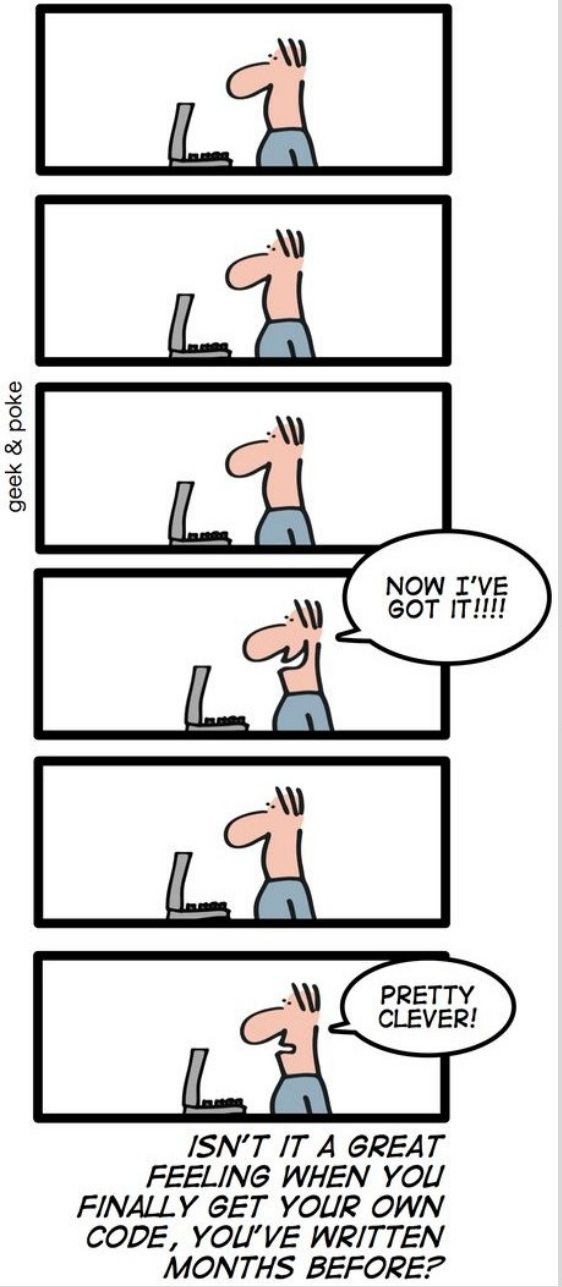
Karlsruher Institut für Technologie

Programmieren WS 2011/2012

Kommentieren & Javadoc

Fakultät für Informatik, Zertifizierbare Vertrauenswürdige Informatiksysteme



Kommentieren

- Verschiedene Zwecke und Zielgruppen
  - Entwickler der Klasse
    - Erklärung schwer verständlicher Codeabschnitte
    - Erklärung interner Implementierungsdetails

    → Interne Block-/Zeilenkommentare für Entwickler

  - Anwender der Klasse
    - Erklärung des Zwecks einer Klasse
    - Beschreibung des Verhaltens einer Methode

    → Öffentliche Javadoc für Anwender

Interne Block-/Zeilenkommentare

- einzeilig

```
// ein einzeiliger Kommentar
```

- mehrzeilig

```
/* Ein sehr langer Kommentar darf
   gerne auch über mehrere Zeilen gehen */
```

- Beispiel:

```
/* Dieser Konstruktor des zweiten Übungsblattes
verwendet die Datums-Konstanten der Klasse Terminal */
public Date() {
   ...
}
```

Javadoc

- http://java.sun.com/j2se/1.5/docs/guide/javadoc/index.html
- Javadoc ist ein Tool, das aus Quellcode automatisch die Dokumentation in HTML erzeugt
- Javadoc-Kommentare müssen daher speziell markiert werden und sollten in HTML geschrieben werden
- Aufbau eines Javadoc-Kommentars

```
/**
 * Kurzzusammenfassung in einem Satz.
 *
 * Aus&uuml;hrliche Beschreibung.
 *
 * Zus&auml;tzliche Informationen (Parameter, R&uuml;ckgabewert,
 * Querverweise, Autor, Version, ...
 */
```

Javadoc-Klassenkommentare

- Zweck der Klasse, angebotene Funktionalitäten und Abhängigkeiten
- Allgemeine Erläuterungen, die für alle Methoden gelten
- Tags, z.B. `author`, `version` ...
- **Keine** Implementierungsdetails / Geheimnisse der Klasse

```
/**
 * Diese Klasse repräsentiert einen Punkt in zwei Dimensionen.
 *
 * Hier folgt eine aus&uuml;hrliche Beschreibung.
 *
 *
 * @author Max Mustermann
 * @version 0.01
 */
public class Point { … }
```

Javadoc-Methodenkommentare

- Methodenzusammenfassung und Beschreibung
- `@param`-Tag für jeden Parameter: Parametername und Beschreibung des Parameters einschließlich der Vorbedingungen
- `@return`-Tag beschreibt den Rückgabewert, falls dieser ≠ `void`

```
/**
 * Verschiebt den Punkt um die angegebenen Werte.
 *
 * Diese Methode verschiebt den Punkt um den Wert dx in x-Richtung
 * und um den Wert dy in y-Richtung.
 *
 * @param dx Verschiebung in x-Richtung
 * @param dy Verschiebung in y-Richtung
 */
void shift(double dx, double dy) { … }
```

