Kapitel 12 Dokumentation und Zugriffsrechte

Lernziele:
Dokumentation mit Hilfe von Javadoc
Datenkapselung über Zugriffsrechte

12.1 Dokumentation

Abbildung 1 zeigt das erweiterte Klassendiagramm der Klasse MAMPFI. Da aussagekräftige Namen für die Attribute und Methoden gewählt wurden, ist es für die Verwendung von dieser Klasse bereits sehr hilfreich. Jedoch beantwortet es folgende Fragen nicht.

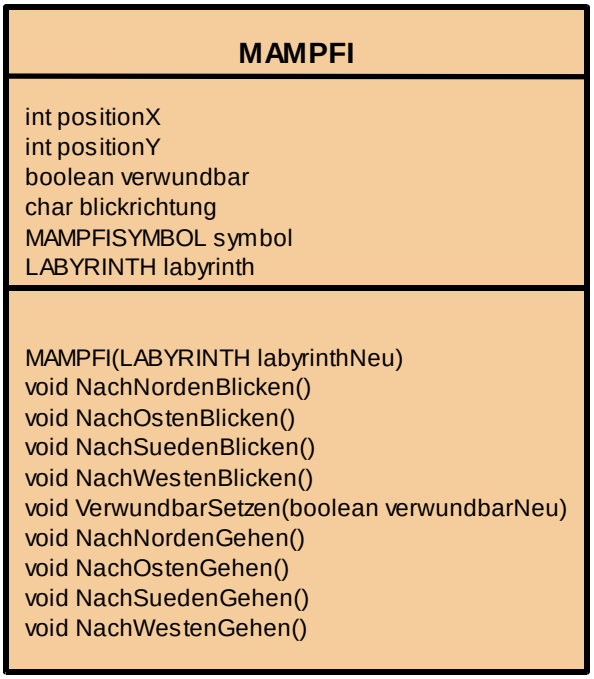


Abbildung 1: Erweitertes Klassendiagramm der Klasse MAMPFI

- Welche Werte sind für das Attribut blickrichtung erlaubt? Sind vielleicht 'R', 'O', 'L' und 'U' für rechts, oben, links und unten zulässige Werte?
- Wie viele Schritte bewegt sich ein Objekt, beim Aufruf der Methoden NachNordenGehen, NachOstenGehen, usw. ?
- Welche Werte werden für die Attribute im Konstruktor gesetzt?

Aufgabe 12.1
Warum könnten diese Informationen für einen Außenstehenden interessant sein?
Warum könnte es sogar für dich selbst wichtig sein, diese Details schnell nachlesen zu können?

Software wird in der Regel von mehreren Entwicklern zusammen entwickelt. Dabei ist es wichtig, dass jeder Programmierer sich schnell über das Wesentliche aller Klassen, d.h. auch der von ihm nicht verfassten, informieren kann. Auch für den Entwickler selbst kann es schwer sein, sich an Details zu erinnern, wenn die Implementierung zeitlich weit zurück liegt und die Anzahl der Klassen hoch ist. Ein Einlesen in den Quelltext ist zeitaufwändig und mühsam.
Aus diesen Gründen ist es wichtig, selbst verfasste bzw. geänderte Klassen zu dokumentieren und damit Wissen zu teilen. Die Dokumentation einer Klasse sollte genau die Informationen beinhalten, die ein anderer Entwickler benötigt, um die Klasse ohne einen Blick auf den Quelltext verwenden zu können. Abbildung 2 zeigt eine mögliche Dokumentation der Klasse MAMPFI.

Klasse MAMPFI

Die Klasse MAMPFI beschreibt Hauptakteure im Spiel Krümel & Monster.

