universität innsbruck

Javadoc

Programmiermethodik

Lukas Kaltenbrunner, Simon Priller
Universität Innsbruck

Motivation

- Konsistente Dokumentation von Quelltext ist schwierig, wenn die Dokumentation und der Quelltext getrennt sind.
  - Der aktuelle Stand des Quelltextes kann sich von der Dokumentation unterscheiden (Programm funktioniert auch ohne neue Dokumentation).
- Dokumentationskommentare heben diese Trennung auf.
  - Es gibt nur eine einzige Datei – Quelltext – wobei der Quellcode diese speziellen Dokumentationskommentare enthält.
  - Die Dokumentation kann aus dem Quellcode generiert werden.
- In Java:
  - Dokumentationskommentar:

```
/**
...
*/
```

  - Aus Dokumentationskommentaren kann durch das Javadoc-Tool eine HTML-Dokumentation generiert werden.

Inhalt von Kommentaren

- Englische Sprache verwenden → guter Stil
- Allgemein:
  - Sollten nur verwendet werden, wenn sie tatsächlich einen Mehrwert bringen.
  - Deuten vielfach auf Verbesserungsmöglichkeiten im Code hin:
    - Vergabe sprechender Namen.
    - Anwenden der Clean Code Tipps (z.B. Fail Fast, Java API over DIY, …).
  - Dokumentieren von Implementierungsentscheidungen.
  - `++i; // increment i by one` bringt beispielsweise keinen Mehrwert.

Dokumentation

- Alle Elemente der exportierten API sollten mit Dokumentationskommentaren versehen werden.
  - Klassen, Interfaces, Records, Members, Konstruktoren und die serialisierte Form werden als API-Elemente bezeichnet.
  - Die exportierte API besteht aus allen Elementen, welche außerhalb des Pakets sichtbar sind.
- Sollte möglichst viele Informationen bereitstellen.
- Durch Lesen der Dokumentation sollten Mehrinformationen erhalten werden.
- Alle formalen Typparameter von generischen Typen sollten dokumentiert werden.
- Exceptions bei public und protected Methoden sollten dokumentiert werden.

Dokumentationskommentare

- Werden nicht beliebig im Text verstreut.
- Stehen vor der Definition von
  - Klassen und Interfaces
  - Methoden
  - Datenelementen
- Aufbau (typisch)
  1. Zusammenfassung in einem einzigen Satz mit einem Punkt am Ende.
  2. Ausführliche Beschreibung.
  3. Block Tags (jeweils gefolgt von einem Text) um zusätzliche Abschnitte in der Dokumentation zu erzeugen.
     - z.B. ein Abschnitt der die Parameter oder den Rückgabewert einer Methode erklärt.

Javadoc-Tags

- Tags markieren Informationen mit bestimmter Bedeutung.
- Es werden zwei Arten von Tags unterschieden:
  - Block Tags (der Form `@tag`)
    - Müssen am Anfang der Zeile stehen (nach eventuellen Leerzeichen und *).
    - Werden ansonsten als normaler Text angesehen.
    - Gleichnamige Tags sollten gruppiert werden.
  - Inline Tags (der Form `{@tag}`)
    - Können in der anfänglichen Beschreibung und in den Texten von Block Tags stehen.
    - Sind immer von `{}` umgeben.

Auszug Javadoc-Tags (1)

Tag	Paket	Klasse/ Interface	Feld	Methode
@author	✓	✓		
@version	✓	✓		
@param		✓		✓
@return				✓
@throws und @exception				✓
@see	✓	✓	✓	✓
@since	✓	✓	✓	✓
@serial	✓	✓	✓	
@serialField			✓	
@serialData				✓
@deprecated	✓	✓	✓	✓
{@link}	✓	✓	✓	✓
{@linkplain}	✓	✓	✓	✓
{@inheritDoc}				✓
{@docRoot}	✓	✓	✓	✓
{@value}			✓	
{@literal}	✓	✓	✓	✓
{@code}	✓	✓	✓	✓

Weitere Informationen: Javadoc Tags, Tag-Konventionen

Auszug Javadoc-Tags (2)

- Pakete, Klassen/Interfaces, Felder und Methoden

  `@see` *`reference`*
  - Referenz auf andere Ressourcen (z.B. URLs, Klassen, Methoden, Pakete).
  - `@see` *`Class#method(Type argname, Type argname,...)`*

  `@since` *`version`*
  - Gibt an seit welcher Version dieses Element existiert.

  `@deprecated` *`text`*
  - Markiert ein veraltetes Element, das nicht mehr verwendet werden soll.

  `{@link` *`package.class#member label`*`}`
  - Um inline Links in Beschreibungen anzugeben .

  `{@literal` *`text`*`}`
  - Interpretiert *`text`* nicht als HTML bzw. verschachtelte Javadoc Tags.

  `{@code` *`text`*`}`
  - Wie `@literal` und *`text`* wird zusätzlich Quellcodeformatierung durchgeführt.

Auszug Javadoc-Tags (3)

- Pakete, Klassen und Interfaces

  `@author` *`text`*
  - Beschreibt den*die Autor*in. Kann mehrfach verwendet werden.

  `@version` *`text`*
  - Gibt die aktuelle Versionsnummer der Software an.

