	Kontrollstrukturen		AnPr
	Name	Klasse	Datum

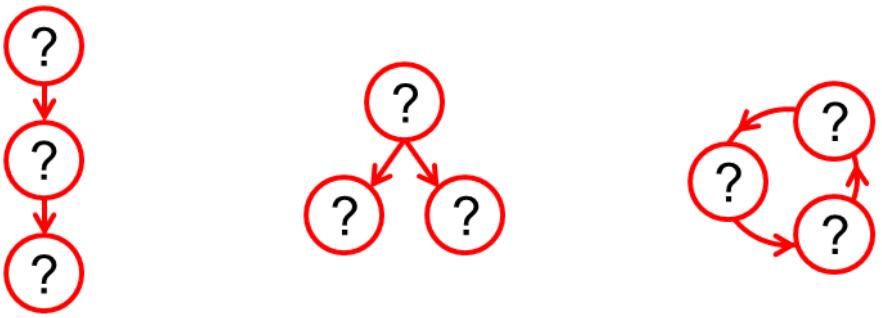
1 Allgemeines

In der Programmierung unterscheidet man prinzipiell zwischen:

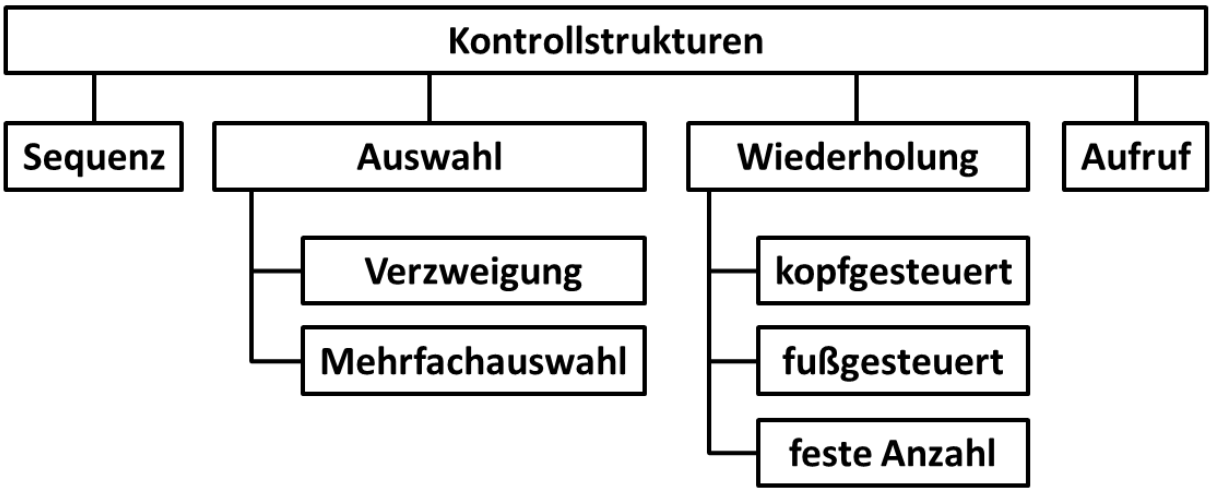
- Einfachen Anweisungen (Zuweisungen); z.B. Radius = Durchmesser /2
- Kontrollstrukturen

Kontrollstrukturen steuern die Ausführung von Anweisungen

Sie legen fest, in welcher Reihenfolge welche Aktion durchgeführt wird, ob verschiedene Verarbeitungszweige existieren und wenn ja, welcher Zweig wann verarbeitet werden soll, bzw. ob und wie oft Codeelemente wiederholt werden müssen:



Man kann also die verschiedenen Anweisungen wie folgt klassifizieren:



Im Folgenden wird die Semantik (Lehre von der Bedeutung der Wörter) dieser vier Kontrollstrukturen skizziert. Gleichzeitig werden die Kontrollstrukturen in zwei verschiedenen Notationen (Aufzeichnungen) angegeben:

Struktogramm-Notation

Programmmablaufplan-Notation

Isaac Nassi und Dr Ben Shneiderman führten 1972 eine Notationsform zur Beschreibung von Algorithmen ein, welche sich sehr nahe an den Möglichkeiten von modernen Programmiersprachen orientierte. Es wurde dabei bewusst auf Sprungmöglichkeiten verzichtet, da dies in Programmiersprachen wie bspw. C, aber auch Java aus Übersichtsgründen nicht mehr existieren. Dieser Diagrammtyp, welcher in DIN 66261 genormt ist, wird **Nassi-Shneidermann-Diagramm**, oder auch **Struktogramm** genannt.

