



Einführung in die Programmierung

- JavaScript & Java -

Claudia Scheffler

23. September 2013



1 Methoden

2 Klassen

3 Objekte



1 Methoden

2 Klassen

3 Objekte



1 Methoden

2 Klassen

3 Objekte



Fragen ?



Methode

Unter einer Methode versteht man einen Programmcode, der nicht sofort ausgeführt wird, aber explizit auf- bzw. abgerufen werden kann. Diese führt entweder ein paar Befehle aus (Prozedur), oder es wird ein Wert zurückgegeben.

- Bezeichnung in anderen Programmiersprachen:
 - Funktion
 - Prozedur



Methoden

Namen von Methoden



- Methoden benötigen wie Variablen einen Namen
- damit sie aufgerufen werden können und um dem Programmieraufwand zu minimieren
- **Erlaubt ist:**
 - Buchstaben
 - Ziffern
 - Unterstrich (_)
- **Nicht erlaubt ist:**
 - Leer- und Sonderzeichen
 - das erste Zeichen ist eine Ziffer
 - Java Schlüsselwörter und -begriffe
- der Name der Methode beginnt grundsätzlich mit einem Kleinbuchstaben
- der Name der Methode sollte aussagekräftig sein



Methoden

Eigenschaften von Methoden



- Methoden sind **Bestandteile einer Klasse**
- Methoden können nur in **einer** Klasse deklariert und implementiert werden
- Methoden sind **Interaktionsmittel von Objekten**
- Methoden können **maximal einen Rückgabewert** besitzen
 - der Rückgabewert kann ein beliebiger Datentyp sein
- Methoden können **beliebig viele Parameter übergeben** werden
- Methoden können **beliebig oft überladen** werden



Parameter

Unter einem Parameter versteht man einen Wert, der als Eingangsgröße beim Aufruf einer Methode von dem aufrufenden Programm übergeben oder als Ergebnis von der Methode an das aufrufende Programm zurückgegeben wird.

- mehrere Parameter werden durch Komma getrennt
- für jeden Parameter wird innerhalb der Methode ein eigener Speicherplatz erzeugt, in den der Parameterwert kopiert wird
- Variablen bleiben dadurch unverändert



- die Übergabeparameter werden als **call-by-value** übergeben

call-by-value

Bei call-by-value werden die an einer Methode übergebenen Daten **kopiert**. Es besteht dann keine Verbindung mehr zwischen den Daten beim Aufruf der Methode und den Daten in der Methode.



- in vielen anderen Programmiersprachen werden die Übergabeparameter als **call-by-reference** übergeben

call-by-reference

Anstatt die Daten zu kopieren werden Referenzen auf die Daten übergeben. Man kann sich das als Zeiger vorstellen: Beim Aufrufer zeigt eine Variable auf ein bestimmtes Datum, in der Methode zeigt eine andere Variable auf dasselbe Datum. Methodenaufrufe an einem so übergebenen Objekt arbeiten also auf demselben Objekt, das auch außerhalb sichtbar ist.



Syntax

Methodendeklaration

```
{  
...  
}
```

- Methodenkopf kann durch einen Zugriffsmodifikatoren eingeleitet werden → public, protected, private
 - gibt Informationen über die Sichtbarkeit der Methode

Tipp!

Ein Modifikator ist kein Muss und sollte gut überlegt sein.



Achtung!

Zwingend erforderlich ist hingegen die Angabe des Rückgabedatentypen.

- mit diesem wird angegeben, zu welchem Datentyp der Rückgabewert der Methode gehört
- anschließend wird der Name der Methode und ein Paar runde Klammern angeben
- in dem Klammerpaar werden die möglichen Übergabeparameter angegeben
 - die Übergabeparameter müssen mit einem Datentyp versehen werden
 - können als lokale Variable verwendet werden



Beispiel

```
class NameDerKlasse
{
    int x_koordinate;
    int y_koordinate;
    public void set_x_Koordinate ( int zusetzendeX_Koordinate )
    {
        this.x_koordinate = zusetzendeX_Koordinate;
    }
}
```



main()-Methode

```
public static void main(String[ ] args)
{
    // Anweisungen
}
```

- **Zugriffsmodifikatoren:** public und static
- **Rückgabewert:** void → es wird kein Wert zurück gegeben
- **Name:** main
- **Übergabeparameter:** Array mit Strings



