Softwaretechnik-Praktikum SoSe 2005
Aufgabenblatt 5

Dokumentationskonzept

Für die Softwareentwicklung werden in der Gruppe die folgenden Standards und Regeln festgelegt, an die sich beim Entwickeln des Quellcodes jedes Teammitglied zu halten hat.

Javadoc-Kommentare

Es wird zur Erstellung einer Klassendokumentation auf das Javadoc-Tool zurückgegriffen. Um dieses sinnvoll nutzen zu können ist jeder Klasse und jeder Methode, die mindestens die Sichtbarkeit protected besitzt eine javadoc-konforme Beschreibung voranzustellen. Diese Beschreibung enthält in der ersten Zeile eine kurze Zusammenfassung, was die Klasse oder Methode leistet, sowie ab der zweiten Zeile weitergehende Informationen über die Funkionen. In der Klassendokumentationen ist zusätzlich der Name des Autors nach dem Schlüsselwort `@author` anzugeben, sowie einer Versionsnummer nach dem Schlüsselwort `@version`. Bei Methodendokumentationen ist bei anderen Rückgabewerten als `void` eine Erläuterung des Rückgabewertes nach dem Schlüsselwort `@return`, sowie eine Erläuterung zu jedem Übergabewert nach dem Schlüsselwort `@param` anzugeben. Sinnigerweise ist die Javadoc-Dokumentation schon während des Programmiervorgangs anzufertigen, beziehungsweise beim Erstellen des Code-Gerüsts.
Nachfolgend ein Beispiel für eine typische Javadoc-Dokumentation

```
/**
 * Kurze Beschreibung
 * Ausfuehrliche beschreibung, was die Klasse oder Methode leistet.
 * @author Name (Nur bei Klassendokumentation)
 * @version 1.0  (Nur bei Klassendokumentation)
 * @param parametername1 Beschreibung Uebergabewert, auch mehrfach
 * @param parametername2 (Nur bei Methodendokumentation)
 * @return Beschreibung Rueckgabewert (Nur bei Methodendokumentation)
 */
```

Weitere Kommentare im Quelltext

Allgemeine Informationen, die für einen Entwickler gedacht sind, der an der gleichen Klasse arbeitet, und die noch nicht durch die Javadoc-Dokumentation abgedeckt sind, müssen in einem einführenden Blockkommentar an den Anfang der Datei noch vor der `package` und `import`-Anweisung stehen. In diesen Einführenden Block-Kommentar gehören z.B. Informationen, welche Methoden noch überarbeitet werden müßten, oder welche Fehler und Probleme noch zu behandeln sind.
Beispiel:

```
/*
 * foo(); Laufzeit optimieren
 * bar(int x); liefert falsche Ergebnisse fuer x=2
 */
```

Generell hat der Quelltext leicht verständlich aufgebaut zu sein. Bei komplizierten Vorgängen, die auch in mehreren Schritten sofort verständlich programmierbar sind, ist die letztere Variante zu wählen. Sollte es in Ausnahmefällen vorkommen, daß ein Codefragment nicht auf Anhieb von anderen Teammitgliedern verstanden werden kann, ist vor dem entsprechenden Abschnitt in einem Einzeiligen Kommentar der

nachfolgende Vorgang zu erläutern.
Auch Eigenschaftennamen sind aussagekräftig zu wählen, sollte die Problemstellung Mißverständnisse zulassen, ist unmittelbar hinter die Eigenschaftendeklaration mit einem einzeiligen Kommentar die Eigenschaftenfunktion zu erläutern. Beispiel:

```
int p=0; //p-Wert fuer pq-Formel
int q=0; //q-Wert fuer pq-Formel
...
//pq-Formel
x1=p/(-2)+Math.sqrt(p*p/4-q);
x2=p/(-2)-Math.sqrt(p*p/4-q);
```

Quelltext-Konventionen

Es wird darauf Wert gelegt, daß der zu entwickelnde Quelltext sauber formatiert und leicht lesbar ist. Dazu werden neben den bereits angesprochenen Dokumentationsregeln noch weitere Regeln zum Aussehen des Quelltextes festgelegt.
Mit jeder, sich öffnenden geschweiften Klammer ist der Quelltext um vier Leerzeichen (keine Tabs) einzurücken. Analog dazu ist bei einer schließenden geschweiften Klammer nachfolgender Code wieder um vier Leerzeichen auszurücken.
In einer Datei ist immer nur eine öffentliche Klasse zu speichern, es ist jedoch erlaubt, Klassen mit der Sichtbarkeit `private` mit in die Datei einer öffentlichen Klasse zu schreiben, sofern die beiden Klassen zu einer logischen Gemeinschaft gehören. Dabei steht die öffentliche Klasse aber immer vor der privaten.
Die Reihenfolge in einer Klasse sieht wie folgt aus:

- private Eigenschaften
- Konstruktoren
- öffentliche Getter– und Setter-Methoden
- öffentliche Methoden
- protected und standard-Methoden
- private Methoden

Bezeichner sind stets so zu wählen, daß keine Sonderzeichen in ihnen vorkommen. Also so, daß sie lediglich aus Buchstaben von a bis z und Ziffern von 0 bis 9 bestehen. Dabei darf ein Bezeichner aber nicht mit einer Ziffer beginnen.
Eigenschaftennamen beginnen stets mit einem kleinen Buchstaben, sollte ein Eigenschaftenname aus mehreren Wörtern bestehen, sind diese als ein Wort zu schreiben, wobei jedes neue Wort mit einem Großbuchstaben beginnt. Für Eigenschaftennamen müssen aussagekräftige, aber möglichst kurze Namen gewählt werden. Es ist davon abzusehen, Eigenschaftennamen wie `i` oder `n` zu verwenden. Einzige geduldete Ausnahme sind Iteratorvariablen in `for`-Schleifen, wenn der Iterator erst in dem Schleifenkopf deklariert wird. Eigenschaften haben immer die Sichtbarkeit `private` und dürfen nur durch geeignete Get– oder Set-Methoden verändert werden.
Methodennamen bestehen stets aus einem Verb, einem Verb mit einem Adjektiv, einem Verb mit einem Substantiv oder beidem. Für Methodennamen gilt ansonsten selbiges, wie für Eigenschaftennamen, mit Ausnahme der Sichtbarkeitsregel.
Klassennamen bestehen Stets aus einem Nomen. Ein Klassenname beginnt immer mit einem Großbuchstaben. Sollte ein Klassenname aus mehr als einem Wort bestehen, dann wird jedes weitere Wort auch mit einem Großbuchstaben begonnen. Mit Ausnahme von gängigen Abkürzungen ist auf die Nutzung dieser zu verzichten.
Für Interface-Namen gilt das gleiche, wie für Klassennamen.
Konstanten sind immer mit Großbuchstaben zu bezeichnen. Sollte eine Konstante aus mehreren Worten

bestehen, dann sind diese untereinander durch den Unterstrich zu trennen.
Der Dateiname einer Klassendatei hat immer mit dem Namen der öffentlichen Klasse, gefolgt von der Endung `.java` übereinzustimmen.