Erzeugen der Javadoc

- In der Kommandozeile/Eingabeaufforderung

  `javadoc [Optionen] *.java`

- Ausgewählte Optionen
  - `-public` nur `public` deklarierte Elemente werden gezeigt
  - `-private` alle Elemente werden gezeigt
  - `-d dir` die generierte Javadoc wird im Verzeichnis `dir` abgelegt
  - `-linksource` fügt HTML-Versionen der Java-Quelldateien hinzu

- Dokumentation aller Optionen

  `http://download.oracle.com/javase/1.5.0/docs/tooldocs/windows/javadoc.html#javadocoptions`

- Dokumentation ist nach Veröffentlichung der Klasse verbindlich

Kommentierungsregeln für Programmieren

- Javadoc für alle Attribute, Methoden und Klassen die nicht `private` sind
- Für `private` deklarierte Attribute und Methoden genügen interne Block-/Zeilenkommentare
- Implementierungsdetails sind für den Anwender irrelevant und gehören nicht in einen Javadoc-Kommentar
- Es ist durchaus erlaubt, eine Methode mit einem Javadoc und einem normalen Kommentar zu versehen
- Schreiben von Javadoc ab Übungsblatt 3

Richtig Kommentieren

- zu wenig Kommentare → schlecht
- zu viele Kommentare → schlecht
- falsche oder irreführende Kommentare → schlecht
  - sogar schlechter als gar keine Kommentare
  - z.B. bei nachträglichem Ändern des Codes
- Kommentare während des Programmierens schreiben
  - Nachträgliches Kommentieren wird ungenauer sein und noch weniger Spaß machen
- **Warum** wird etwas gemacht
  - **Nicht:** wie wird etwas gemacht
    → Dies dokumentiert bereits der Programmcode selbst

Richtig Kommentieren

- Kommentieren einer einzelnen Zeile
  - nur wenn wenn diese sehr kompliziert ist (normalerweise ist das nicht der Fall)
- Kommentare am Ende einer Programmcode-Zeile
  - bieten nur wenig Platz für aussagekräftige Kommentare
  - verführen dazu, den einfachen Programmcode einer einzelnen Zeile zu wiederholen
- Kommentieren von Code-Abschnitten
  - Beschreiben des **Zwecks** / der **Absicht** des Codes
  - **Keine** Duplikation des Programmcodes in Umgangssprache

Richtig Kommentieren

- Kommentieren von Deklarationen
  - nur falls notwendig oder sinnvoll
  - z.B. Angabe einer Maßeinheit (Meter, kg, Liter, kW, ...)
  - z.B. Angabe des validen Wertebereichs im jeweiligen Kontext
- Kommentieren von Kontrollstrukturen
  - bei `if`: **Warum** ist die Entscheidung notwendig?
  - bei Schleife: Was ist der **Zweck** der Schleife?
- Kommentieren von Methoden
  - Kurzbeschreibung des **Zwecks** der Methode in 1-2 Sätzen
  - Kommentieren der Parameter und des Rückgabewertes
  - Kommentieren der **Einschränkungen** der Methode
  - eventuelle weitere **Auswirkungen/Seiteneffekte** beschreiben

Richtig Kommentieren: Auf einen Blick

- Kommentiere nichts Offensichtliches
- Kommentiere Methoden
- Kommentiere keinen schlechten Code – schreibe ihn neu
- Widerspreche in den Kommentaren nicht dem Code
- Erläutere den Code
- Verwirre nicht

Gute Kommentare – Schlechte Kommentare

- Sind folgende Kommentare gut oder schlecht?

- ```
  zerocounter++;                    // erhöhe leeren Counter um 1
  ```

- ```
  int leistung;                     // in kW
  ```

- ```
  int sitzplaetze;                  // Anzahl: 1..9
  ```

- ```
  /* Durchlaufe alle Zahlen von 1 bis n und addiere
     diese jeweils zur Endsumme */
  int summe = 0;                    // setze summe = 0
  int i = 1;                        // initialisiere i mit 1
  while (i <= n) {                  // laufe von i bis n
     summe = summe + i;             // addiere i
     i++;                           // erhöhe i um 1
  }
  ```

Gute Kommentare – Schlechte Kommentare

- Sind folgende Kommentare gut oder schlecht?

```
/* Berechne die Summe der Zahlen von 1 bis n */
int summe = 0;
int i = 1;
while (i <= n) {
    summe = summe + i;
    i++;
}
```

```
/* Berechne die Summe der Zahlen von 1 bis n */
int summe = 0;
int i = 1;
while (i <= n) {
    summe = summe + 1;
    i++;
}
```

Gute Kommentare – Schlechte Kommentare

- Sind folgende Kommentare gut oder schlecht?

```
/* vertausche die Werte */
tmp = a;
a = b;
b = tmp;
```

```
// return true
return true;
```

```
/* bei negativem Rückgabewert liegt ein Fehler vor */
if (myMethod() < 0) {
   return false;
}
```