Einführung in die Programmierung

by André Karge

Übung - Verzweigungen und Schleifen

letzte Woche

- Primitives
- Command Line I/O
- Operatoren
- Casting
- Methoden

diese Woche

- Besprechung Übungsblatt 01
- Verzweigungen
- Schleifen
- Übungsblatt 02

Übungsblatt 01

Übungsblatt 01

Aufgabe 01

Beispiel an der Tafel

Übungsblatt 01

Aufgabe 01

Beispiel an der Tafel

Aufgabe 02

Eclipse Code

Übungsblatt 01

Aufgabe 01

Beispiel an der Tafel

Aufgabe 02

Eclipse Code

Aufgabe 03

Vorrechnung an der Tafel

Übungsblatt 01

Aufgabe 01

Beispiel an der Tafel

Aufgabe 02

Eclipse Code

Aufgabe 03

Vorrechnung an der Tafel

Richtigstellung -132 in 8bit

Die Zahl -132 ist natürlich nicht mit 8bit inklusive Vorzeichenbit darstellbar.
Der Versuch diese Zahl mit nur 7bit darzustellen ist daher nicht möglich.
Deswegen fließt die Antwort von Aufgabe 3.a.iii) nicht mit in die Bewertung ein.

Verzweigungen

Verzweigung - Insel 2.5

Definition

Kontrollstrukturen um Teile von Programmen nur unter bestimmten Bedingungen auszuführen.
Konstrukte:

- if Verzweigung
- if-else Verzweigung
- switch-case Verzweigung

Verzweigungen - *if*-Verzweigung - Insel 2.5.1

Generelle if-Verzweigung

```
if ( Bedingung ) {
  // Befehls-Block, der beim Erfüllen der Bedingung ausgeführt wird
}
else { // else + Block ist Optinal
  // Befehls-Block, der beim Nichterfüllen der Bedingung ausgeführt wird
}
```

Beispiel

```
if (true) {
  System.out.println("Verzweigung erreicht");
}

if (false) { // Bedingung ist false -> if(Bedingung) wird nur ausgeführt, wenn Bedingung true ist
  System.out.println("Wird nicht erreicht");
}
else { // Bedingung ist false -> else wird nur ausgeführt, wenn Bedingung false ist
  System.out.println("Wird erreicht");
}
```

Verzweigungen - *if*-Verzweigung - Insel 2.5.1

Nutzung von Bedingungen

- Hier kommen die logischen Operatoren in Benutzung

```
// Gibt die Differenz zwischen zwei Zahlen aus
int getDifference(int a, int b) {
  // Wenn b größer oder gleich a -> ziehe linke Zahl von rechter ab
  if (a <= b) {
    return b - a;
  }
  // Ansonsten ziehe rechte Zahl von linker ab
  else {
    return a - b;
  }
}
```

Zusammengesetzte Bedingung

```
// logisches UND
if (Bedingung1 & Bedingung2) {
  // Block wird nur ausgeführt, wenn beide Bedingungen erfüllt sind
}
// logisches ODER
if (Bedingung1 | Bedingung2) {
  // Block wird ausgeführt, wenn eine der Bedingungen erfüllt ist
}

// Verschachtelung von Bedingungen
if (Bedingung1 & (Bedingung2 | (Bedingung3 & Bedingung4)) & Bedingung5)

// Beispiel:
int a = 15;
if (3 < 5 & (15 >= a | (2 == 2 & 5 == 4)) & 42 > a) {
  System.out.println("Bedingung erfüllt!");
}
else {
  System.out.println("Bedingung nicht erfüllt!");
}
```

Zusammengesetzte Bedingung

```
// logisches UND
if (Bedingung1 & Bedingung2) {
  // Block wird nur ausgeführt, wenn beide Bedingungen erfüllt sind
}
// logisches ODER
if (Bedingung1 | Bedingung2) {
  // Block wird ausgeführt, wenn eine der Bedingungen erfüllt ist
}

// Verschachtelung von Bedingungen
if (Bedingung1 & (Bedingung2 | (Bedingung3 & Bedingung4)) & Bedingung5)

// Beispiel:
int a = 15;
if (3 < 5 & (15 >= a | (2 == 2 & 5 == 4)) & 42 > a) {
  System.out.println("Bedingung erfüllt!");
}
else {
  System.out.println("Bedingung nicht erfüllt!");
}
```

Frage: *Wird die Bedingung erfüllt oder nicht?*

Zusammengesetzte Bedingung

```
// logisches UND
if (Bedingung1 & Bedingung2) {
  // Block wird nur ausgeführt, wenn beide Bedingungen erfüllt sind
}
// logisches ODER
if (Bedingung1 | Bedingung2) {
  // Block wird ausgeführt, wenn eine der Bedingungen erfüllt ist
}

// Verschachtelung von Bedingungen
if (Bedingung1 & (Bedingung2 | (Bedingung3 & Bedingung4)) & Bedingung5)

// Beispiel:
int a = 15;
if (3 < 5 & (15 >= a | (2 == 2 & 5 == 4)) & 42 > a) {
  System.out.println("Bedingung erfüllt!");
}
else {
  System.out.println("Bedingung nicht erfüllt!");
}
```

