

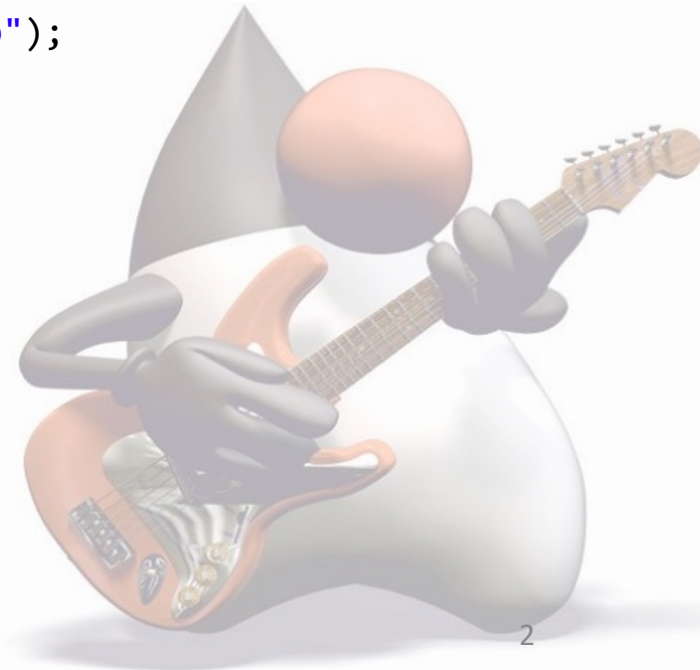
Java Grundlagen

Flußkontrolle



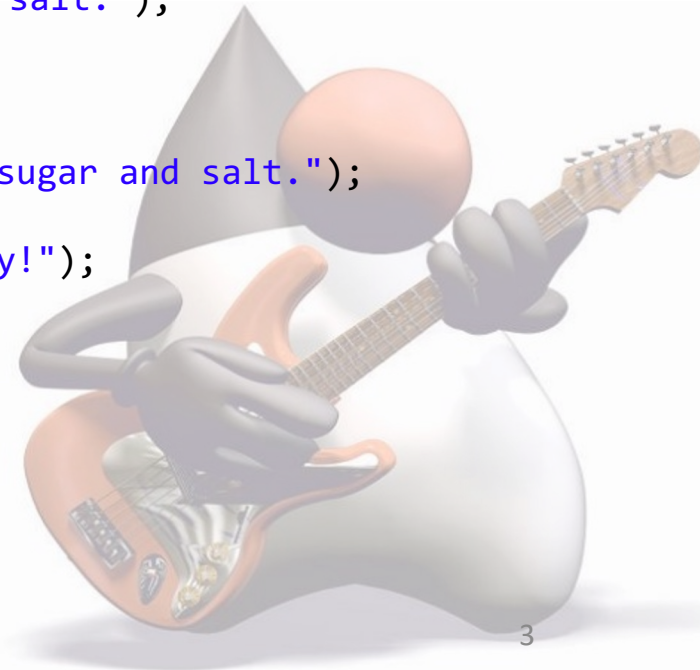
IF-Anweisungen

```
public class FlowControl {  
    public static void main(String[] args) {  
        int a = 12;  
        int b = 13;  
  
        if (a < b) {  
            System.out.println("a is less than b");  
        }  
    }  
}
```