- Klassen, Interfaces und Methoden

  `@param` *`name text`*
  - Beschreibt den Parameter *`name`* der Methode oder des Konstruktors.
  - Beschreibt den Typparameter *`<name>`* einer Methode, eines Konstruktors oder einer Klasse.

- Methoden

  `@return` *`text`*
  - Beschreibt den Rückgabewert einer Methode.

  `@throws` *`exception text`*
  - Beschreibt eine Exception die von dieser Methode oder diesem Konstruktor geworfen werden kann.
  - Checked Exceptions sollten dokumentiert werden.
  - Unchecked Exceptions können dokumentiert werden, falls davon ausgegangen werden kann, dass Aufrufende diese Exception abfangen und behandeln oder, in einer der Abstraktionsebene angemesseneren Exception verpackt, weiterreichen möchten.

Beispiel

```java
/**
 * A bank account that allows depositing and withdrawing money without overdrawing.
 *
 * @author Franz Hacker
 * @version 1.1.1
 * @since 1.0.0
 */
public final class BankAccount {

    private final String accountHolder;

    private BigDecimal balance; // Never use float or double for money because of rounding errors!

    …

    /**
     * Withdraws the amount from the account.
     *
     * <p>
     * Withdraws money up to the total balance. Overdrawing is impossible. In case overdraw would happen only the
     * available money would be withdrawn.
     *
     * @param amount Amount to be withdrawn.
     *               Must not be negative, must not be null.
     * @return Amount that was actually withdrawn.
     * If the balance was not enough to cover the amount requested by the parameter only the available balance
     * is returned.
     * @throws IllegalArgumentException if amount is not greater than zero.
     * @throws NullPointerException     if amount is null.
     */
    public BigDecimal withdraw(final BigDecimal amount) {…}

    …
}
```

src/at/ac/uibk/pm/documentation/BankAccount.java

HTML in Javadoc Kommentaren

- Der Inhalt von Javadoc-Kommentaren wird in HTML-Dateien übersetzt.
- Daher können auch HTML-Tags verwendet werden.
- Beispiel

```
@author Lukas Kaltenbrunner
           <a href="mailto:lukas.kaltenbrunner@uibk.ac.at">
                 lukas.kaltenbrunner@uibk.ac.at
           </a>
```

javadoc-Compiler (1)

- Für die Dokumentation gibt es einen eigenen Compiler – `javadoc`
- Übersetzt die Dokumentation in das HTML-Format.
  - `javac` blendet beim Kompilieren die Dokumentation aus.
- Einige Optionen

`-d path`	Alle generierten HTML-Seiten werden im Directory `path` abgelegt.
-public	Nur `public`-Elemente werden in die Dokumentation aufgenommen.
-author	Übernimmt das `@author`-Tag (nicht default).
-version	Übernimmt das `@version`-Tag (nicht default).
-help	Gibt eine Liste der Schalter von `Javadoc` aus.
-subpackage <pkg>	List von Subpaketen

- Aufruf (Beispiele)
  - `javadoc BankAccount.java`
  - `javadoc -d doc at.ac.uibk.pm.documentation`

javadoc-Compiler (2)

- Auszug erzeugter Dateien

Datei	Inhalt
index.html	Startpunkt
index-all.html	Übersicht über alle Klassen, Schnittstellen, Ausnahmen, Methoden, Felder
overview-tree.html	Baumstruktur der Klassen
allclasses-frame.html	Alle Klassen in allen Unterpaketen
deprecated-list.html	Liste aller veralteten Methoden und Klassen
serialized-form.html	Alle Klassen, die Serializable implementieren
help-doc.html	Kurzbeschreibung zu Javadoc

Output (Auszug)

OVERVIEW PACKAGE CLASS TREE DEPRECATED INDEX HELP

ALL CLASSES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

SEARCH: Search

Package at.ac.uibk.pm.documentation

Class BankAccount

java.lang.Object
    at.ac.uibk.pm.documentation.BankAccount

```
public class BankAccount
extends java.lang.Object
```

A bank account that allows depositing and withdrawing money without overdrawing.

Since:

1.0.0

Constructor Summary

Constructors

Constructor	Description
`BankAccount(java.lang.String accountHolder)`	Creates an empty bank account for the account holder.
`BankAccount(java.lang.String accountHolder, java.math.BigDecimal balance)`	Creates a bank account with an initial balance for the account holder.

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
`void`	`deposit(java.math.BigDecimal amount)`	Increases the balance of the account.
`java.lang.String`	`getAccountHolder()`	Returns the name of the account holder.
`java.math.BigDecimal`	`getBalance()`	Returns the current balance.
`java.math.BigDecimal`	`withdraw(java.math.BigDecimal amount)`	Withdraws the amount from the account.

Javadoc mit der IDE generieren

- Mit Eclipse
  - `Project -> Generate Javadoc`
- Mit IntelliJ
  - `Tools -> Generate Javadoc`

Quellen

- Joshua Bloch: **Effective Java**, Addison-Wesley Professional, 3. Auflage, 2018
- Oracle: **Documentation Comment Specification for the Standard Doclet (JDK 17)**, besucht am 13.05.2022, https://docs.oracle.com/en/java/javase/17/docs/specs/javadoc/doc-comment-spec.html
- Oracle: **How to Write Doc Comments for the Javadoc Tool**, besucht am 13.05.2022, https://www.oracle.com/technical-resources/articles/java/javadoc-tool.html