

Programmierung
Prof. Dr.-Ing. Nobert Fuhr

Gudrun Fischer
Sascha Kriewel
programmierung@is.informatik.uni-duisburg.de

Übungsblatt Nr. 13

Aufgabe 25: Dokumentieren mit Javadoc

Das Dokumentationswerkzeug `javadoc` und das Format von `javadoc`-Kommentaren wurden in der Vorlesung vorgestellt.

Versieh deine Lösung zum zweiten Testat (alternativ: die Musterlösung zu Aufgabe 23, Blatt 12) mit `javadoc`-Kommentaren.

Kommentiere jede Klasse, und in jeder Klasse jede Instanzvariable, jede Klassenvariable und alle Methoden. Verwende die Tags `@param` und `@return`, um Parameter und Rückgabewert von Methoden explizit zu erklären. Verwende ggf. das Tag `@throws`, um die in einer Methode auftretenden Exceptions zu beschreiben, und `@see` für Querverweise.

Einen Überblick über `javadoc`-Kommentare findest du unter
<http://java.sun.com/j2se/javadoc/writingdoccomments/index.html>

Generiere zum Schluss HTML-Seiten zur Dokumentation:
`javadoc -private <deine java-Dateien inklusive Pfad>`
Verwende dabei die Option `-private`, damit auch private Variablen und Methoden angezeigt werden.

Tipp:

Auch die Java-API selbst wurde mit `javadoc` dokumentiert. Nimm sie im Zweifelsfall als Beispiel, wenn du dir in einer Frage nicht sicher bist.

Aufgabe 26: Erstellen eines ausführbaren Jar-Archivs

Erstelle für deine Lösung zum zweiten Testat (alternativ: für die Musterlösung zu Aufgabe 23, Blatt 12) mit Hilfe des Werkzeugs `jar` ein Jar-Archiv, das alle Quelldateien, alle kompilierten Dateien und die in Aufgabe 25 generierte HTML-Dokumentation enthält.

Schlage das Vorgehen ggf. im folgenden Tutorial nach:
<http://java.sun.com/docs/books/tutorial/jar/basics/build.html>

Erweitere das Manifest deines Jar-Archivs um die Information, welche Klasse standardmäßig ausgeführt werden soll.

Siehe dazu:

<http://java.sun.com/docs/books/tutorial/jar/basics/mod.html> und
[http://java.sun.com/docs/books/tutorial/jar/basics/manifest.html#
applications](http://java.sun.com/docs/books/tutorial/jar/basics/manifest.html#applications)

Hinweis:

Statt eines direkten Aufrufs von `jar` kannst du auch in `eclipse` bei `export` die Form `jar` wählen und dann die richtigen Optionen aktivieren.

Teste schließlich dein so erzeugtes Jar-Archiv, indem du es mit

```
java -jar <Archivname>.jar
```

aufrufst.

Aufgabe 27: Komposition, Delegation, Aggregation, Assoziation

Als Material zu dieser Aufgabe findest du auf der Vorlesungsseite das Interface `Verkehrsteilnehmer`, sowie die Klassen `Fahrrad`, `Mensch` und `VerkehrsteilnehmerTest` im Archiv `verkehrsteilnehmer.zip`. Das Archiv enthält auch mit Javadoc generierte HTML-Seiten zur Dokumentation.

Was modelliert das Interface `Verkehrsteilnehmer`? Warum ist ein Mensch in diesem Beispiel ein Verkehrsteilnehmer, ein Fahrrad jedoch nicht?

Ein Fahrrad kann einen Besitzer vom Typ Radfahrer haben. Diese Klasse fehlt im Beispiel offensichtlich. Schreibe eine Klasse `Radfahrer`, die einen Radfahrer modelliert. Verwende dabei die Klassen `Mensch` und `Fahrrad` sinnvoll.

In einer Umgebung, in der Fahren erlaubt ist, fährt der Radfahrer auf dem Rad. In einer Umgebung, in der Fahren nicht erlaubt ist, geht der Radfahrer zu Fuß. Überschreibe die Methoden `setUmgebung` und `fortbewegen` passend.

Wo und inwiefern treten in deiner Lösung die Konzepte Komposition, Delegation, Aggregation und Assoziation auf?