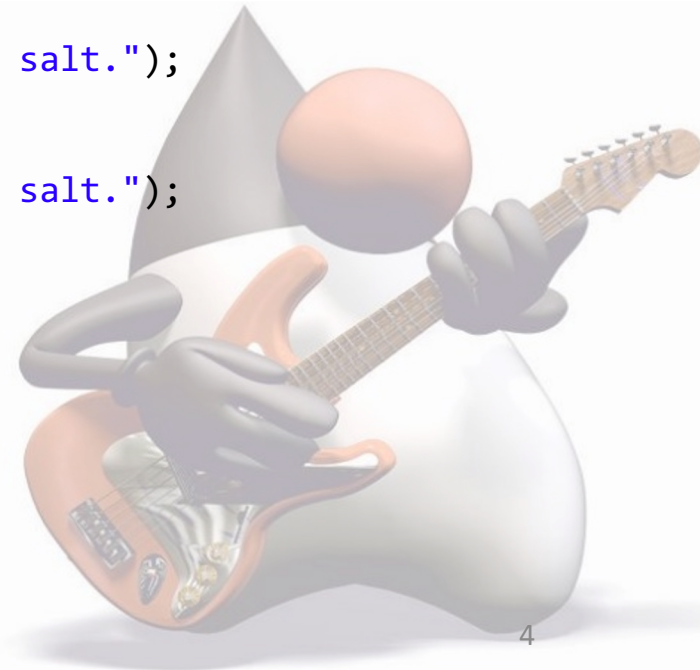
Verschachtelte If-Anweisungen

```
public class FlowControl {  
    public static void main(String[] args) {  
        double weight = 83.0F;  
        double height = 181.0F;  
        double bmi = weight / (height * height);  
  
        if (bmi > 25) {  
            System.out.println("You should eat less");  
            System.out.println("Possibly less fat and sugar and salt.");  
        } else {  
            if (bmi < 20){  
                System.out.println("You should eat more");  
                System.out.println("Possibly more fat and sugar and salt.");  
            } else {  
                System.out.println("Perfect. Go on this way!");  
            }  
        }  
    }  
}
```



If Kaskaden

```
public class FlowControl {  
    public static void main(String[] args) {  
        double weight = 83.0F;  
        double height = 181.0F;  
        double bmi = weight / (height * height);  
  
        if (bmi > 25) {  
            System.out.println("You should eat less");  
            System.out.println("Possibly less fat and sugar and salt.");  
        } else if (bmi < 20) {  
            System.out.println("You should eat more");  
            System.out.println("Possibly more fat and sugar and salt.");  
        } else {  
            System.out.println("Perfect. Go on this way!");  
        }  
    }  
}
```



If ohne Klammern

syntaktisch korrekt aber fehleranfällig

```
public class FlowControl {  
    public static void main(String[] args) {  
        double weight = 83.0F;  
        double height = 181.0F;  
        double bmi = weight / (height * height);  
  
        if (bmi > 25)  
            System.out.println("You should eat less");  
        else if (bmi < 20)  
            System.out.println("You should eat more");  
        else  
            System.out.println("Perfect. Go on this way!");  
    }  
}
```



String Vergleich

- Strings nicht mit == vergleichen
- == prüft bei Referenzdatentypen, ob es sich um das selbe Objekt handelt
- Der Inhalt von Strings wird mit der equals() Methode verglichen (Gleichheit)

```
package at.java;

import java.util.Scanner;

public class FlowControl {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Please enter your name");
        String name = scanner.next();
        if (name.equals("Christoph")) {
            System.out.println("VIP customer");
        } else {
            System.out.println("ordinary customer");
        }
    }
}
```



Schleifen (loops)