Aufgrund der Einfachheit der Darstellungen, hat sich das Struktogramm als sehr geeignetes und am meisten genutztes Mittel zur übersichtlichen Dokumentation von Programmabläufen etabliert. Es gibt diverse kostenlose Programme, welche die Notation elektronisch ermöglichen, was durchaus eine Arbeitserleichterung im täglichen Leben darstellt (bspw. Strucotizer).

Eine weitere, oft genutzte Möglichkeit zur Dokumentation ist der Programm Ablauf Plan (PAP). Hier werden Symbole mittels Linien miteinander verbunden, was den Programmablauf darstellen soll. Da die Notation schnell zur Unübersichtlichkeit führt, wird sie in professionellen IT-Kreisen selten genutzt. Trotzdem werden manche Anfragen von nicht technischen Kunden gerne in dieser Notation dokumentiert. Genormt ist der PAP in DIN 66001. Aufgrund der Unübersichtlichkeit wird der PAP hier nur am Rande behandelt.

Grundsätzlich gilt jedoch für alle Notationen:

Notationen helfen bei der Strukturierung von Problemen und machen eine Lösung anschaulich.

Hier steht also das Problem samt Lösung im Vordergrund, nicht die eigentliche programmtechnische Umsetzung. Man könnte auch sagen, dass der **Algorithmus** gerne als Struktogramm notiert wird.

Die IHK Prüfer gehen davon aus, dass Sie in der Lage sind, Struktogramme zu lesen und aufzuschreiben. Vor allem das notieren von Struktogrammen ist für manche Schüler schwierig zu bewerkstelligen. Insofern hier ein paar Hinweise:

- Überlegen Sie sich einen groben Lösungsweg, bevor Sie den Stift in die Hand nehmen.
- Da Struktogramme beliebig ineinander verschachtelt werden können, ist es wichtig, den Überblick nie zu verlieren!
- Skizzieren Sie Lösung ganz grob auf einem Schmierzettel – es ist fast unmöglich auf Anhieb das Struktogramm richtig zu zeichnen!
- Lassen Sie beim Zeichnen genug Platz, für eventuelle Ergänzungen.
- Übertragen Sie das Struktogramm ins Reine.
- Nutzen Sie ein Lineal.

Ansonsten kann nur empfohlen werden das Thema einzuüben. Unter www.codeconcert.de finden Sie eine kostenfreie Software „Structomat“, welche Ihnen das Üben erleichtern soll.

