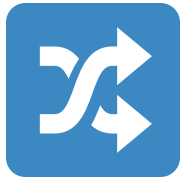
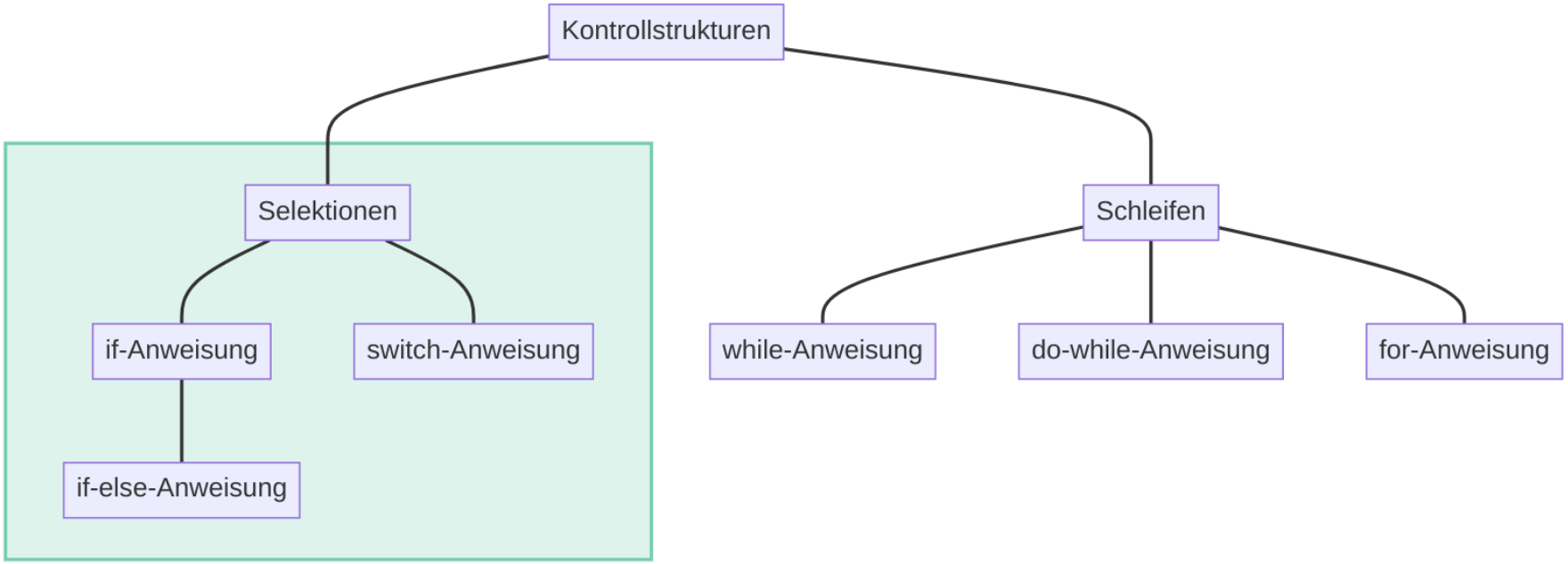
