



Universität Stuttgart
MINT-Kolleg Baden-Württemberg



Objektorientierung 1: Klassen, Methoden

Vorkurs Informatik



Universität Stuttgart

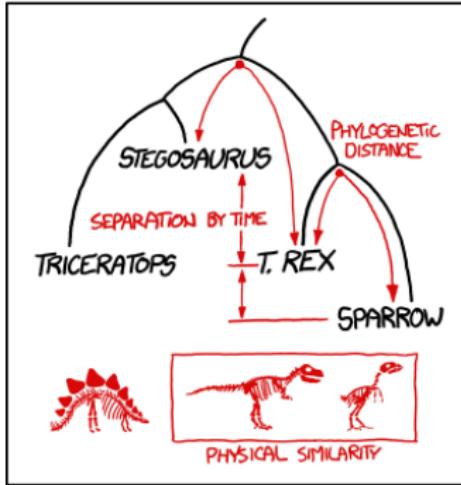


F. Schweiner



Getting started ...

BY ANY REASONABLE DEFINITION, T.REX IS MORE CLOSELY RELATED TO SPARROWS THAN TO STEGOSAURUS.



BIRDS AREN'T DESCENDED FROM DINOSAURS,
THEY ARE DINOSAURS.

WHICH MEANS THE FASTEST ANIMAL ALIVE TODAY IS
A SMALL CARNIVOROUS DINOSAUR, FALCO PEREGRINUS.



IT PREYS MAINLY ON OTHER DINOSAURS, WHICH
IT STRIKES AND KILLS IN MIDAIR WITH ITS CLAWS.

THIS IS A GOOD WORLD.

Outline

- 1 Klassen, Objekte, Methoden
- 2 Objekte
- 3 Klassen
- 4 Klassen und Methoden in Java
- 5 Klassendokumentation
- 6 Strings
- 7 Zusammenfassung

(nach Folien von L. Vettin, V. Weidler, W. Kessler)  CC Attribution-NonCommercial-ShareAlike 4.0 Licence

(und *Programmierung und Softwareentwicklung*, Prof. Dr.-Ing. Steffen Becker und Sandro Speth,
Institute of Software Engineering, Universität Stuttgart)

Outline

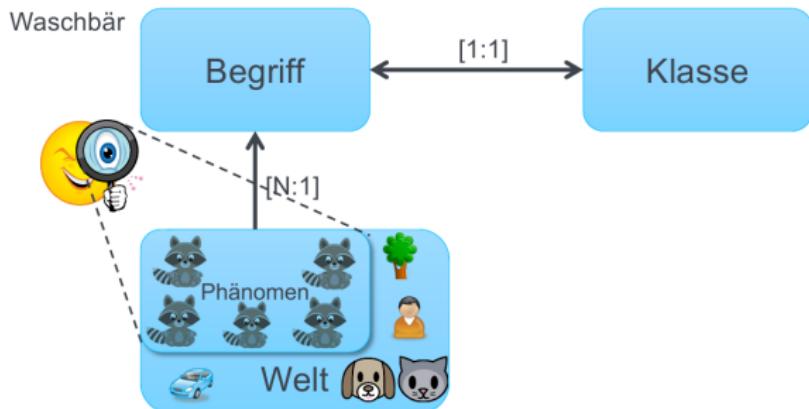
- 1 Klassen, Objekte, Methoden
- 2 Objekte
- 3 Klassen
- 4 Klassen und Methoden in Java
- 5 Klassendokumentation
- 6 Strings
- 7 Zusammenfassung

Einige Hinweise vorab ...

- ▶ Objektorientierung ist ein komplexes und facettenreiches Konzept, das im Rahmen dieses Kurses nicht vollständig erklärt und vermittelt werden kann.
- ▶ Objektorientierung spielt ihre Stärken erst in größeren Projekten wirklich aus. Eine solche Projektgröße lässt sich in der Kürze der Zeit hier nicht erreichen.
- ▶ Gutes objektorientiertes Design zu lernen braucht Zeit und Erfahrung. Lassen Sie sich also nicht entmutigen, wenn die neuen Konzepte erst mal nicht vollkommen überzeugend wirken.