Methoden

Aufruf von Methoden



- um auf eine Methode einer Klasse zugreifen zu können, muss zuerst ein Objekt von der Klasse erstellt werden
- über die Punktnotation kann dann auf die Methode zugegriffen werden

Beispiel

```
class MethodenBeispiel {  
    public static void main(String[] args) {  
        Punkt punkt_eins = new Punkt();  
        punkt_eins.set_x_Koordinate(5);  
    }  
}  
class NameDerKlasse {  
    int x_koordinate;  
    int y_koordinate;  
    public void set_x_Koordinate ( int zusetzendeX_Koordinate) {  
        this.x_koordinate = zusetzendeX_Koordinate;  
    }  
    public int get_x_Koordinate () {  
        return this.x_koordinate;  
    }  
}
```



Überladen

Unter dem sogenannten Überladen versteht man, dass derselbe Methodenname mehrfach in einer Klasse verwendet werden kann. Das Überladen wird häufig bei Konstruktoren oder Konvertierungsmethoden verwendet.

- Voraussetzung:
 - der Datentyp mindestens eines Übergabeparameters ist anders als in den übrigen gleichnamigen Methoden
 - die Anzahl der Übergabeparameter ist unterschiedlich

Beispiel

```
public static int abs (int)
public static float abs (float)
public static long abs (long)
public static double abs (double)
```



Iteration

Unter einer Iteration (lat. Wiederholung) versteht man die mehrfache Ausführung einer oder mehrerer Anweisungen. Die Iteration wird realisiert durch Schleifen (for, while..). Mittels einer Abbruchbedingung wird die Schleife beendet.

- Vorteil:
 - performanter



Rekursion

Von Rekursion (von lateinisch recurrere = zurücklaufen) spricht man, wenn eine Methode sich selbst immer wieder aufruft bis eine Abbruchbedingung erfüllt ist. Jede Rekursion lässt sich auch in eine iterative Lösung umwandeln und umgekehrt.

- Vorteil:
 - weniger Quellcode
 - übersichtlicher
- Nachteil:
 - speicherintensiver



Beispiel

```
class IterativFakultaet {  
    static long berechneFakultaet(int n) {  
        long faku =1;  
        for(int i = 1; i<= n; i++) {  
            faku *= i;  
        }  
        return faku;  
    }  
    public static void main(String[ ] args) {  
        long faku = berechneFakultaet(5);  
        System.out.println("5! = "+faku);  
    }  
}
```



Beispiel

```
class RekursivFakultaet {  
    static long berechneFakultaet(int n) {  
        System.out.println(" Aufruf mit " + n); if(n >= 1) {  
            return n * berechneFakultaet(n - 1);  
        }  
        else {  
            return 1;  
        }  
    }  
    public static void main(String[] args) {  
        long faku = berechneFakultaet(5);  
        System.out.println(" 5! = " + faku);  
    } }  
}
```



Analyse

```
if(n>=1) {  
  return n*berechneFakultaet(n-1);  
} else {  
  return 1; }
```

1. Aufruf mit 5: $5 * \text{berechneFakultaet}(5-1)$
2. Aufruf mit 4: $5 * 4 * \text{berechneFakultaet}(4-1)$
3. Aufruf mit 3: $5 * 4 * 3 * \text{berechneFakultaet}(3-1)$
4. Aufruf mit 2: $5 * 4 * 3 * 2 * \text{berechneFakultaet}(2-1)$
5. Aufruf mit 1: $5 * 4 * 3 * 2 * 1 * \text{berechneFakultaet}(1-1)$
6. Aufruf mit 0: $5 * 4 * 3 * 2 * 1 * 1$



Fragen ?



Klassen

Was ist eine Klasse ?



Klasse

Eine Klasse ist eine allgemeingültige Beschreibung von Dingen, die in verschiedenen Formen vorkommen können, aber alle eine gemeinsame Struktur und ein gemeinsames Verhalten haben. Sie ist ein Bauplan für die Erzeugung von einzelnen konkreten Formen. Diese Form bezeichnet man auch als **Objekte** oder **Instanzen** der Klasse.

