Softwareentwicklung II (IB)

Javadoc

Prof. Dr. Oliver Braun

Fakultät für Informatik und Mathematik
Hochschule München

Letzte Änderung: 28.01.2020 17:34

Inhaltsverzeichnis

API-Dokumentation

- Spezielle Dokumentation im Java Source Code um HTML-Dateien zu erzeugen
- Javadoc-Kommentar gleicht dem Blockkommentar, wird aber mit `/**` eingeleitet
- Beendet wird der Javadoc-Kommentar mit `*/`
- Damit er auch im Quelltext gut lesbar ist, beginnt man üblicherweise jede Zeile mit einem `*`

Beispiel

```
/**
 * Ein Hello-World-Programm in Java.
```

```
 *
 * @author Hugo Meier
 * @version 1.0
 */
public class Hello {
    /**
     * Hauptprogramm.
     *
     * @param args Kommandozeilenparameter
     */
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

Was wird kommentiert?

- Klassen
- Objektvariablen
- Methoden
- Der Javadoc-Kommentar steht immer **über** der Entität die kommentiert wird.
- mit Javadoc-Kommentaren **muss** alles kommentiert werden, was `public` ist
- selbstverständlich bietet es sich an, auch alles Sonstige so zu dokumentieren

Aufbau

- Titelzeile (Pflicht)
  - kann über mehrere Zeilen verteilt sein
  - endet mit einem Punkt

```
/**
 * Ein Hello-World-Programm in Java.
 */
public class Hello {
```

- Langtext (optional)
  - beginnt nach der Titelzeile
  - freier Text
- Tags
  - je nach Entität

Tags

- Klasse

```
* @author Oliver Braun
* @author David Yow
* @version 1.0
```

  - beide Tags können mehrfach vorkommen
  - nur Text

- Objektvariablen
  - keine

- Methoden

```
* @param numberOne  erste Zahl, kleiner als 1000
* @param numberTwo  zweite Zahl, beliebig groß
* @return Maximum der beiden Zahlen
* @throws NumberOneIsTooLargeException wenn numberOne >= 1000
```

  - zwei der Tags können mehrfach vorkommen

`@return` Text, Beschreibung **was** das Ergebnis ist
`@param` Parametername Zweck und Einschränkungen der Werte
`@throws` Exceptiontyp und wann sie geworfen wird

Aufruf

- das `javadoc`-Tool ist im JDK enthalten

```
$ javadoc -help
usage: javadoc [options] [packagenames] [sourcefiles] [@files]
...
-private     Show all classes and members
...
-version     Include @version paragraphs
-author      Include @author paragraphs
...
-link <url>  Create links to javadoc output at <url>
...
```

- nutzen Sie z.B.

```
$ javadoc -private -version -author \
    -link http://docs.oracle.com/javase/8/docs/api/ \
    <java-files>
```

- Gradle bietet ein `javadoc`-Target:

  `./gradlew javadoc`

- wenn das erfolgreich durchläuft, finden Sie die generierte Dokumentation im Verzeichnis `build/docs/javadoc`
  - öffnen Sie die Datei `index.html` mit einem Browser (rechte Maustaste in IntelliJ IDEA)

Beispiele

```
/**
 * Ein Hello-World-Programm in Java.
 *
 * @author Oliver Braun
 * @author David Yow
 * @version 1.0
 */
public class HelloApp {

  /**
   * Hauptprogramm.
   *
   * @param args Kommandozeilenparameter
   */
  public static void main(String[] args) {
    // ...
  }

  /**
   * Repräsentiert den Text
   * eines Hello-Objektes.
   * Default ist "Hello, world!".
   */
  public final String helloTxt = "Hello, world!";

  /**
   * Leerer Default-Konstruktor.
   */
  public Hello() {}

  /**
   * Custom-Konstruktor mit dem der Hello-Text
   * gesetzt werden kann.
   *
```

```
   * @param helloTxt Text für das neue Objekt.
   */
  public Hello(String helloTxt) {
    this.helloTxt = helloTxt;
  }

  /**
   * Umwandlung eines Hello-Objektes in einen String.
   *
   * @return den Hello-Text
   */
  @Override
  public String toString() {
    return this.helloTxt;
  }
}
```