Es gibt in Java 3 verschiedene Arten von Schleifen

1. for
2. do/while
3. while



For-Schleife

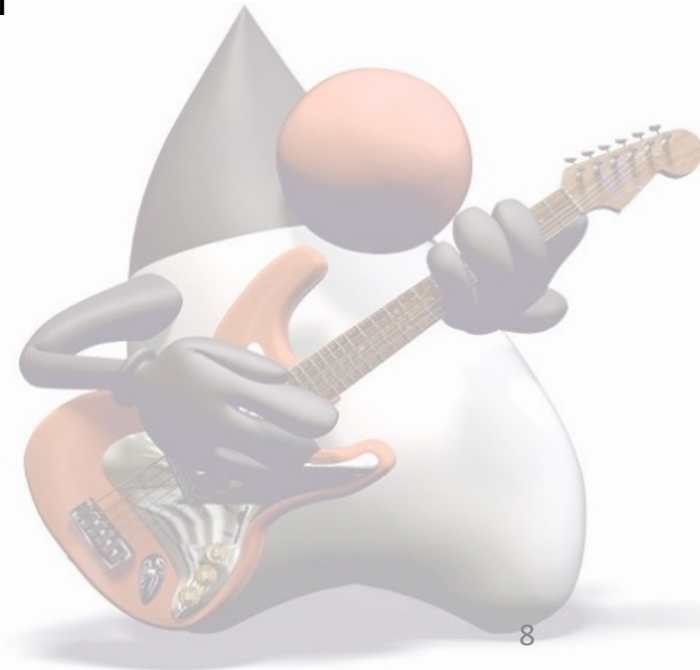
Initialisierungsausdruck –
wird für die erste Iteration
ausgeführt. Variablen
Deklarationen sind erlaubt.

Abbruchbedingung – wird vor
jedem Durchlauf ausgewertet

Aktualisierungsausdruck –
wird nach jedem Durchlauf
ausgewertet

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

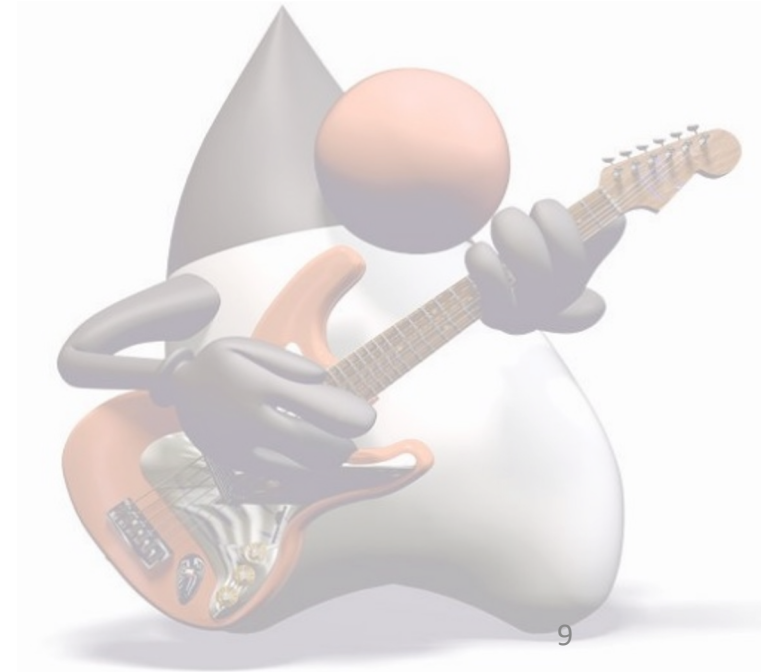

Die Variable `i` ist nur innerhalb der Schleife zugreifbar.
Merke: bei Java haben die Variablen „Block Scope“



While Schleife

Abbruchbedingung –
wird vor jedem Durchlauf
ausgewertet

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



Schleife unterbrechen

Die break Anweisung bricht die innerste Schleife ab.