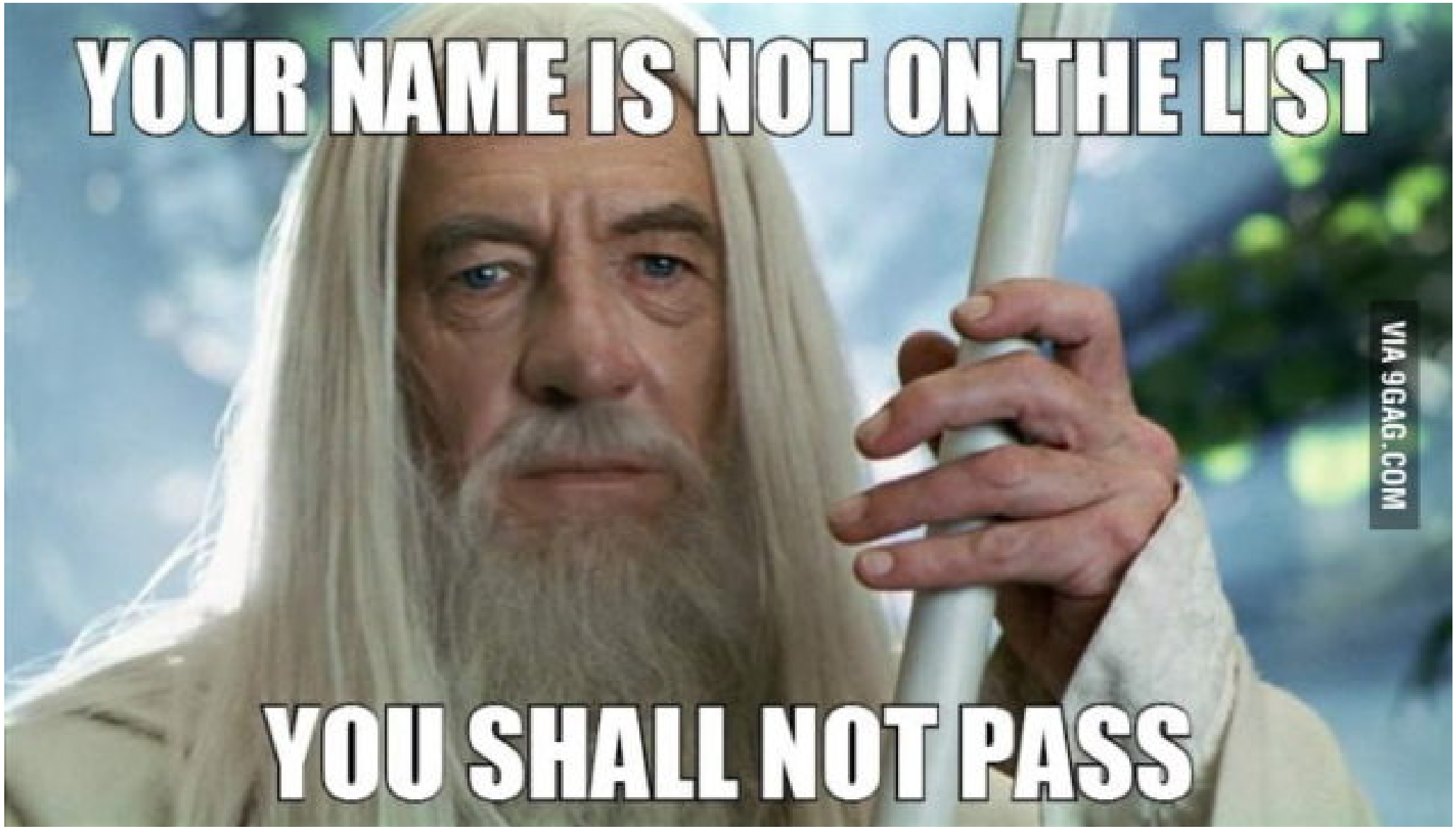


