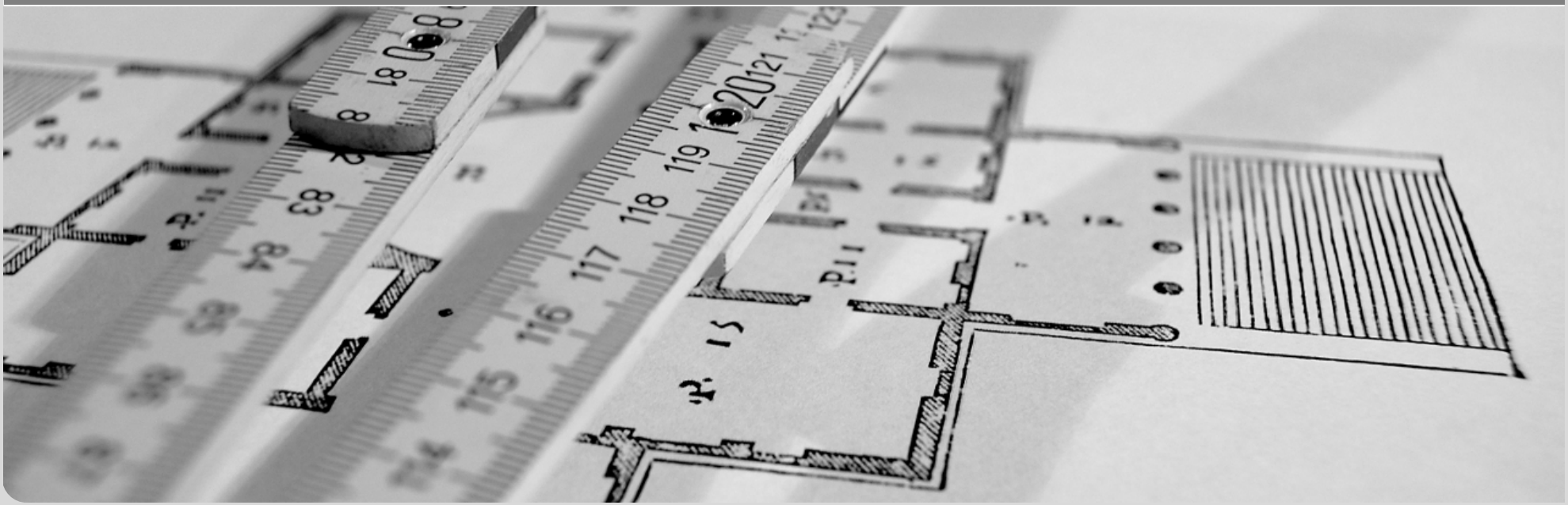
Karlsruher Institut für Technologie

SDQ Software Design and Quality

Programmieren
10. Tutorium

Robin Rüde | 26. Januar 2015



Gliederung

1

Organisatorisches

Organisatorisches

- Anmeldung zum Übungsschein: 19.01 - 30.01.
- Anmeldung Abschlussaufgaben: 02.02. - 13.02.
- Anmeldung unter `campus.studium.kit.edu`
- Nicht vergessen. Wenn ihr zu spät seid, habt ihr Pech gehabt

Blatt 5

- Polymorphie nutzen, zum Beispiel bei der Traversierung
- Traversierung und Suche hatten nichts miteinander zu tun
- KEIN System.out.println
- Fehler mit Ërror,äusgeben. **NICHT** anders.
- Stabilitätstests. Testet eure Programme mit leeren Dateien, falschen Eingabeparametern

2

Fehler

Failure vs. Fault vs. Error

- Failure -> System verhält sich nicht gemäß der Spezifikation
- auftreten zur Laufzeit
- Fault -> tatsächliche oder vermutetet Ursache des Versagens
- Fault == Ursache -> Code ist falsch
- Failure == Wirkung -> Absturz, falsche Ausgabe o. ä.
- Error: Unterschied zwischen tatsächlichem und erwünschten Programmzustand
- kann zur Laufzeit repariert werden (Redundanz)

3

Assertions

assert-statement

- Syntax: `assert <boolean>;`
- Verwendung: Prüfen von Invarianten, Vor- und Nachbedingungen
- Wird standardmäßig ignoriert
- Aktivieren mit `java ... -ea`
- Muss true zurückgeben, sonst: Programmabruch
- nicht verwenden zum Prüfen von Parametern (stattdessen: `IllegalArgumentException`)

assert-statement

Beispiel:

```
if (i % 3 == 0) {
...
} else if (i % 3 == 1) {
    ...
} else {
    assert i % 3 == 2;
...
}
```

In der realen Welt: selten verwendet

Testen mit JUnit 4

- Ziel: Robuste Programmierung
- Prüfen der Korrektheit einzelner Programmteile durch Test-Funtionen
- Testfunktionen verwenden beispielhafte Eingaben mit bekannten Ausgaben
- $\Rightarrow$ Möglichst hohe Testüberdeckung, prüfen z.B. mit EclEmma

(Eclipse-Beispiel)

4

Maps

Map

- Index: beliebige Klassen und Objekte
- `Map<String, String>` speichert Strings als Index (Schlüssel) und Strings als Werte
- Map ist Interface für die folgenden beiden Implementierungen
- Anwendung zum Beispiel bei Wörterbüchern oder zum effizienten finden von Objekten

HashMap

- Index/Schlüsselklasse muss korrekte Implementierung von hashCode() und equals() besitzen
- Intern: Indizes mittles hashCode in einem Array platziert
- erwartet O(1)-Zugriff -> sehr schnell
- Verlangsamung wenn array-Größe geändert werden muss
- Verlangsamung bei Hashkonflikten

TreeMap

- Implementierung eines Binärbaums
- Index/Schlüsselklasse muss korrekte compareTo-Methode implementieren.
- Jeder Knoten hat 2 Kinder -> ein größeres und ein kleineres
- langsamer als HashMap (O(log n)), aber gut wenn die Map immer sortiert sein soll

5

Suche

Suche

- Methoden in einem Array ein bestimmtes Element zu finden
- Lineare Suche
  - Array von links durchgehen, jedes Element vergleichen
  - Laufzeit: O(n)

Suche

- Methoden in einem Array ein bestimmtes Element zu finden
- Lineare Suche
  - Array von links durchgehen, jedes Element vergleichen
  - Laufzeit: O(n)
- Binäre Suche
  1. Gesuchtes Element mit mittlerem Element vergleichen

  2.a Wenn gleich, Element zurückgeben

  2.b Wenn kleiner, rekursiv auf der linken Hälfte weitersuchen

  2.c Wenn größer, rekursiv auf der rechten Hälfte weitersuchen
  - Laufzeit: O(log(n))
  - Array muss sortiert sein!

6

Sortieren

Wichtigste Sortieralgorithmen

- Insertionsort - für sehr kleine Arrays (≅Bubblesort/Selectionsort)
- Mergesort
- Quicksort
- Heapsort
- Siehe Vorlesung / Internet

7

Aufgaben

Aufgabe: Binäre Suche

Implementiere eine Funktion, die mit rekursiver binärer Suche ein Element in einem sorierten Array findet

Ende

Vielen Dank für eure Aufmerksamkeit!
Fragen?