Frage: *Wird die Bedingung erfüllt oder nicht?*

Antwort: *(true & (true | (true & false)) & true) = true*

Verzweigungen - Mehrfachverzweigung - Insel 2.5.2

- Wenn man mit einer Verzweigung mehr als nur 2 unterschiedliche Fälle abdecken möchte
- Bspw.: Annahme von 3 Fällen (Kind, Jugendlicher, Erwachsener) → jenachdem, welches Alter ein Nutzer hat, soll eine von 3 Ausgaben gezeigt werden

```
void zeigeAusgabe(int alter) {
  if (alter < 10) {
    System.out.println("Die Legokiste ist hinten rechts");
  }
  else if (alter >= 10 && alter < 18) {
    System.out.println("Die Playstation ist hinten links");
  }
  else { // nur noch Zahlen >= 18 übrig, daher müssen wir das nicht mehr angeben
    System.out.println("Das Bier ist im Kühlschrank");
  }
}
```

Bedingungsvarianten

- Bedingungen, die durch einfaches "&"getrennt sind, werden alle der Reihe nach geprüft und ihr Wahrheitswert berechnet
- Bedingungen, die durch doppeltes "&&"getrennt sind, werden so lange der Reihe nach geprüft, bis ein Ausdruck false ist. Alle folgenden Ausdrücke werden ignoriert
- selbiges gilt auch für den ODER Operator | (| vs. ||)

Verzweigungen - Switch-Case - Insel 2.5.4

Switch-Case Verzweigung löst das Problem von zu vielen Mehrfachverzweigungen.

```
switch ( meineVariable ) {
  case 1:
    // Ausführung wenn meineVariable == 1
    System.out.println("first case");
    break;
  case 2:
    // Ausführung wenn meineVariable == 2:
    System.out.println("first case");
    break;
  default:
    System.out.println("wenn weder case 1 noch case 2 eintreffen");
}
```

Verzweigungen - Switch-Case - Insel 2.5.4

Switch-Case Verzweigung löst das Problem von zu vielen Mehrfachverzweigungen.

```
switch ( meineVariable ) {
  case 1:
    // Ausführung wenn meineVariable == 1
    System.out.println("first case");
    break;
  case 2:
    // Ausführung wenn meineVariable == 2:
    System.out.println("first case");
    break;
  default:
    System.out.println("wenn weder case 1 noch case 2 eintreffen");
}
```

Wichtig: **break;** am Ende jedes Cases verhindert Ausführung von mehreren Cases

Schleifen

Schleifen - Insel 2.6

Definition

Eine Schleife ist ein Programmkonstrukt, das dazu dient bestimmte Anweisungen mehrfach abzuarbeiten.
Sie besteht aus einer **Schleifenbedingung** und einem **Schleifenkörper**.
Konstrukte:

- while-Schleife
- do-while-Schleife
- for-Schleife

While Schleife - Insel 2.6.1

- While = Abweisende Schleife
- Schleifenbedingung muss vor erstem Abarbeiten erfüllt sein
- vor jedem Schleifendurchlauf wird die Bedingung geprüft

```
int i = 0;
while (i < 10) { // Prüfen der Bedingung bei jedem Ausführen
  System.out.println("aktueller Durchlauf: " + i);
  i++;
  // Springe wieder in die Zeile der Bedingung
}
```

While Schleife - Insel 2.6.1

- While = Abweisende Schleife
- Schleifenbedingung muss vor erstem Abarbeiten erfüllt sein
- vor jedem Schleifendurchlauf wird die Bedingung geprüft

```
int i = 0;
while (i < 10) { // Prüfen der Bedingung bei jedem Ausführen
  System.out.println("aktueller Durchlauf: " + i);
  i++;
  // Springe wieder in die Zeile der Bedingung
}
```

Frage: *Wie häufig wird die Schleife durchlaufen? Was ist der Output?*

While Schleife - Insel 2.6.1

- While = Abweisende Schleife
- Schleifenbedingung muss vor erstem Abarbeiten erfüllt sein
- vor jedem Schleifendurchlauf wird die Bedingung geprüft

```
int i = 0;
while (i < 10) { // Prüfen der Bedingung bei jedem Ausführen
  System.out.println("aktueller Durchlauf: " + i);
  i++;
  // Springe wieder in die Zeile der Bedingung
}
```