■ Bestandteile:

- Variablen → Attribute
- Methoden

Attribute

Attribute beschreiben den Zustand und Methoden, also das Verhalten eines Objektes.



Syntax

```
[Modifizierer]class NameDerKlasse {  
  Attributdeklartion  
  Methodendeklaraion  
}
```



- Erlaubt:
 - Buchstaben
 - Nummern
 - Währungssymbole
 - Unterstrich (_)
- Nicht Erlaubt:
 - Ziffer als erstes Zeichen
 - Schlüsselwörter
 - Leerzeichen



Klassen

Namen von Klassen



- der erste Buchstabe sollte immer groß sein
- das erste Wort des Klassennamens ist klein geschrieben und alle anderen ohne Leerzeichen folgende Wörter beginnen mit großen Anfangsbuchstaben
- nur Nomen (Substantive) verwenden
- aussagekräftige Namen verwenden
- es sollten keine Abkürzungen verwendet werden



Globale Variable (Instanzvariablen)

Eine Variablen, die **nicht** innerhalb einer Methode definiert ist, wird globale Variable genannt.

- dieser Art von Variablen können überall im Quelltext verwendet werden

Lokale Variable

Eine Variablen, die innerhalb einer Methode definiert ist, wird lokale Variable genannt.

- diese Art von Variablen können nur innerhalb der Methode in der sie definiert wurden sind, genutzt werden



Beispiel

```
class NameDerKlasse {  
    int globaleVariable;  
    public void NameDerMethode()  
    {  
        int lokaleVariable;  
    }  
}
```



Klassenvariable

Ein Attribut einer Klasse, dessen Deklaration mit dem Schlüsselwort **static** versehen ist, wird Klassenvariable genannt.

- es existiert immer nur eine Klassenvariable, unabhängig von der Anzahl der Instanzen der Klasse
- die Lebensdauer der Variablen erstreckt sich über das ganze Programm
- der Zugriff auf diese Variable erfolgt nach dem gleichen Schema wie bei Methoden



Beispiel

```
class NameDerKlasse {  
    int x;  
    static int y;  
    void nameDerMethode () {  
        int z=0;  
    }  
}
```



- ist in jeder Klasse vorhanden und kann auch als Zeiger bezeichnet werden
- this bezieht sich stets auf das Objekt, das die jeweilige Methode aufgerufen hat

Tipp!

Auf Variablen und Methoden kann natürlich auch ohne this zugegriffen werden. Wenn auf die eigenen Variablen und Methoden einer Klasse verwendet werden möchte, sollte man sich angewöhnen, die this-Referenz vorweg zu schreiben.



Beispiel

```
class NameDerKlasse {  
  int x;  
  void nameDerMethode (int val) {  
    this.x=val;  
  }  
}
```



Fragen ?



Modifizierer

Mithilfe bestimmter Schlüsselwörter in Java, sogenannte Modifizierer, können die Eigenschaften von Klassen, Attributen und Methoden verändert werden.

Modifizierer	Klasse	Attribute	Methode	Konstruktor
public	x	x	x	x
protected		x	x	x
private		x	x	x
static		x	x	
final	x		x	
abstract	x		x	



- Zugriffsrechte auf Klassen, Attribute und Methoden werden über die Modifizierer `public`, `protected` und `private` gesteuert

Paket

Ein Paket ist eine Sammlung von Klassen, die aus bestimmten Gründen zusammengefasst werden. Zum Beispiel, weil sie all zu einem bestimmten Anwendungsprojekt gehören.

- jede Klasse gehört zu genau einem Paket



■ **public:**

- Klassen die mit diesem Modifizierer gekennzeichnet sind, können überall verwendet werden
- in einer Quelldatei darf höchstens eine Klasse public deklariert werden
- Attribute können von überall her lesend und schreibend zugegriffen werden
- Methoden und Konstruktoren können von allen Klassen verwendet werden



■ **protected:**

- Attribute, Methoden und Konstruktoren können die eigene Klasse und alle anderen Klassen innerhalb des Paketes zugreifen
- derartige Attribute und Methoden werden an alle Subklassen weiter vererbt und sind dann dort als geerbte Komponenten zugänglich



■ **private:**

- Attribute, Methoden und Konstruktoren können nur in der sie deklarierten Klasse verwendet werden

Tipp!