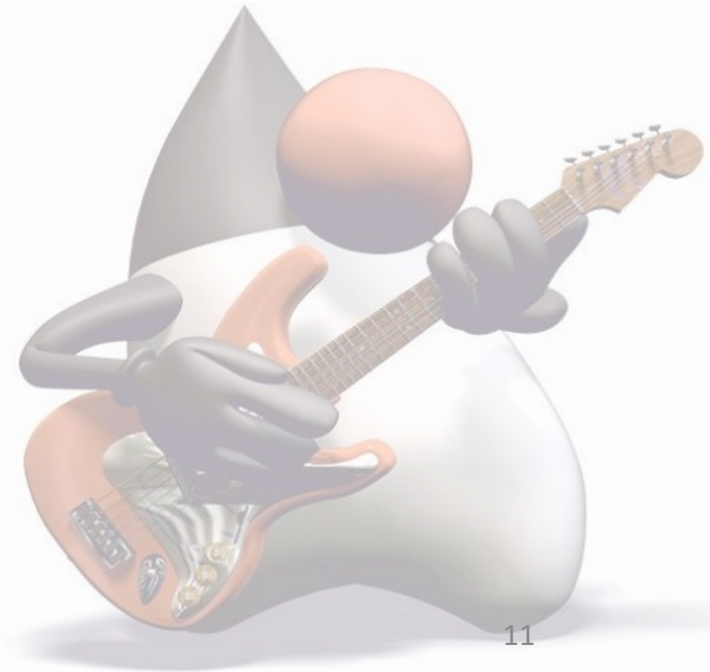
```
public class FlowControl {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
        while (true){  
            String input = scanner.next();  
            if (input.equals("exit")){  
                break;  
            } else {  
                // ...  
            }  
        }  
        System.out.println("Goodbye");  
    }  
}
```



Do-While Schleife

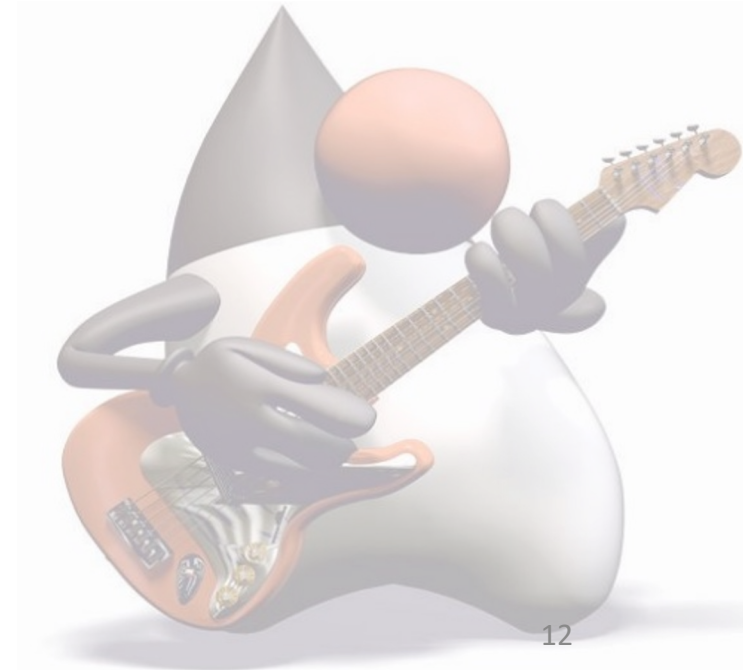
```
int i = 0;  
do {  
    System.out.println(i);  
    i++;  
} while (i < 10);
```

Abbruchbedingung – wird nach jedem Durchlauf ausgewertet.
Die Do-While Schleife wird daher zumindest 1 mal durchlaufen.



Komplexe Boolsche Ausdrücke

- Es gibt folgende boolsche Operatoren in Java
 - &.. And
 - |... or
 - !... not
- Shortcut Operatoren:
 - &&
 - ||