Frage: *Wie häufig wird die Schleife durchlaufen? Was ist der Output?*
Antwort: *10x - Zahlen von 0 - 9*

While Schleife - Insel 2.6.1

```java
int i = 42;
while (i < 10) { // Prüfen der Bedingung bei jedem Ausführen
  System.out.println("aktueller Durchlauf: " + i);
  i++;
  // Springe wieder in die Zeile der Bedingung
}
```

While Schleife - Insel 2.6.1

```java
int i = 42;
while (i < 10) { // Prüfen der Bedingung bei jedem Ausführen
  System.out.println("aktueller Durchlauf: " + i);
  i++;
  // Springe wieder in die Zeile der Bedingung
}
```

Frage: *Wie häufig wird die Schleife durchlaufen? Was ist der Output?*

While Schleife - Insel 2.6.1

```
int i = 42;
while (i < 10) { // Prüfen der Bedingung bei jedem Ausführen
  System.out.println("aktueller Durchlauf: " + i);
  i++;
  // Springe wieder in die Zeile der Bedingung
}
```

Frage: *Wie häufig wird die Schleife durchlaufen? Was ist der Output?*
Antwort: *0x - kein Output*

Do-While Schleife - Insel 2.6.2

- Do-While = Annehmende Schleife
- Schleifenbedingung muss am Ende des Blocks erfüllt sein um ihn erneut auszuführen
- wird immer mindestens einmal ausgeführt

```
int i = 0;
do {
  System.out.println("aktueller Durchlauf: " + i);
  i++
} while(i < 10);
```

Do-While Schleife - Insel 2.6.2

- Do-While = Annehmende Schleife
- Schleifenbedingung muss am Ende des Blocks erfüllt sein um ihn erneut auszuführen
- wird immer mindestens einmal ausgeführt

```
int i = 0;
do {
  System.out.println("aktueller Durchlauf: " + i);
  i++
} while(i < 10);
```

Frage: *Wie häufig wird die Schleife durchlaufen? Was ist der Output?*

Do-While Schleife - Insel 2.6.2

- Do-While = Annehmende Schleife
- Schleifenbedingung muss am Ende des Blocks erfüllt sein um ihn erneut auszuführen
- wird immer mindestens einmal ausgeführt

```
int i = 0;
do {
  System.out.println("aktueller Durchlauf: " + i);
  i++
} while(i < 10);
```

Frage: *Wie häufig wird die Schleife durchlaufen? Was ist der Output?*
Antwort: *10x - Zahlen von 0-9*

Do-While Schleife - Insel 2.6.2

```java
int i = 42;
do {
  System.out.println("aktueller Durchlauf: " + i);
  i++
} while(i < 10);
```

Do-While Schleife - Insel 2.6.2

```
int i = 42;
do {
  System.out.println("aktueller Durchlauf: " + i);
  i++
} while(i < 10);
```

Frage: *Wie häufig wird die Schleife durchlaufen? Was ist der Output?*

Do-While Schleife - Insel 2.6.2

```
int i = 42;
do {
  System.out.println("aktueller Durchlauf: " + i);
  i++
} while(i < 10);
```

Frage: *Wie häufig wird die Schleife durchlaufen? Was ist der Output?*
Antwort: *1x - Zahl 42*

For Schleife - Insel 2.6.3

- Spezielle Variante der While Schleife
- For Schleife = Abweisende Schleife
- Hat einen komplexeren Schleifenkopf als die While Schleife

Schleifenkopf

```
for (Initialisierung; Schleifenbedingung; Schleifen-Inkrement) {...}
```

- Initialisierung: Schleifenvariable (vgl. *int i = 0;* der while und der do-while Schleife)
- Schleifenbedingung: Bedingung für Eintritt und erneutes Ausführen der Schleife (vgl. *(i< 10)* der while und der do-while Schleife)
- Schleifen-Inkrement: Wie soll die Schleifenvariable am Ende jedes Durchlaufes verändert werden (vgl. *i++;* der while und der do-while Schleife)

Beispiel For Schleife

```
for (int i=0; i<10; i++) {
  System.out.println("aktueller Durchlauf: " + i);
}
```

Selbes Beispiel als While Schleife

```
int i = 0;
while(i<10) {
  System.out.println("aktueller Durchlauf: " + i);
  i++;
}
```

Wir sparen 2 Zeilen mit einem For Loop

Programmierbeispiel

Aufgabe

Schreiben Sie eine Methode, mit der Sie Pyramiden auf der Konsole ausgeben können.
Die länge der Grundfläche soll per Parameter regelbar sein.
Zudem soll ein weiterer Parameter steuern, ob die Pyramide auf dem Kopf steht oder nicht.
Beispiel:

```
printPyramid(5, false);
>    #
>   ###
>  #####

printPyramid(5, true);
>  #####
>   ###
>    #
```

Fragen?