Autor Peter Brichzin
Version 9.2.09

Attribute

int	**positionX**
int	**positionY**
boolean	**verwundbar** Falls der Wert von verwundbar true ist, kann Mampfi von den Monstern gefressen werden. Falls der Wert von verwundbar false ist, kann Mampfi nicht von den Monstern gefressen werden, sondern sie selbst fressen.
boolean	**blickrichtung** Es gibt vier mögliche Werte für die Blickrichtung. 'N': Mampfi blickt nach Norden (oben). 'O': Mampfi blickt nach Osten (rechts). 'S': Mampfi blickt nach Süden (unten). 'W': Mampfi blickt nach Westen (lins).

Referenzattribute

MAMPFISYMBOL	**symbol** Referenz auf das Symbol, das Mampfi auf der Spieloberfläche darstellt.
LABYRINTH	**labyrinth** Referenz auf das Labyrinth, in dem sich Mampfi bewegt.

Konstruktor

MAMPFI(LABYRINTH labyrinthNeu)
Vor dem Erzeugen eines Objekts muss ein Labyrinth existieren, in dem sich dann Mampfi bewegt. Beim Erzeugen eines Objekts werden momentan die Startposition und die Startblickrichtung willkürlich, der Wert des Attributs verwundbar auf true gesetzt.

Eingangsparameter:
labyrinthNeu - Labyrinth, in dem sich Mampfi bewegt

Methoden

void	**NachNordenBlicken()** Setzt den Wert des Attributs blickrichtung auf 'N' und passt die entsprechenden Attributwerte des darstellenden Symbols an.
void	**NachNordenGehen()** Lässt Mampfi einen Schritt nach Norden gehen, falls die Grenzen der Spielfläche nicht überschritten werden und keine Mauer vor ihm ist.
void	**VerwundbarSetzen(boolean verwundbarNeu)** Setzmethode des Attributs verwundbar Eingangsparameter: verwundbarNeu -
	. . .

Abbildung 2: Dokumentation der Klasse MAMPFI

12.2 Dokumentationen automatisch erzeugen

Im Internet gibt es unzählige Java Klassen, die zur freien Verfügung stehen. Um Programmierern die Dokumentation zu erleichtern, stellt Java ein eigenes Programm zur Verfügung, das Dokumentationen in Form von Webseiten generiert. Die Information holt sich Java aus speziellen Quelltext-Kommentaren, den Javadoc-Kommentaren. Diese beginnen nicht wie einfache Kommentare mit "/*" sondern mit "/**".

Abbildung 3 zeigt den Javadoc-Kommentar für das Attribut blickrichtung. Durch die Position des Kommentars vor der Deklaration müssen Datentyp und Attributname nicht in den Javadoc-Kommentar aufgenommen werden, sie werden automatisch in die Dokumentation übernommen.

```
/** Es gibt vier mögliche Werte für die Blickrichtung. <BR>
 *  'N': Mampfi blickt nach Norden (oben).<BR>
 *  'O': Mampfi blickt nach Osten (rechts).<BR>
 *  'S': Mampfi blickt nach Süden (unten).<BR>
 *  'W': Mampfi blickt nach Westen (lins).
 */
 char blickrichtung;
```

Abbildung 3: Javadoc-Kommentar für das Attribut blickrichtung: Der Kommentar steht vor der Attributdeklaration und beginnt mit der Zeichenfolge "/**"

Hinweis:
Werden besondere Formatierungen gewünscht, sind diese mit HTML-Tags möglich. So wird in dem Javadoc-Kommentar in Abbildung 3 nach der Beschreibung jedes konkreten Werts ein Zeilenumbruch (engl. break bzw. linebreak) durch den HTML-Tag
 erzwungen, weil die Dokumentation dadurch übersichtlicher wird.
Da im englischen Zeichensatz keine deutschen Umlaute vorkommen, müssen für eine korrekte Darstellung Umlaute kodiert werden:
ä: auuml; Ä: Azuml; ö: ö Ö:Ouml; ü: ü Ü: Ü ß:ß
Eine einfacheAlternative ist die Umlaute als ae, oe, ue usw. zu schreiben.

