

### Blockbildung

```

1 // alle Variablen initialisieren
2 int recordIndex = 0;
3 int total = 0;
4 done = false;
5
26 while (recordIndex < recordCount) {
27     recordIndex++;
28
64 while (!done) {
65     if (total < protectedTotal) {
66         done = true;
67     }
68 }

```

- Zeilen mit Ähnlicher Aufgabe nahe bei einander
- Zeilen, die die selben Variablen verwenden nahe beieinander
- Initialisierungen der Variablen kurz vor ihrer Verwendung

### Verwendung von if

- wahrscheinlichster Fall (Normalfall) immer in if – Bedingung
- Sonderfälle im else-Zweig
- bei mehreren if: wahrscheinlichster Fall oben, unwahrscheinlichster Fall im letzten else-Zweig
- komplexe Bedingungen explizit machen und in Teilausdrücke zerlegen

#### Gegenbeispiel:

```

if ( done ) ; // leeres if
else { // Hier hier sind die eigentlichen Anweisungen
}

```

### Switch-Case

#### Ordnen der case –Fälle

- alphabetisch oder numerisch
- Normalfall an der Spitze, dann abnehmende Wichtigkeit

#### case-Fälle möglichst kurz

Gesamte switch / case Anweisung sollte möglichst auf eine Bildschirmseite passen

#### default

- sinnvolle Standardwerte
- zur Fehlerbehandlung

**Mehrere ähnliche switch-Blöcke im Programm ggf. durch Polymorphie ersetzen.**

### Zwischenergebnisse explizit machen

- Vermeidung des Nachschlagens und der Notizzettel ...
- Problem: Lange Aufrufketten (= Aufruf von Methoden auf Rückgabewerten von Methoden \*)

```

if (smodule.getDependSubsystems().contains(subSysMod.getSubSystem())) {
// Tuwas
}

```

- Besser: Teilausdrücke durch eigene Variable explizit benannt

### Zwischenergebnisse explizit machen (cont)

```

List moduleDependees = smodule.getDependSubsystems();
String ourSubSystem = subSysMod.getSubSystem();
if (moduleDependees.contains(ourSubSystem)) {
// tuwas
}

```

### Komplexe if-Ausdrücke zerlegen

#### Beispiel für schwere Lesbarkeit durch komplexe Teilausdrücke

```

if (( elementIndex < 0 ) || (MAX_ELEMENTS < elementIndex) || (elementIndex == lastElementIndex)) {
// tuwas }

```

#### Teilausdrücke jeweils über Variablen benennen

```

boolean finished = ( elementIndex < 0 ) || (
MAX_ELEMENTS < elementIndex);
boolean repeatedEntry = (elementIndex == lastElementIndex);
if (finished || repeatedEntry) { // tuwas }

```

### Vermeidung von if-Kaskaden

```

void do() {
if (cond1) {
if (cond2) {
...
if (condN) {
// Hier passiert
}
else ( alternativN )
}
else ( alternativ2 )
}
else ( alternativ1 )
}
} // if-Kaskade

void do() {
if (cond1) {
alternativ1; return;
}
if (cond2) {
alternativ2; return;
}
...
if (condN) {
alternativN; return;
}
// Hier passiert
} // Gegenvorschlag mit "Multiple Exit Points"

```

### Gute Methoden

- haben genau eine Aufgabe (Warnsignale: boolean Flags zur Steuerung einer Operation)
- Code auf derselben Abstraktionsebene haben (wenn nein -> Teile in eigene Operationen/Routinen/Methoden auslagern)

**Übergebene Parameter am Anfang der Methode/Operation/-Routine prüfen, bei Fehlern sofort zurückspringen**

#### Eventuell: Vergleichslogik überarbeiten

- Logische Bedingungen und if umdrehen (negieren)
- Fallunterscheidungen anders schneiden
- Bei gleichartigen Fällen: switch einführen
- Methode sollte nicht mehr als 50 Zeilen lang sein