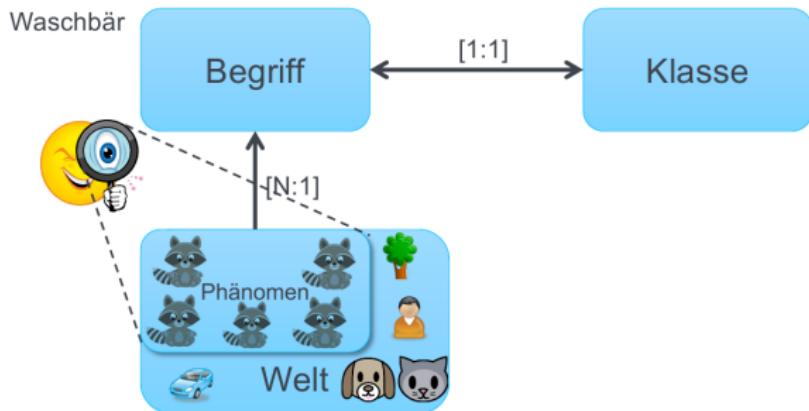
Realwelt vs. Modell

- In der Realwelt begegnen wir verschiedensten Dingen.



Realwelt vs. Modell

- ▶ In der Realwelt begegnen wir verschiedensten Dingen.



- ▶ Wenn mehrere ähnliche Dinge auftreten, spricht man von einem Phänomen. Diesem wird ein Begriff zugeordnet (hier: Waschbär).
- ▶ Für diesen Begriff können wir eine **Klasse** erstellen.

Klassen

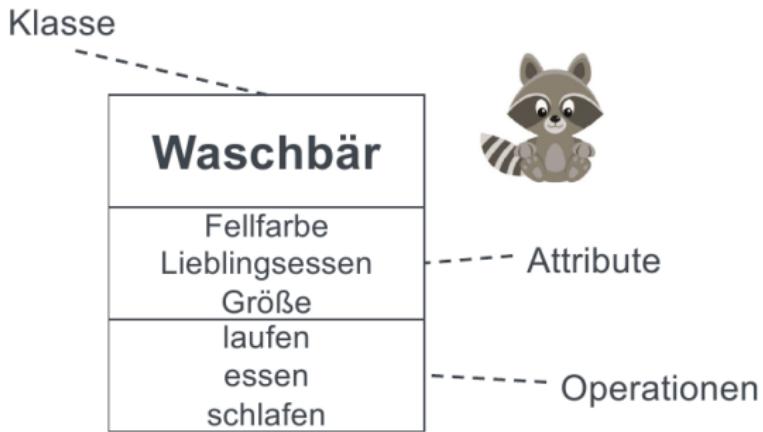
Eine **Klasse** ist eine verallgemeinerte Sammlung von Eigenschaften.

- Sie dient nur als Bauplan von Objekten.

Klassen

Eine **Klasse** ist eine verallgemeinerte Sammlung von Eigenschaften.

- ▶ Sie dient nur als Bauplan von Objekten.
- ▶ Sie definiert für die Objekte u.a.
 - ▶ die Eigenschaften (**Attribute**) und
 - ▶ die Fähigkeiten (**Operationen**)



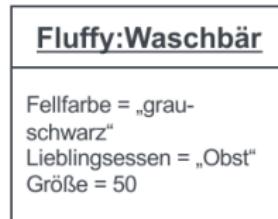
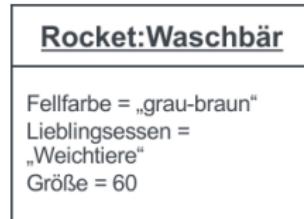
Vereinfachtes Klassendiagramm

Klassen

- ▶ Von einer Klasse können beliebig viele Objekte erzeugt werden.