Selektion

Kontrollstrukturen

Vergleichs- und logische Operatoren kommen häufig dann zum Einsatz, wenn man etwas nur unter einer **bestimmten Bedingung** ausführen soll.




YOUR NAME IS NOT ON THE LIST
YOU SHALL NOT PASS
VIA 9GAG.COM

⚖️ Bedingungen | *Vergleichsoperatoren*

Gegeben: `int a = 3; int b = 3;`

Operator	Beschreibung	Beispiel	Resultat
`==`	überprüft auf Gleichheit	`a == b`	`true`
`!=`	überprüft auf Ungleichheit	`a != b`	`false`
`>`	ist linker Operand grösser	`a > b`	`false`
`>=`	ist linker Operand grösser oder gleich	`a >= b`	`true`
`<`	ist linker Operand kleiner	`a < b`	`false`
`<=`	ist linker Operand kleiner oder gleich	`a <= b`	`true`

* nur bei primitiven Datentypen. Nicht bei `String`!

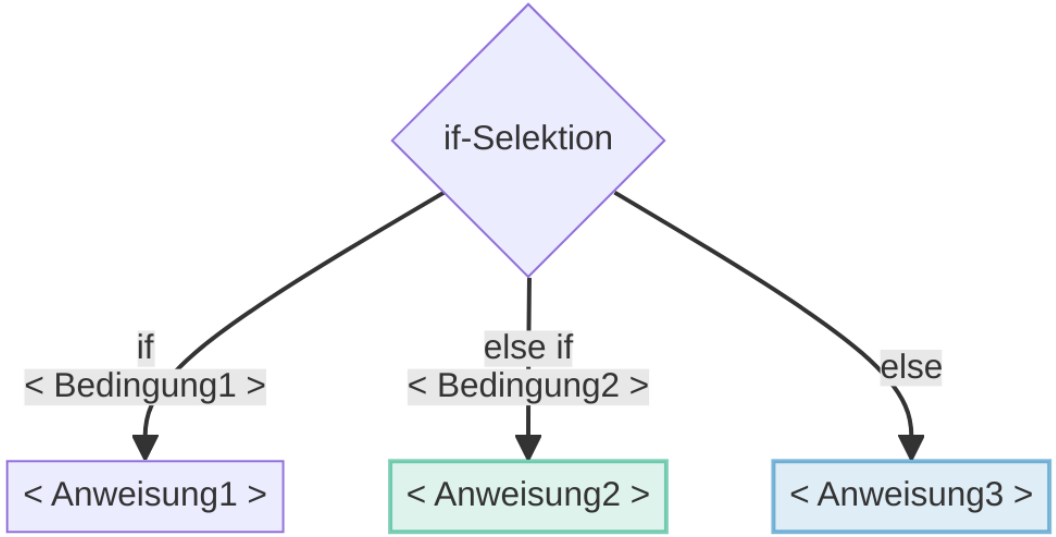
🥘 Bedingungen kombinieren

Gegeben: `boolean a = true; boolean b = false;`

Operator	Beschreibung	Beispiel	Resultat
`&&`	UND: **beide** Ausdrücke sind `true`	`a && b`	`false`
`||`	ODER: **mindistens ein** Ausdruck ist `true`	`a || b`	`true`
`^`	XOR: **genau einer** der Ausdrücke ist `true`	`a ^ b`	`true`
`!`	NOT: wandelt ein `boolean` ins Gegenteil um	`!b`	`true`

if / else if / else

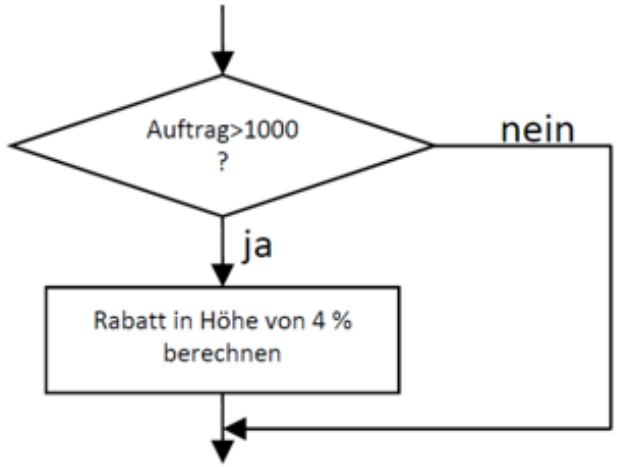
- Wird dafür verwendet, **Bedingungen zu überprüfen**
- Als Bedingung dient ein **Bool'scher Wert** (`true` , `false`), welche über ein **Vergleichsoperator** erzeugt wird.
- Kontrolliert ob ein Codeabschnitt durchlaufen wird



if / else if / else

Beispiel

Wenn, `if`, ein Kunde einen Auftrag über 1000.-- erteilt, bekommt er 4 % Rabatt.