2 Struktogrammelemente

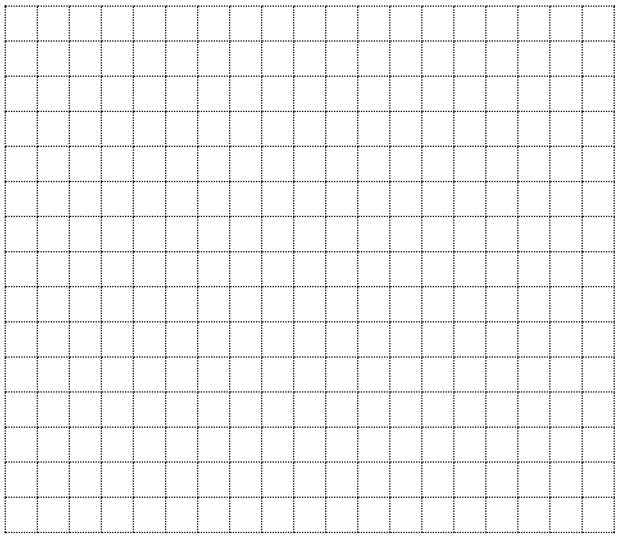
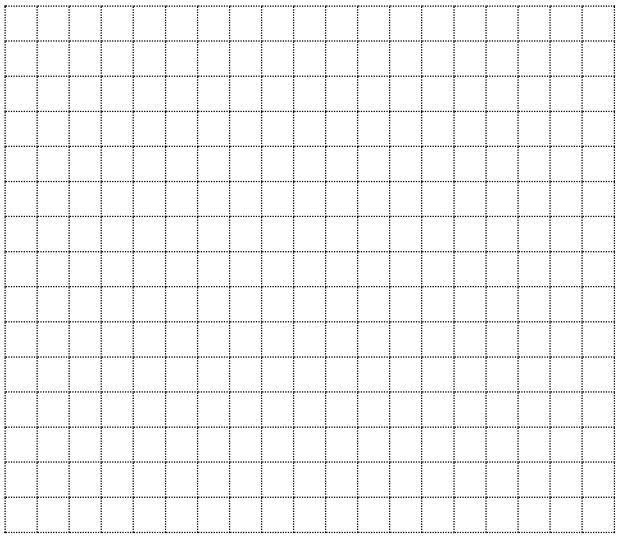
Im Anschluss werden nun die wichtigsten Struktogrammelemente erklärt und der passende Java Code ergänzt. Der Vollständigkeit halber werden zusätzlich noch die wichtigsten PAP Elemente aufgezeigt.

Die Grafiken werden im Rahmen des Unterrichts erklärt. Schreiben Sie diese bitte sauber mit. Sollten Sie aufgrund von Krankheit die Grafiken nicht mitschreiben, versuchen Sie über das Internet die richtigen Einträge (mit Bleistift) zu ergänzen. Dies kann im Anschluss mit der Lehrkraft korrigiert werden.

2.1 Sequenz

Erfordert eine Problemlösung, dass mehrere Anweisungen hintereinander ausgeführt werden, dann formuliert man eine Sequenz (Aneinanderreihung).

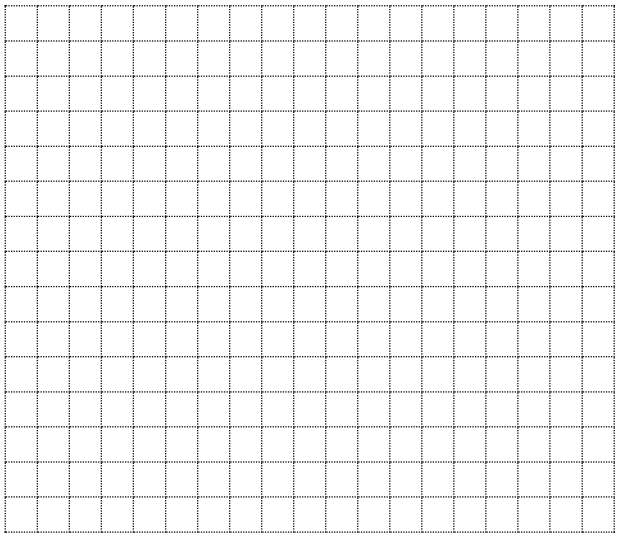
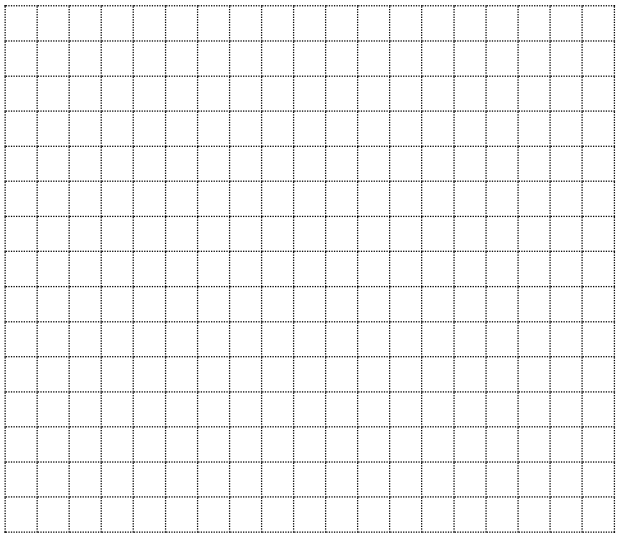
Bei einer Sequenz erfolgt die Abarbeitung immer von oben nach unten.

