Programmierrichtlinien zur OOS

Im Folgenden werden einige Richtlinien für die Programmierung in Java aufgeführt, die bei der Bearbeitung der Übungen, Praktika und vor allem der Klausur zwingend einzuhalten sind.
Die angegebene Liste ist nicht vollständig und wird bei Einführung neuer Konstrukte ergänzt.

1. **Allgemeine Richtlinien**

   Programme nur soweit optimieren, dass sie lesbar bleiben.

2. **Kommentierung**
   - Jede Datei, die Quelltext enthält, muss kommentiert werden.
   - pro Datei einige einleitende Zeilen, die die Funktionalität des Quelltextes beschreiben.
   - vor jeder Klassendefinition Kommentar zur Funktionalität der Klasse
   - wenn aus Attributname nicht hervorgeht, was darin abgespeichert wird, muss dazu ein Kommentar in der Zeile vor dem Attribut verwendet werden.
   - vor jeder Methodenimplementierung Kommentar zur Funktionalität.
   - die Verwendung von `javadoc` ist zwingend vorgeschrieben, ab dem Zeitpunkt wo es in der Vorlesung vorgestellt wird.

3. **Formatierung**
   - genügend Freiraum für die Programmzeilen schaffen, die den Code deutlich hervorheben
   - gewöhnliche Leerzeichen anstelle von Tabulatoren verwenden, da unterschiedliche Editoren Tabulatoren unterschiedlich interpretieren.
   - Die *Zeilenlänge* sollte 78 Zeichen nicht überschreiten.
   - Die *Rümpfe* von Blöcken, Klassen, Methoden und Schleifen werden jeweils um 2 Spalten eingerückt.
   - Die *öffnende Klammer* steht in der übernächsten Spalte hinter dem Methodennamen bzw. der Schleifenbedingung.
   - Die *schließende Klammer* steht in der Zeile unmittelbar hinter dem letzten Befehl des Blockes und 2 Spalten links der ersten Spalte des Blockes.

Bsp.:
```
class EinKlassenName { …
    for (int i = 0; i < 10; i++) {
      System.out.println("i= " + i);
    }
}
```

- Jede Anweisung in extra Zeile.
- Bei Kontrollflussbefehlen `if`, `else`, `for`, `while` immer einen Block verwenden, auch wenn der Block nur aus einer oder keiner Anweisung besteht, da dadurch spätere Umprogrammierung nicht so schnell zu Fehlern führt.

Bsp.:
```
while ( Bedingung ) {
  // Empty !
}
```

- `else` immer unterhalb des zugehörigen `if`
- *Parameter* bei Methoden in eine Zeile, wenn es passt, sonst untereinander jeder in eine extra Zeile. Schließende Klammer hinter letzten Parameter

Bsp.:
```
int plus(int summand1, summand2) {
    ....
}
AdjustmentListener remove(AdjustmentListener l,
                          AdjustmentListener oldl) {
    ...
}
```

- Der *Punktoperator* für den Zugriff auf Objekt- oder Klassenvariablen bzw. -methoden, darf nicht durch Freiraum von dem Klassenname bzw. dem Ausdruck für die Objektreferenz oder dem Methoden- bzw. Variablennamen abgetrennt werden.

Bsp.: `persons.whoAmI.myName`

- `try_catch`-Klauseln sind wie folgt zu formatieren:

```
try {
  Anweisung;
  ...
} catch (Ausnahmetyp1 x) {
  Anweisung;
  ...
}
...
} catch (Ausnahmetypn x) {
  Anweisung;
  ...
}
```

4. **Bezeichner**

- *Klassennamen* werden mit einem Großbuchstaben begonnen.
- *Attribut-, Methoden- und Objektnamen* werden mit einem Kleinbuchstaben begonnen.
- Alle *Bezeichner* sind ein Wort ohne Verwendung von Sonderzeichen, wobei innere Wörter mit einem Großbuchstaben beginnen.

Bsp.:
```
EinKlassenName
einMethodenName()
```

- `static final`-Bezeichner bestehen nur aus Großbuchstaben
- Bezeichner von *Paketen* bestehen nur aus Kleinbuchstaben, auch für innere Wörter.

5. **Ausdrücke**

- Shift-Operatoren als Ersatz für arithmetische Operationen sind zu vermeiden.
- Möglichst Klammern setzen.
- Nur eine Zuweisung pro Zeile, also nicht:

Bsp.:
```
a = (b = c + d) + e;
```
stattdessen:
```
b = c + d;
a = b + e;
```

- Zahlenliterale sind in Ausdrücken zu vermeiden; stattdessen Konstante mit `final` definieren.

Bsp.:
```
final int ZEHN = 10;
```

Programmierrichtlinien zur OOS

6. **Anweisungen**
   - Zur *Formatierung* siehe Abschnitt 3.
   - Verwendung von `for`-Schleife immer dann, wenn eine Variable um eine konstante Größe erhöht (vermindert) wird und wenn vor Ausführung feststeht, wie oft sie durchlaufen wird.
   - Verwendung von `while`-Schleife, wenn eine solche Variable nicht vorliegt.
   - Verwendung von `do`/`while`-Schleife, wenn Abbruchbedingung erst nach einem Schleifendurchlauf ausgewertet werden kann.
   - möglichst Verzicht auf `continue`.
   - `break` immer dort einsetzen, wo benötigt.
   - bei `switch`/`case` aufpassen, dass man `breaks` verwendet.

7. **Klassen**
   - Gruppenweise in der Reihenfolge ihrer Sichtbarkeit programmieren:
   a) `public`
   b) `protected`
   c) paketsichtbar
   d) `private`
   - Innerhalb einer Gruppe folgende Reihenfolge:
   a) alle Attribute
   b) alle Konstruktoren
   c) alle Methoden

8. **Zugriffsrechte**
   - Zugriff auf `public`- und `protected`-Variablen vermeiden, da dies zu Schwierigkeiten führen kann.
   - möglichst nur lesend auf Variablen zugreifen
   - setter und getter einführen

9. **Codelänge und switches**
   - Programmcode von Methoden sollte maximal 2 Seiten lang sein.
   - lieber `switch` mit sehr vielen Fällen als massenweise geschachtelte `ifs`.