Klassen

- ▶ Von einer Klasse können beliebig viele Objekte erzeugt werden.

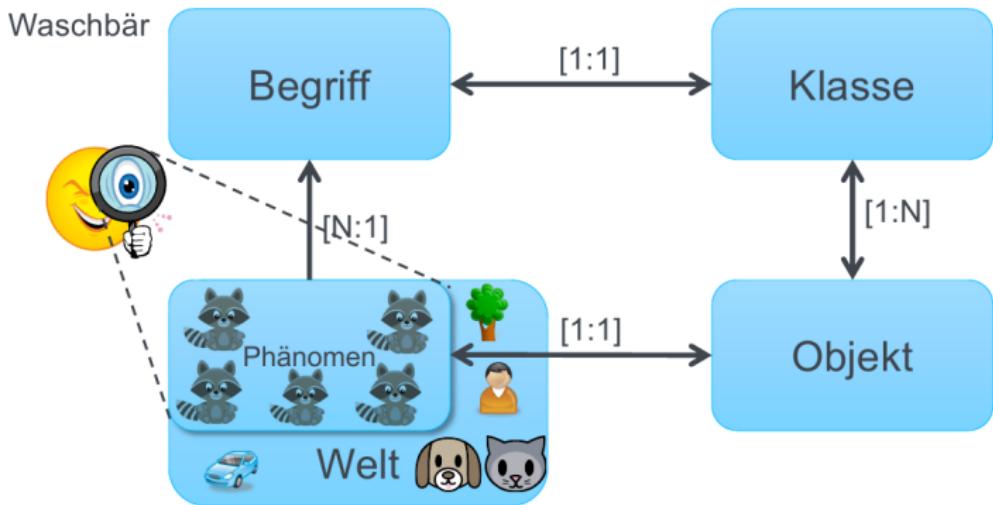


Vereinfachtes Objektdiagramm

- ▶ Es ist in der realen Welt z.B. nicht natürlich, dass es von einer Klasse nur eine Instanz gibt.

Klassen

- ▶ Insgesamt ergibt sich folgendes Bild:



Klassen

- ▶ Die OOP ermöglicht ein realistisches Modell der Realität, das man sich leicht vorstellen kann.
- ▶ Klassen bzw. Objekte können in Beziehung zueinander stehen.
 - ▶ Beispiel: Waschbären sammeln Obst, Weichtiere, ...



<https://pixabay.com/de/illustrations/n%C3%A4hen-verbindung-kommunikation-5034027/>

Klassen

- ▶ Können Sie weitere Beispiele für Klassen und zugehörige Objekte geben?

Klassen

- ▶ Können Sie weitere Beispiele für Klassen und zugehörige Objekte geben?
- ▶ Klasse: Person/Mensch
 - ▶ Objekte: Angela Merkel, Michael Jackson, ...
- ▶ Klasse: Hund
 - ▶ Objekte: Fifi, Rex, Snoopy, ...
- ▶ Klasse: Bankkonto
 - ▶ Objekte: BWBank Konto 123, Volksbank Konto 543, ...

Klassen

Entscheiden Sie: Was sind Klassen, was sind Objekte?

- ▶ Ihr Kugelschreiber
- ▶ Vorlesung
- ▶ Heutiger Vorkurs-Termin
- ▶ MINT-Kolleg
- ▶ Hamster
- ▶ Beamer an der Decke

Klassen

Entscheiden Sie: Was sind Klassen, was sind Objekte?

- ▶ Ihr Kugelschreiber (O)
- ▶ Vorlesung (K)
- ▶ Heutiger Vorkurs-Termin (O)
- ▶ MINT-Kolleg (O)
- ▶ Hamster (K)
- ▶ Beamer an der Decke (O)

Outline

- 1 Klassen, Objekte, Methoden
- 2 Objekte
- 3 Klassen
- 4 Klassen und Methoden in Java
- 5 Klassendokumentation
- 6 Strings
- 7 Zusammenfassung

Objekte

- ▶ Ein Objekt hat eine **Identität**. Diese gibt einem abstrakten Objekt einen Namen und identifiziert das Objekt eindeutig.