PAP	Struktogramm
	

Auf den Java Code wird hier verzichtet, da die Sequenz jeden beliebigen Java Code beinhalten kann.

2.2 Die (zweiseitige) Auswahl, oder auch Verzweigung

Sollen Anweisungen nur in Abhängigkeit von bestimmten Bedingungen ausgeführt werden, dann verwendet man das Konzept der **Auswahl**, auch **Verzweigung** oder **Fallunterscheidung** genannt. Durch die Bedingung wird eine Auswahl der auszuführenden Anweisungen vorgenommen. Ist die Bedingung erfüllt bzw. wahr, dann werden die Ja-Anweisungen ausgeführt, sonst im falsch Fall die Nein-Anweisungen.

PAP	Struktogramm
	

Der Bedingungsbereich einer Verzweigung ist entweder „Wahr“ oder „Falsch“.

Insofern müssen dort Elemente stehen, welche einen derartigen Zustand annehmen können. Dies sind:

- Vergleiche (bspw. `a < b`)
- Boolean Variablen
- Methoden, welche Wahr oder Fasch zurückgeben (bspw. `sName.equals("Paul")`)
- Logische Verknüpfungen der oben genannten Elemente
  (bspw. (a<b)&&(`sName.equals("Paul")`)

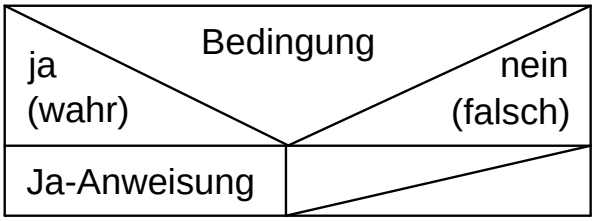
Der Java Code sieht wie folgt aus:

Java allgemein:	**Java Beispiel:**

2.3 Einseitige Auswahl, oder auch einseitige Verzweigung

Es kommt recht häufig vor, dass die Sequenz im Falschzweig der zweiseitigen Auswahl leer ist. Diesen Sonderfall bezeichnet man als einseitige Auswahl.

Struktogramm:



Java allgemein

```
if (Bedingung)
{
    Ja-Anweisung;
}
```

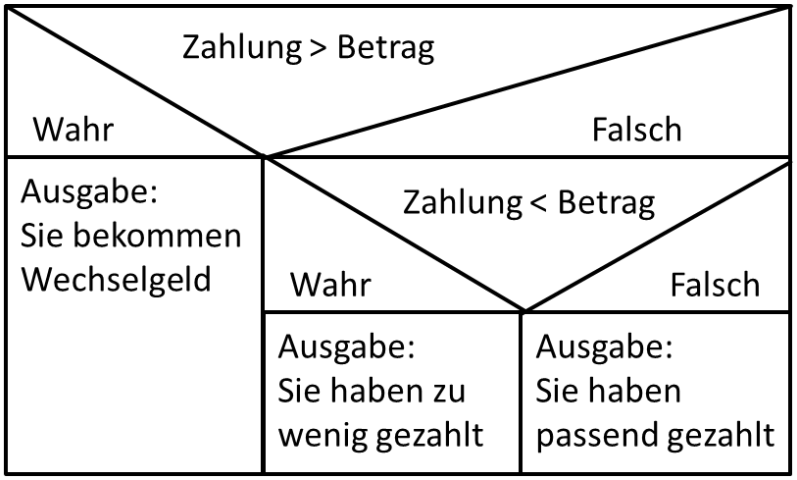
Java Beispiel:

```
if (a == 0)
{
    sAusgabe = "keine Division";
}
```

2.4 Mehrstufige Verzweigung

Oft muss eine bedingte Anweisung wieder in Fälle unterteilt werden. Dann liegt eine mehrstufige Verzweigung vor.