In Java

```java
double price = StdInput.readDouble();

if (price > 1000) {
    price *= 0.96;
}

System.out.println("Your price " + price);
```

- Bedingung: `price > 1000`
  - Operator: `>` grösser als
- Anweisung: `price *= 0.96`
  - Oder: `price = price * 0.96`


65
Bild: www.tf-ausbildung.de

if / else if / else

Mit `else if` kann priorisiert auf weitere Bedingungen reagiert werden

Schema

```
if (<Bedingung1>) {
    <Anweisung1>
} else if (<Bedingung2>) { // Optionaler Block
    <Anweisung2>
} else {                   // Optionaler Block
    <Anweisung3>
}
```

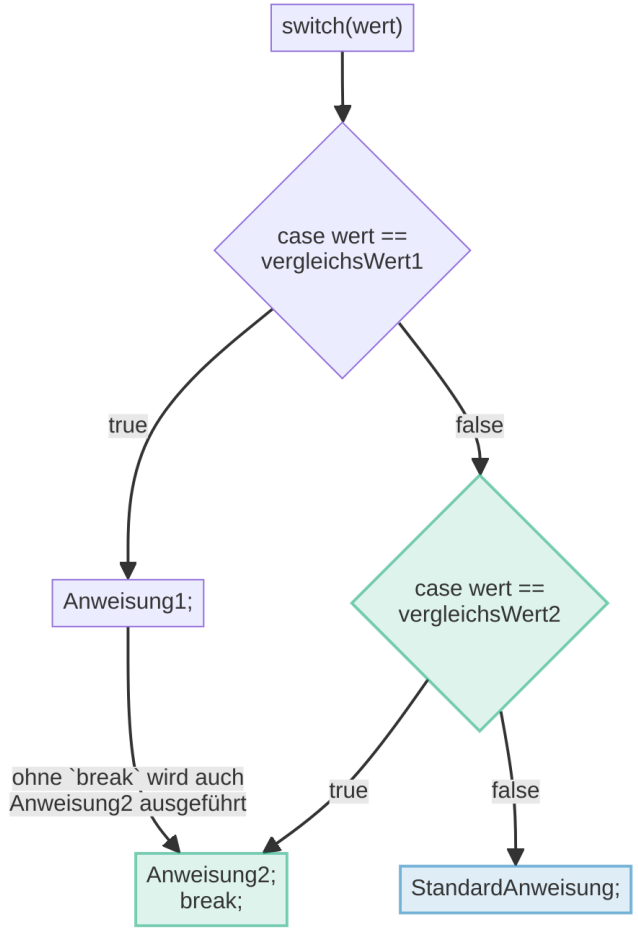
💡 if-Statements können beliebig verschachtelt werden!

Code-Beispiel

```
int age; // beliebiges alter
double betrag; // beliebiger Betrag
if (betrag > 10000 && age < 18) {
    // mehr als 10000 ausgegeben
    // UND unter 18 Jahre alt
    betrag *= 0.9;
} else if(betrag > 1000) {
    betrag *= 0.96;
} else { // Für alle andern
    betrag *= 0.98;
}
```

switch / case

- Wird dafür verwendet, **Gleichheit zu überprüfen**
- Als Bedingung dient die exakte Gleichheit von Werten (`==`)
- Mit `break` wird abgebrochen
- Ohne `break` wird die nächste Anweisung auch ausgeführt



switch / case

Schema

```
switch(wert) {
  case vergleichsWert1:
    <Anweisung1>;
    // ohne `break` wird auch <Anweisung2>
    // bis zum `break` ausgeführt.
  case vergleichsWert2:
    <Anweisung2>;
    break;
  default:
    <StarndardAnweisung>;
}
```

💡 Natürlich können beliebig viele `case` Blöcke folgen!

Code Beispiel

```
switch(kunde) {
    case "Hans":
        System.out.println("Hallo Hans!");
    case "Fritz":
        System.out.println("wie goots?");
        break;
    default:
        System.out.println("Ciao");
    }
}
```

Hans: Hallo Hans! wie goots?

Fritz: wie goots?

Alle Anderen: Ciao