Weiterhin stehen einige Schlüsselwörter zur Verfügung, mit denen die Dokumentation automatisch formatiert wird. Die wichtigsten zeigt die folgende Tabelle:

Schlüsselwort	**Bedeutung**
@author	Autor der Klasse
@version	Versionsnummer bzw. Erstelldatum
@param	Eingabeparameter einer Methode
@return	Rückgabewert einer Methode

```
/**
* Vor dem Erzeugen eines Objekts muss ein Labyrinth existieren, in dem sich dann
* Mampfi bewegt.Beim Erzeugen eines Objekts werden momentan die
* Startposition und die Startblickrichtung  willk&uuml;rlich, der Wert des
* Attributs verwundbar auf true gesetzt.
* @param labyrinthNeu Labyrinth, in dem sich Mampfi bewegen soll.
*/
MAMPFI(LABYRINTH labyrinthNeu)
{
   positionX = 3;
   positionY = 2;
    ...
```

Abbildung 4:
Javadoc-Kommentar für den Konstruktor mit Verwendung des Schlüsselworts "@param"

Aufgabe 12.2
Ergänze in der Klasse MAMPFI Javadoc-Kommentare. Eine Anzeige der Dokumentation ist möglich über das Ausklappmenü im Editorfenster rechts oben (Abbildung 5). Ähnlich zum Programmieren muss man die erstellte Dokumentation lesen (= testen) und dann bei Bedarf noch nachbearbeiten.