Attribute sollten immer als private deklariert werden und indirekt über Methoden zugänglich gemacht werden



Fragen ?



Objekte

Was ist ein Objekt ?



Objekt

Ein Objekt ist eine zusammengesetzte Datenstruktur mit Eigenschaften zur Speicherung von Informationen über das Objekt und Methoden, die bereitgestellt werden, um Objektwerte zu erhalten, zu setzen oder komplexere Operationen auszuführen.



Objekthinhalt

Bei den Inhalten eines Objektes, handelt es sich um eine Referenz (ein Verweis) auf das Objekt selbst.

- es können alle möglichen Daten zusammengefasst werden
- zum Beispiel:
 - ganze Zahlen
 - Arrays
 - Objekte



Objekteigenschaften

Bei den Eigenschaften eines Objektes handelt es sich um Variablen von Objekten. Diese sind festgelegt, da sie einen vorgegebene Wert liefern.

- können aus einen beliebigen Datentyp, Array oder Objekt bestehen
- **Inhalte von Eigenschaften:**
 - konkrete Werte
 - Eigenschaften können mit neuen Werten versehen werden



Objektmethoden

Bei Methoden von Objekten handelt es sich, um Methoden, die von den Objekten bereitgestellt werden.



Ein Auto ist ein Objekt mit

- Eigenschaften:

- Farbe
- Anschaffungspreis
- Baujahr
- aktuelles Gewicht
-

- Methoden:

- starten
- beschleunigen
- bremsen
- schalten
-



Konstruktor

Unter einem Konstruktor versteht man eine spezielle Methode, die die nötigen Anweisungen enthält, um ein neues Objekt gemäß einer Vorlage zu konstruieren.

- Aufruf erfolgt mit dem Schlüsselwort **new** und liefert ein neues Objekt zurück

Beispiel

```
String NameDesObjektes = new String ("Ich bin eine  
Zeichenkette");
```



- bei manchen Konstruktoren können auch Argumente übergeben werden
- trägt den Namen der zugehörigen Klasse und hat keinen Rückgabebetyp
- in einer Klasse können mehrere Konstruktoren enthalten sein
- müssen dann aber in ihren Parametern verschieden sein
- durch die Angabe der Parameter wird entschieden, welcher der Konstrukturen, bei der Erzeugung des Objektes aufgerufen wird
- Datentyp sowie die Anzahl der Parameter müssen übereinstimmen



Standardkonstruktor

Ist für eine Klasse kein Konstruktor implementiert wurden, so wird ein Standardkonstruktor ohne Parameter vom Compiler eingebaut



- können sich gegenseitig aufrufen, somit kann vorhandener Programmcode wiederverwendet werden
- Aufruf eines Konstruktors muss als erste Anweisung mit `this` erfolgen

Beispiel

```
Konto(int kontonummer, double saldo) {  
    this(kontonummer);  
    this.saldo = saldo;  
}
```



Instanz

Unter einer Instanz versteht man, wenn ein konkretes Objekt einer Familie gleichartiger Objekte, durch einen Konstruktoraufruf erzeugt wird.

Beispiel

```
Punkt p = new Punkt();
```



Beispiel

```
public class Beispiel {  
    public static void main(String[] args) {  
        Punkt p = new Punkt();  
        System.out.println("Objekt p: " + p);  
        System.out.println("Standardkonstruktor xKoordinate: " + p.xKoordinate);  
        System.out.println("Standardkonstruktor yKoordinate: " + p.yKoordinate);  
        p = new Punkt(25);  
        System.out.println("Objekt p nach dem zweiten Aufruf eines Konstruktors: " + p);  
        System.out.println("Konstruktor mit einem Übergabeparameter xKoordinate: " + p.xKoordinate);  
        System.out.println("Konstruktor mit einem Übergabeparameter yKoordinate: " + p.yKoordinate);  
        p = new Punkt(33,45); System.out.println("Objekt p nach dem dritten Aufruf eines Konstruktors: " + p);  
        System.out.println("Konstruktor mit zwei Übergabeparametern xKoordinate: " + p.xKoordinate);  
        System.out.println("Konstruktor mit zwei Übergabeparametern yKoordinate: " + p.yKoordinate);  
    }  
}
```



Fragen ?

Claudia Scheffler
siclsche@stud.hszg.de