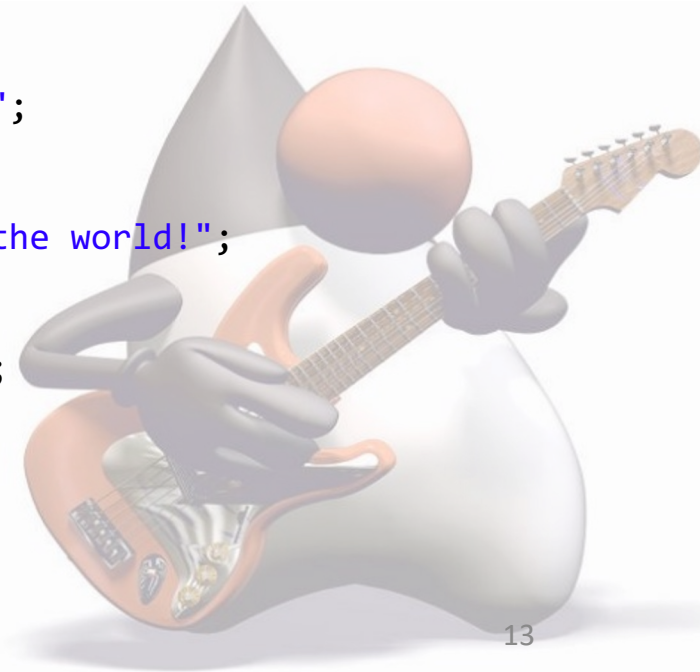


Switch Statement

```
public class Switcher {  
    public String getPerspectiveOfLife(int numberOfChildren) {  
        String result = "";  
        switch (numberOfChildren) {  
            case (0):  
                result = "What a wonderful life.";  
                break;  
            case (1):  
                result = "Do you remember? In your former life you had time for yourself.";  
                break;  
            case (2):  
                result = "With only one child life was so wonderful";  
                break;  
            case (3):  
                result = „Children are the most beautiful thing in the world!";  
                break;  
            default:  
                result = „The old ones take care of the young ones";  
                break;  
        }  
        return result;  
    }  
}
```