Abbildung 2: Dokumentation der Klasse MAMPFI

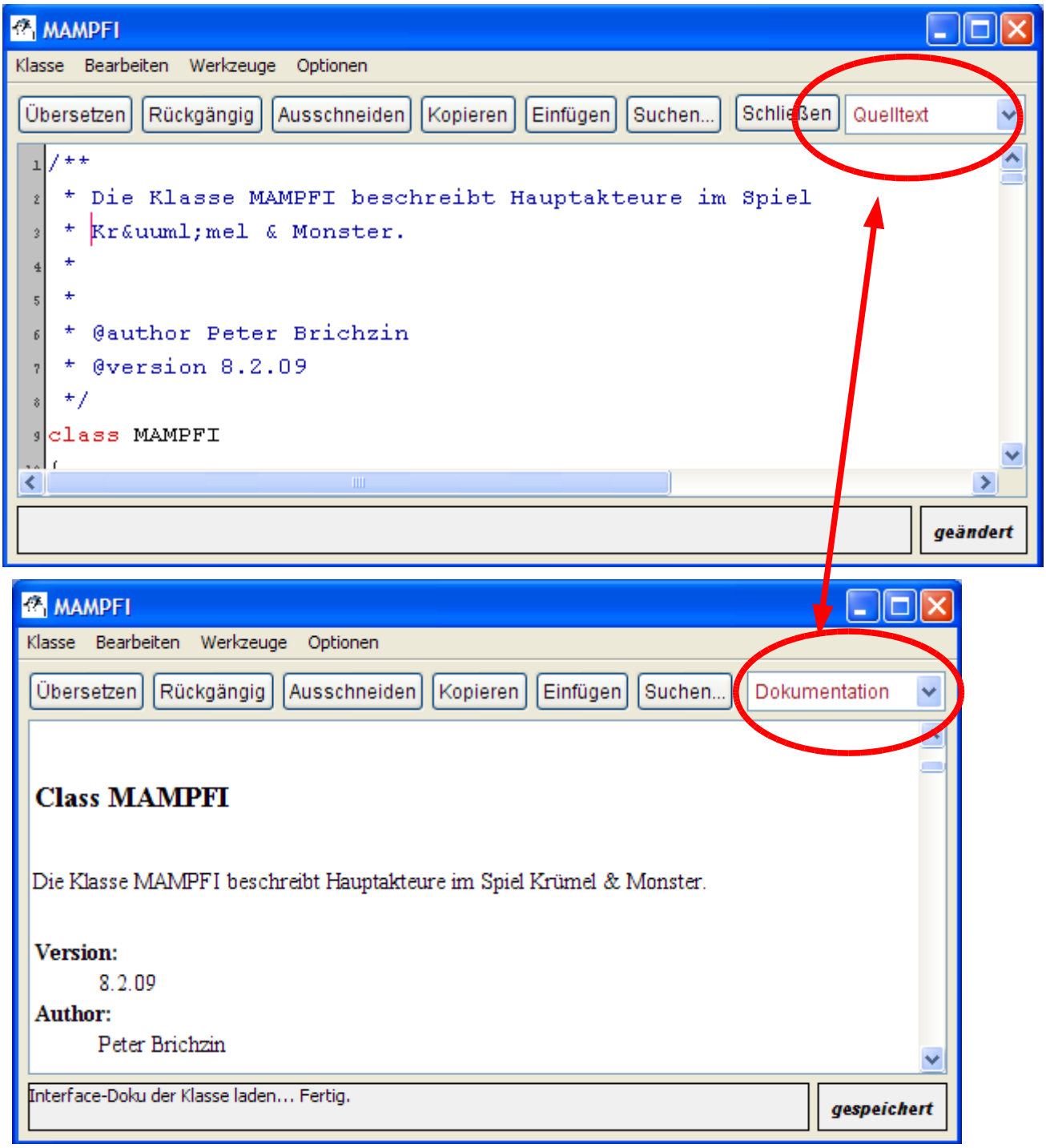


Abbildung 5: Wechsel zwischen Quelltext und Dokumentation in BlueJ

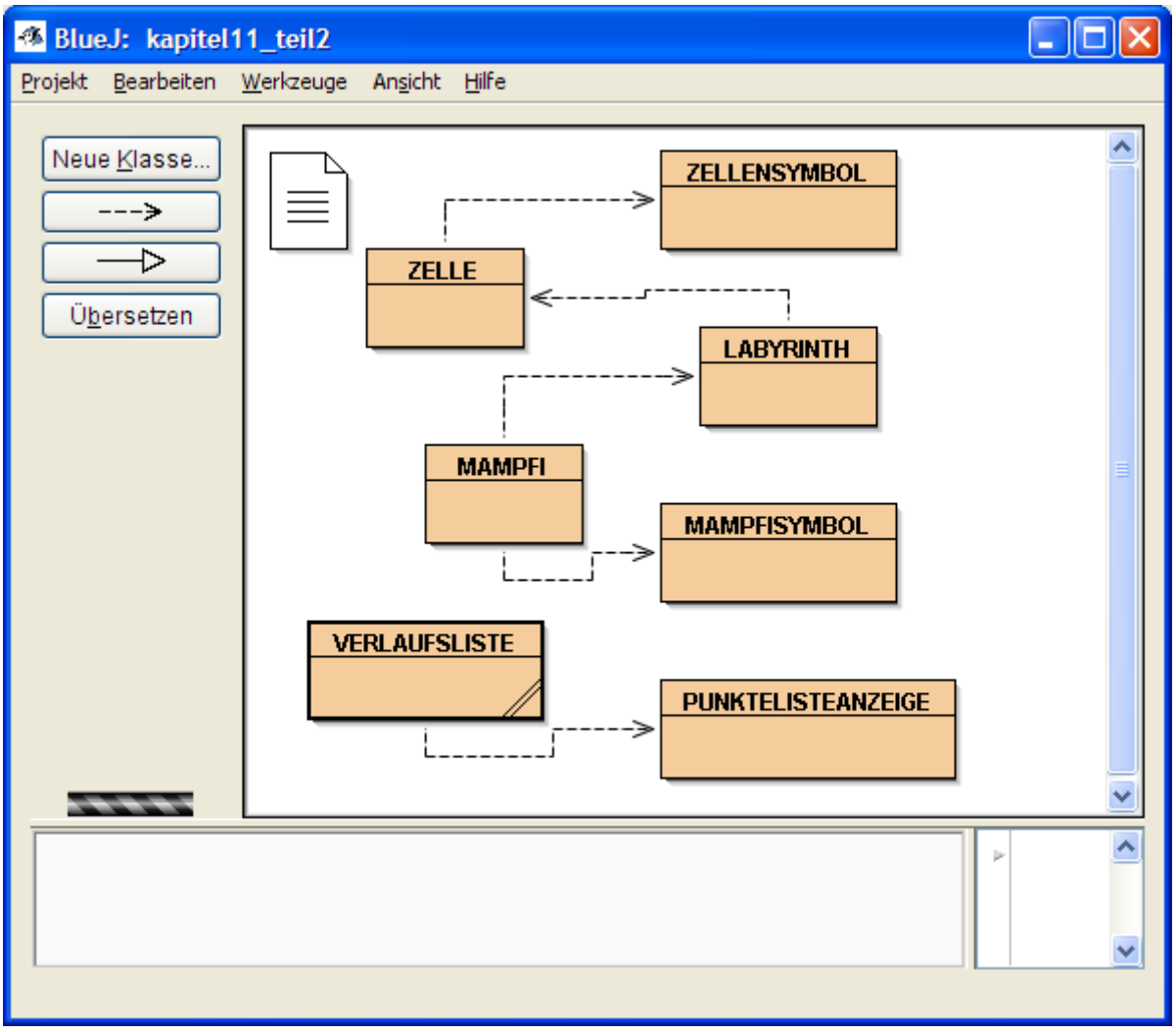
Hinweise:
Mit Javadoc lässt sich die Dokumentation aus Abbildung 2 nicht exakt nachbilden.
Unterschiede sind:

