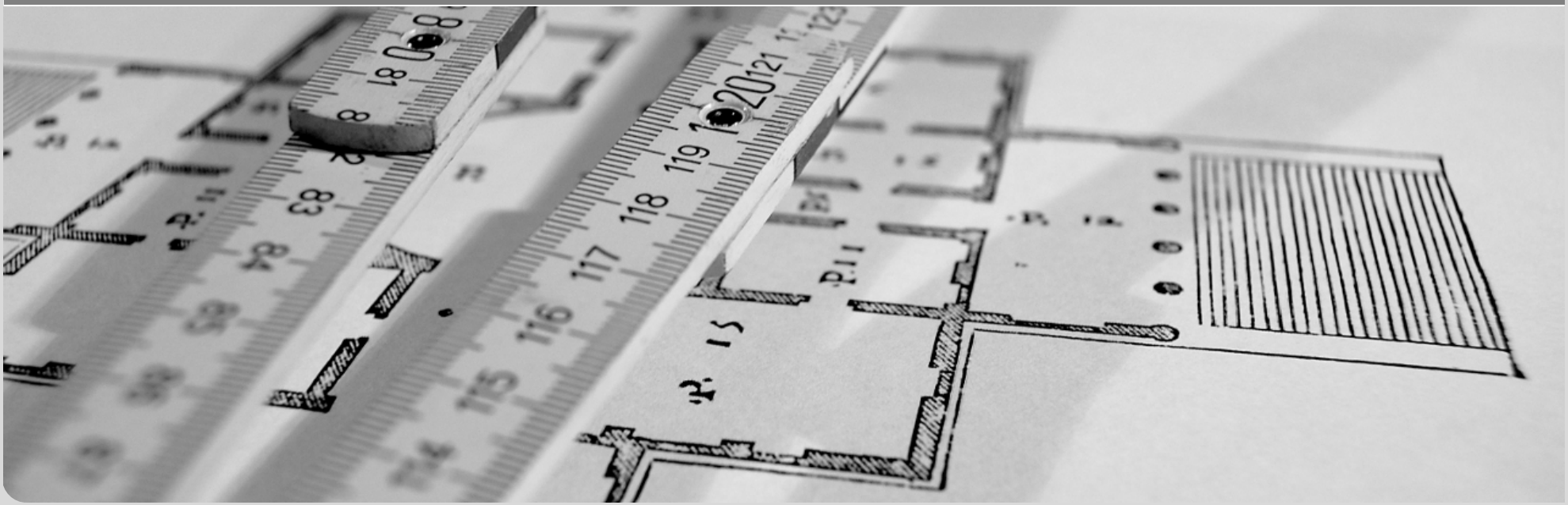
Karlsruher Institut für Technologie

Software Design and Quality

Programmieren
06. Tutorium

Robin Rüde | 8. Dezember 2014



Gliederung

1

Organisatorisches

Weihnachten

- Wer ist am Montag, 22.12. noch da?

2

Diverses

static vs. final

- static: Klassenattribut/Klassenmethode $\rightarrow$ gehört zur **Klasse**
- final: Konstanten $\rightarrow$ gehören zum Objekt und können nach Initialisierung nicht mehr geändert werden
- static final: Konstante Klassenattribute $\rightarrow$ gehören zur Klasse und sind nicht veränderbar

3

Benutzung von Generics

Generics

- Parametrisierte Klassen
- Klassen mit anderen Klassen als Parameter
- An entsprechendes Generic angepasste Rückgabe/Eingabe

Beispiel

```
// Creates a list of Integer.
LinkedList<Integer> list = new LinkedList<Integer>();
// Creates a list of String
LinkedList<String> list = new LinkedList<String>();
// Methods return different types e.g.
// list.get(int index) returns an Integer / String
```

Generics

Generics funktionieren nur mit Objekten, nicht mit primitiven Datentypen.

Deswegen:

int	$\rightarrow$	Integer
float	$\rightarrow$	Float
double	$\rightarrow$	Double
boolean	$\rightarrow$	Boolean

z.B. für eine Array-Liste aus ints:

```
ArrayList<Integer> list = new ArrayList<Integer>();
list.add(5);
list.add(10);
System.out.println(list.get(0)); // 5
```

4

java.util

Queue / LinkedList<E>

- doppelt verkettete Listen
- Nutzung: listen ohne random access.
- Zugriff auf Elemente O(n)
- Methoden:
  - add(E item) - Adds item at end of list
  - int size() - Größe der Liste
  - E get(int index) - Get item at index

ArrayList<E>

- Ähnlich `NaturalNumberTuple` von Blatt 3
- wachsende Arrays
- Nutzung: Listen mit Random access
- Zugriff auf Elemente O(1)
- Methoden:
  - add(E item) - Adds item at end of list
  - int size() - Größe der Liste
  - E get(int index) - Get item at index

Stack<E>

- Stapel
- LIFO - Last in - first out
  - empty() - if the stack is empty
  - E peek() - gets element on top
  - E pop() - pops element on top and returns int