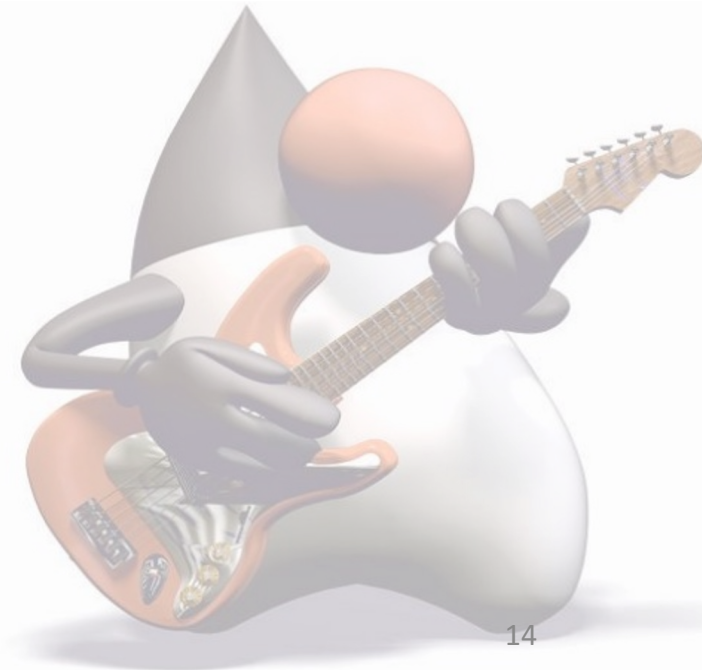
}

19.09.22



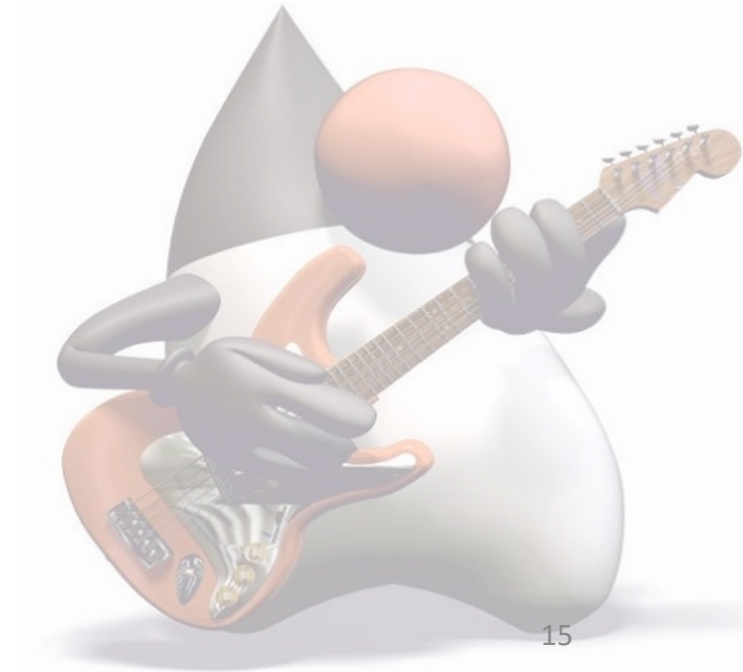
Switch Pfeil Notation (seit 14)

```
String result = "";
switch (numberOfChildren) {
    case (0) ->
        result = "What a wonderful life.";
    case (1) ->
        result = "Do you remember? In your former life you had time for yourself.";
    case (2) ->
        result = "With only one child life was so wonderful";
    case (3) ->
        result = "Children are the most beautiful thing in the world!";
    default -> {
        result = "The old ones take care of the young ones";
        result += "!";
    }
}
```



Switch Expressions (seit 14)

```
int numberOfPurchases = 5;
double discount = switch (numberOfPurchases){
    case 0 -> 0.;
    case 1 -> 5.;
    case 2 -> 7.;
    case 3 -> 10.;
    default -> {
        System.out.println("Very important customer!");
        yield 15.;
    }
};
```



Übung 3

- Schreiben sie ein Programm, indem der User eine Zahl zwischen 1 und 100 erraten soll.
- Die Zufallszahl wird mit `(int)(Math.random() * 100) + 1` berechnet.
- Der User erhält Feedback ob die geratene Zahl höher und niedriger ist als die eingegebene
- Ziel ist es, die Nummer zu erraten.



Übung 3.2

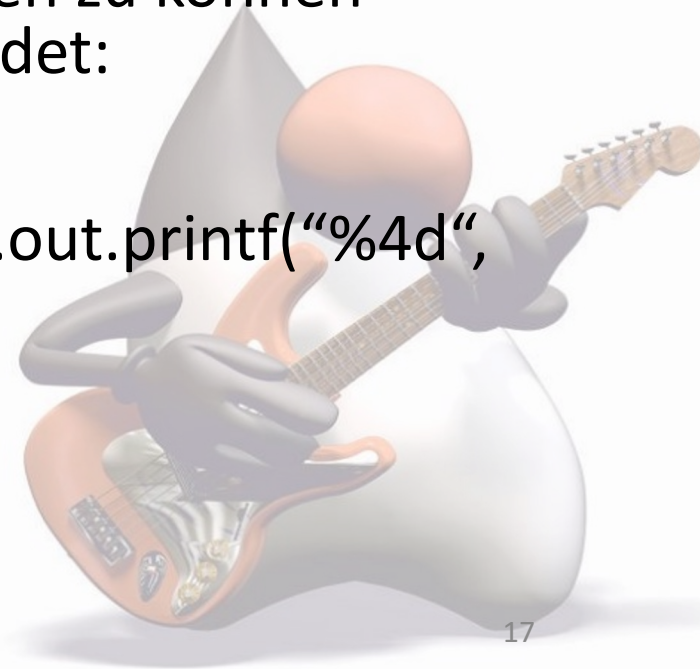
Schreiben sie ein Programm das die folgende Tabelle ausgibt:

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

Hinweis:

Um Zahlen formatiert
ausgeben zu können
verwendet:

`System.out.printf("%4d",
value)`



Ergänzende Literatur

- Java Tutorial
<https://docs.oracle.com/javase/tutorial/java/nutsandbolts/flow.html>
- Java Language Specification
<https://docs.oracle.com/javase/specs/jls/se11/html/jls-14.html#jls-14.12>