- Verwendung der englischer Fachbegriffe field (Attribut), constructor (Konstruktor) und method (Methode)
- Referenzattribute werden von Attributen nicht unterschieden.
- Der Aufbau der Webseite ist dergestalt, dass zuerst Zusammenfassungen (summaries) aller Javadoc-Kommentare aufgelistet sind. Als Zusammenfassung wird der Text bis zum ersten Punkt interpretiert. Danach folgen Detailinformationen (details).

- Bei den Attributen und Methoden werden auch die Zugriffsrechte angezeigt, diese werden erst in Kapitel 12.3 vorgestellt.

Aufgabe 12.3

Im Werkzeugkasten 4 vom Schulbuch Informatik II wird erklärt, wie man eine Dokumentation von dem gesamten Projekt automatisch erzeugen kann. Erstelle diese Dokumentation von den selbst erstellten Klassen, d.h. MAMPFI, ZELLE, LABYRINTH und VERLAUFSLISTE.



12.3 Daten kapseln durch Zugriffsrechte

Dieses Teilkapitel wird zu einem späteren Zeitpunkt verfasst:

Die Inhalte können erarbeitet werden durch das Buch Informatk II, Kapitel 8
S 69 unten ab der Teilüberschrift "Testen"
bis zum Merkkasten auf Seite 71 oben.

Aufgabe 12.4
Kapsle die Daten in deinem Projekt, indem du allen Attributen (auch Referenzattributen) das Zugriffsrecht `private` und allen Methoden das Zugriffsrecht `public` gibst. Teste dein Projekt. Sollte es nicht mehr funktionieren, hat ein Objekt auf die Attribute eines anderen Objekts direkt (z.B. durch eine Zuweisung) und nicht indirekt über Methoden zugegriffen. Korrigiere bei Bedarf.

12.4 Zusammenfassung

Aufgabe 12.5
Fasse die wesentlichen Inhalte dieses Kapitels zusammen!
Dazu gehören Antworten auf folgende Fragen:

- Warum ist eine Dokumentation sinnvoll?
- Wie kann man in Java einfach eine Dokumentation eines Projekts durch eine Webseite erstellen? (Nur eine knappe Antwort ist nötig!)
- Was ist eine Datenkapselung?
- Wie lässt sich eine Datenkapselung mit Hilfe der Zugriffsrechte erreichen?