Struktogramm:



Java allgemein

```
if (Bedingung1)
{
    Ja-Anweisung1;
}
else
{
     if (Bedingung2)
     {
        Ja-Anweisung2;
     }
     else
     {
        Nein-Anweisung;
     }
}
```

Java Beispiel:

```
if (dZahlung > Betrag)
{
     sAusgabe = "Sie bekommen Wechselgeld";
}
else
{
     if (dZahlung < Betrag)
     {
          sAusgabe = "Sie haben zu wenig gezahlt";
     }
     else
     {
          sAusgabe = "Sie haben passend gezahlt";
     }
}
```

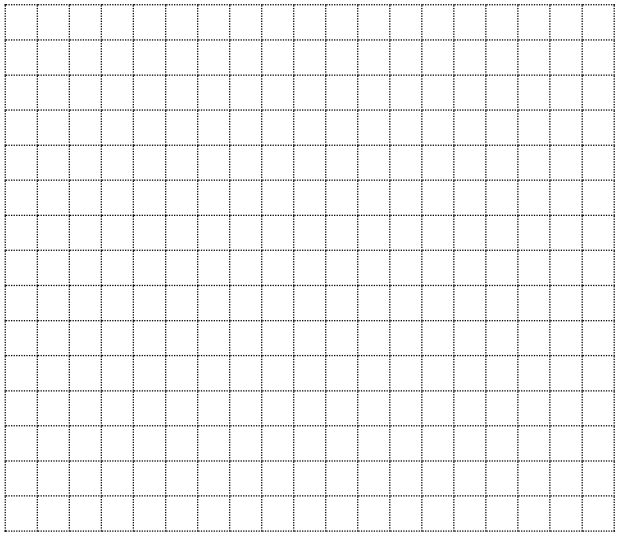
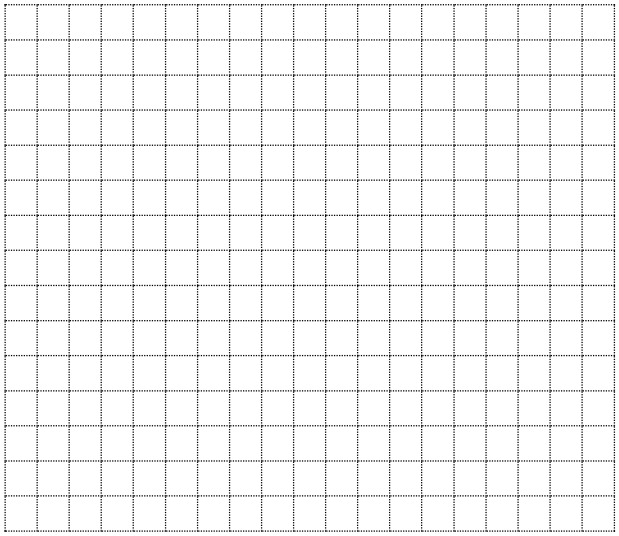
Alternativ kann man auch das verkürzte else if Statement verwenden:

Java Beispiel:

```
if (dZahlung > Betrag)
{
     sAusgabe = "Sie bekommen Wechselgeld";
}
else if (dZahlung < Betrag)
{
     sAusgabe = "Sie haben zu wenig gezahlt";
}
else
{
     sAusgabe = "Sie haben passend gezahlt";
}
```

