Dokumentation mit Javadoc

Dokumentation

Wenn man nicht sagt, was ein Programm machen soll, ist die Wahrscheinlichkeit gering, dass es das macht.

Dokumentation soll helfen,

- Schnittstellen zu verstehen
- Entwurfsideen zu erklären
- implizite Annahmen (z.B. Klasseninvarianten) auszudrücken

Nicht sinnvoll:

```
x++; // erhöhe x um eins
```

Was soll man dokumentieren?

- welche Werte zulässige Eingaben für eine Methode sind
- wie Methoden mit unzulässigen Eingaben umgehen
- wie Fehler behandelt und an den Aufrufer der Methode zurückgegeben werden
- Verträge (Vorbedingungen, Nachbedingungen, Klasseninvarianten)
- Seiteneffekte von Methoden (Zustandsänderungen, Ein- und Ausgabe, usw.)
- wie die Klasse mit Nebenläufigkeit umgeht
- (wie Algorithmen funktionieren, wenn das nicht leicht vom Programmtext ablesbar ist)

Javadoc-Kommentare

- Javadoc-Kommentare
  `/** HTML-formatierter Kommentar */`
- stehen vor Packages, Klassen, Methoden, Variablen
- spezielle Tags wie `@see`, `@link`
- HTML-Dokumentation wird erzeugt mit

```
javadoc Package
javadoc Klasse1.java Klasse2.java ...
```

Javadoc-Tags

Allgemein

- `@author Name`
- `@version text`

Vor Methoden

- `@param Name Beschreibung` – beschreibe einen Parameter einer Methode
- `@return Beschreibung` – beschreibe den Rückgabewert einer Methode
- `@throws Exception Beschreibung` – beschreibe eine Exception, die von einer Methode ausgelöst werden kann

Javadoc-Tags

Querverweise

- `{@link Klasse}`
  `{@link Klasse#Methode}`
  ...
  Verweis auf andere Klassen und Methoden im laufenden Text (in HTML-Version werden daraus Links, Warnung wenn nicht existent)
- `@see Klasse`
  `@see Klasse#Methode`
  ...
  Fügt der Dokumentation einen „See Also“-Abschnitt mit Querverweisen hinzu.

  `@see`-Tags können nicht im laufenden Text stehen

Javadoc-Beispiel

```
/**
 * Allgemeine Kontenklasse
 * @author banker
 * @see NichtUeberziehbaresKonto
 */
public class Konto {

   /**
    *  Geld auf Konto einzahlen.
    *  <p>
    *  Wenn vorher <code> getKontoStand() = x </code>
    *  und <code> betrag >=0 </code>,
    *  dann danach <code> getKontoStand() = x + betrag </code>
    *  @param betrag positive Zahl, der einzuzahlende Betrag
    *  @throws ArgumentNegativ wenn betrag negativ
    */
   public void einzahlen(double betrag)
}
```