PriorityQueue<E>

- sortierte Liste
- nach remove / add wieder sortiert
- Methoden:
  - add(E item) - Adds an item
  - E peek() - returns top element (Standard: kleinstes ist oben)
  - E poll() - Removes top of the Queue
  - int size() - Returns size of the Queue

5

Javadoc

Javadoc

- Dokumentation von Klassen/Methoden im Code
- Beschreibt, was eine Klasse/Methode tut und welche Argumente erwartet werden
- Form: Ähnlich zu mehrzeiligen Kommentaren
- Beginn: /**
- in jeder Zeile: *
- Ende: */
- verschiedene Tags, beginnend mit @
- z. Bsp @param, @return, @version, ...
- eclipse generiert javadoc für Methoden/Klassen autmatisch, wenn man /** und [Enter] eingibt (kann auch angepasst werden ;-)).

Beispiel

```java
/**
 * Encrypts the given String.
 *
 * @param str
 *      string to be encrypted
 * @return
 *      encrypted form of s
 */
public String encrypt(String s) {
    // Code
}
```

Tag: @author und @version

@author

Wird verwendet um den Autor einer Klasse oder Methode anzugeben.

Verwendung: `@author [Name]`

Beispiel: `@author Tino Fuhrmann`

Tag: @author und @version

@author

Wird verwendet um den Autor einer Klasse oder Methode anzugeben.

Verwendung: `@author [Name]`

Beispiel: `@author Tino Fuhrmann`

@version

Angabe der Version der Methode/Klassn.

Verwendung: `@version [Versionsnummer]`

Beispiel: `@version 1.0`

Tag: @param und @return

@param

Erklärung der Parameter einer Funktion.

Zum Beispiel möglicher Wertebereich / Bedeutung etc.

Verwendung: `@param [ParameterName] [Beschreibung]`

Beispiel: `@param str   String to be encrypted`

Tag: @param und @return

@param

Erklärung der Parameter einer Funktion.
Zum Beispiel möglicher Wertebereich / Bedeutung etc.
Verwendung: `@param [ParameterName] [Beschreibung]`
Beispiel: `@param str  String to be encrypted`

@return

Erklärt, was eine Methode zurückgibt.
Verwendung: `@return [Beschreibung]`
Beispiel: `@return encrypted form of s`

Beispiel Teil 1

```java
/**
 * Models a circle and can calculate its area.
 *
 * @author Tino Fuhrmann
 * @version 1.0
 */
public class Circle {

        private double radius;

        /**
         * Initializes the circle with its radius.
         *
         * @param radius
         *      radius of the circle
         */
        public Circle(double radius) {
                this.radius = radius;
        }
```

Beispiel Teil 2

```
        /**
         * Calculates and returns the area of the circle.
         *
         * @return area of the circle
         */
        public double getArea() {
                return this.radius * radius * Math.PI;
        }

        /**
         * Sets the radius.
         * @param radius new radius
         */
        public void setRadius(int radius) {
                this.radius = radius;
        }
}
```

6

Eclipse

Eclipse: Generate getters and setters, Javadoc-Generierung

- Eclipse kann getter und setter generieren
- Code Templates können angepasst werden
- zum Beispiel: automatische Generierung von javadoc für getter und setter
- Templates anpassen: Preferences -> Java -> Code Templates -> Getters / Setters -> Edit.
- Getter und Setter generieren: Source -> generate getters and setters -> create method comments
- /** [Enter] zur Erzeugung eines Skelettes für javadoc (direkt über Methoden/Klassen)

7

Aufgaben

Eigene Liste

Baue eine eigene Listenklasse, die folgendes kann:

- Speicherung von Elementen vom Typ int
- Methode: public void append(int element) (fügt an das Ende der Liste ein Element mit element als Inhalt an)
- Methode: public int get(int index) (gibt den Inhalt des Elementes an Stell index zurück)
- Optional: einfügen, löschen, etc

Semifibonacci

Schreibe eine Funktion, die den folgenden Algorithmus für n=1..100 in ein Array (nicht-rekursiv) berechnet

$$f(n) = \begin{cases} 1 & \text{falls } n \leq 1 \\ f(n/2) & \text{falls n gerade} \\ f(n-1) + f(n-2) & \text{falls n ungerade} \end{cases}$$

TODO-List

Input: `java TodoList <Pfad zur Datei> <Personenname>`

list.txt

```
Hans;Wäsche waschen
Peter;Klo putzen
Hans;Abspülen
```

Ausgabe

```
$ java TodoList list.txt Hans

Aufgaben von Hans
- Wäsche waschen
- Abspülen
```

Korrekte Klammerausdrücke

- Eingabe: Ein Klammerausdruck wie z.B. `(()(()))`
- Funktion prüft, ob der Klammerausdruck valide ist

Beispiel:

```
()      → true
(())    → true
()()    → true
(       → false
())      → false
```

Zusatz: Akzeptiere mehrere Klammerarten

```
([])    → true
([][])  → true
([{}])  → true
([)]    → false
```

Hinweis: Stacks

Ende

Fragen?

Ende

Fragen?

Vielen Dank für eure Aufmerksamkeit!