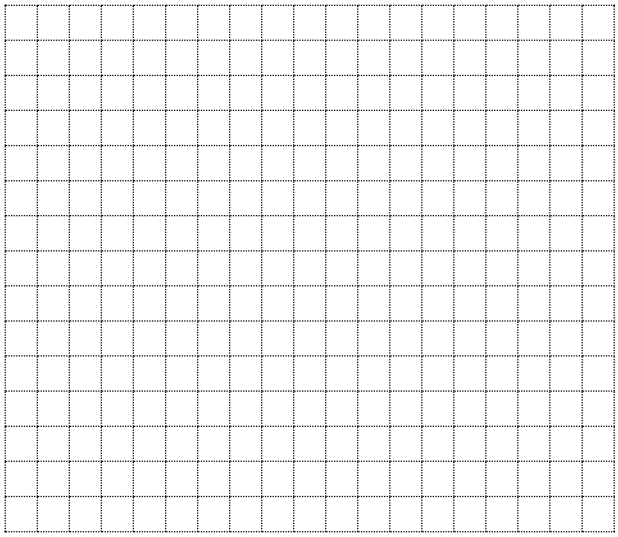
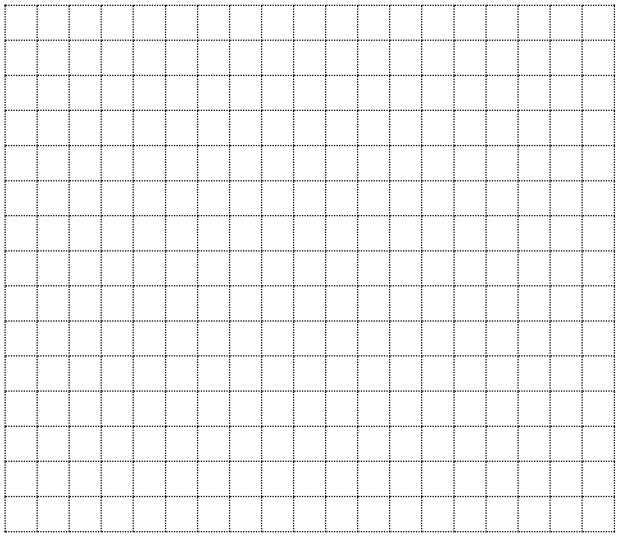
2.5 Mehrfachauswahl

Mehrfachauswahlen ermöglichen eine übersichtliche Zuordnung von Variablenwerten zu Code. Abhängig vom Inhalt **einer Variablen** kann das Programm eine von mehreren Möglichkeiten auswählen.

PAP	**Struktogramm**
	

Bis einschließlich Java 6 war es ausschließlich möglich, als Prüfvariable in der switch Anweisung ganzzahlige Datentypen (wie int oder byte) oder Charactervariablen zu nutzen. Ab Java 7 ist es nun auch möglich, Strings in den Switch einzubauen. Sollten Sie sich dafür entscheiden, so ist Ihr Code jedoch nicht mehr abwärtskompatibel. Die Zuordnung der verschiedenen Ausprägungen mit dem Code kann auch als verschachtelte Verzweigung umgesetzt werden.

Der Java Code sieht wie folgt aus:

Java allgemein:	**Java Beispiel:**
	

Eine Mehrfachauswahl lässt sich immer als verschachtelte Verzweigung darstellen.

2.6 Die Wiederholung (Schleifen)

Wenn eine oder mehrere Anweisungen wiederholt werden sollen, müssen sogenannte Schleifen eingesetzt werden. Jede Schleife benötigt eine Abbruchbedingung (alternativ eine Laufbedingung), da sonst eine Endlosschleife erzeugt werden würde. Es gibt zwar im Microcontrollerbereich Anwendungsfälle für Endlosschleifen, in der Programmierung von PCs oder Servern sind Endlosschleifen jedoch nicht sinnvoll einsetzbar.
Bei den Schleifen (oder Wiederholungsschleifen) unterscheidet man drei verschiedene Konstrukte:

- Kopfgesteuerte Schleife (Abfrage vor jedem Durchlauf)
- Fußgesteuerte Schleife (Abfrage nach jedem Durchlauf)
- Zählschleife (Wiederholung mit fester Wiederholanzahl)

Schleifen mit Abfragen sind bedingte Wiederholschleifen. Diese werden in vielen Programmiersprachen mit dem Schlüsselwort „do/while“, also „tue, solange“ definiert. In einigen Programmiersprachen wird das Schlüsselwort „Loop“ für „Schleife“ verwendet.

2.6.1 Kopfgesteuerte Schleife

Kopfgesteuerte Schleifen werden immer dann verwendet, wenn die Aktion nur bei einer sinnvollen Bedingung durchgeführt werden soll. Man prüft zuerst und führt dann aus (Zuerst prüft man, ob man noch etwas im Glas hat und trinkt dann. Nicht umgekehrt. Das wird wiederholt, bis das Glas leer ist.).

PAP	**Struktogramm**

Wie Sie sehen, ist das Struktogramm streng normiert aufgebaut, wohingegen der PAP mit mehr oder weniger beliebig eintragbaren Pfeilen der Sprung zurück realisiert. Weiterhin erkennt man, dass das PAP lediglich ein Zusammenschluss von mehreren einfacheren Elementen ist.

Auch ist zu beachten ist, dass in Java die Bedingung eine „Weiterlaufbedingung“ ist, keine Abbruchbedingung. Es wird also die Schleife solange wiederholt, bis die Weiterlaufbedingung **nicht mehr erfüllt** ist. Es gibt allerdings Programmiersprachen, in denen man auch Schleifen realisieren kann, bei denen eine Abbruchbedingung vorgesehen ist. In Java kann dies durch eine einfache Negation der Bedingung erreicht werden.

Machen Sie im Struktogramm kenntlich, ob es sich um eine Abbruch- oder Weiterlaufbedingung handelt.

Java allgemein:	Java Beispiel:

2.6.2 Fußgesteuerte Schleife

Fußgesteuerte Schleifen werden immer dann verwendet, wenn das Ergebnis einer Aktion geprüft wird, und diese Aktion in einer definierten Situation wiederholt werden muss. Man führt zuerst aus und prüft dann (Zuerst gibt man bspw. einen Login ein und prüft danach, ob er sinnvoll war. Wenn nicht, muss der Login nochmal eingegeben werden.).

PAP	Struktogramm

Vom Prinzip her gilt für die Fußgesteuerte Schleife das gleiche, wie für die kopfgesteuerte – sie akzeptiert in Java nur Weiterlaufbedingungen. Auf einen wichtigen Unterschied muss hier jedoch nochmal eingegangen werden:

Fußgesteuerte Schleifen werden auf jeden Fall mindestens einmal ausgeführt!

Das ist wichtig zu verstehen, da man hier gegebenenfalls in fehlerhafte Zustände gelangen kann.

Java allgemein:	**Java Beispiel:**

2.6.3 Zählschleifen

Wiederholungen mit fest definierter Wiederholungsanzahl nennt man Zählschleifen. Solche Schleifen werden oft zum Auslesen von Arrays (also Listen) verwendet. Da sich die Schleife den aktuellen Zählerstand merken muss, ist eine Zählvariable notwendig.

PAP	**Struktogramm**

Bei Zählschleifen gibt man üblicherweise folgende Punkte an:

- Name der Zählvariablen
- Startwert
- Endwert
- Schrittweite und Schrittrichtung

Zählschleifen sind im Prinzip nichts anderes als kopfgesteuerte Wiederholschleifen mit einem Zähler versehen. Die PAP Notation macht dies auch deutlich!

Java allgemein:	Java Beispiel:

2.6.4 Expliziter Schleifenabbruch

Schleifen können auch explizit abgebrochen werden. Hierzu dient in Java die „break“ Anweisung. Sobald der Interpreter die break Anweisung passiert, wird die umschließende Schleife unterbrochen. Es wird jedoch empfohlen, mit dem expliziten Abbruch (oder dem „Aussprung“) sparsam umzugehen, da er den Code unübersichtlicher machen kann.

Struktogramm:	Java allgemein

	`if (Bedingung)`
	`{`
	`  break;`
	`}`

	Java Beispiel:

	`while (a > 10)`
	`{`
	`  if (c == 0)`
	`  {`
	`  break;`
	`  }`
	`}`

2.6.5 Aufruf / Unterverarbeitung

Da Struktogramme schnell sehr Umfangreich werden können, nutzt man gerne Unterverarbeitungen. Hierbei werden abgeschlossene Bereiche in einem eigenen Struktogramm beschrieben, welche mittels eines Aufrufs angesprochen werden. Dies dient ausschließlich zur Steigerung der Übersichtlichkeit. Der Aufruf ist nicht Teil der DIN Norm.

3 Lizenz