Objekte

- ▶ Objekte sind konkrete **Instanzen** einer Klasse.
Jedes Objekt besitzt genau eine Klasse.
- ▶ Objekte besitzen alle Operationen der Klasse.
- ▶ Objekte sind **einzigartig**.
Mehrere Objekte können zur selben Klasse gehören.

Objekte

- ▶ Objekte sind konkrete **Instanzen** einer Klasse.
Jedes Objekt besitzt genau eine Klasse.
 - ▶ Objekte besitzen alle Operationen der Klasse.
 - ▶ Objekte sind **einzigartig**.
Mehrere Objekte können zur selben Klasse gehören.
-
- ▶ Objekte enthalten individuelle Werte in den **Attributen**, die verändert werden können.
 - ▶ Objekte besitzen Fähigkeiten / **Operationen**, die ausgeführt werden können.
 - ▶ Objekte können mit anderen Objekten **interagieren**.

Outline

- 1 Klassen, Objekte, Methoden
- 2 Objekte
- 3 Klassen
- 4 Klassen und Methoden in Java
- 5 Klassendokumentation
- 6 Strings
- 7 Zusammenfassung

Klassen

- ▶ Jedes Objekt gehört also zu einer bestimmten Klasse, die die Attribute sowie die anwendbaren Operationen definiert.

Klassen

- ▶ Jedes Objekt gehört also zu einer bestimmten Klasse, die die Attribute sowie die anwendbaren Operationen definiert.

Beispiele:

- ▶ Die Klasse aller **Waschbären**:
 - ▶ Attribute: Fellfarbe, Lieblingsessen, Größe
 - ▶ Operationen: laufen, essen, schlafen

Klassen

- ▶ Jedes Objekt gehört also zu einer bestimmten Klasse, die die Attribute sowie die anwendbaren Operationen definiert.

Beispiele:

- ▶ Die Klasse aller **Waschbären**:
 - ▶ Attribute: Fellfarbe, Lieblingsessen, Größe
 - ▶ Operationen: laufen, essen, schlafen
- ▶ Die Klasse aller **Bankkonten**:
 - ▶ Attribute: Kontostand, Kontonummer, Kontoinhaber
 - ▶ Operationen: Geld abheben, Kontostand abfragen, ...
- ▶ ...

Klasse

Eine **Klasse** ist ein Bauplan für eine Reihe von möglichen Objekten, auf denen die gleichen Methoden/Operationen ausgeführt werden können.

Outline

- 1 Klassen, Objekte, Methoden
- 2 Objekte
- 3 Klassen
- 4 Klassen und Methoden in Java
- 5 Klassendokumentation
- 6 Strings
- 7 Zusammenfassung

Eine eigene Klasse definieren

- ▶ Für jede Klasse muss eine weitere Datei angelegt werden!
- ▶ Die Klassen dienen als Schablone für Objekte, d. h. sie beschreibt, welche Eigenschaften/Methoden diese haben
- ▶ Konkrete Objekte werden meist außerhalb(!) der Klasse - bei kleinen Programmen häufig in der `main`-Methode - erzeugt. Diese steht in einer weiteren Datei.

Erstes Beispiel

Datei Raccoon.java:

```
public class Raccoon {  
  
    public String color = "gray";  
  
    public void sleep() {  
        System.out.println("I'm sleeping!");  
    }  
  
    public void eat(int times) {  
        for (int i=0; i<times; i++) {  
            System.out.println("I'm eating!");  
        }  
    }  
  
    public String getColor() {  
        return this.color;  
    }  
}
```

Datei Test.java:

```
public class Test {  
  
    public static void main  
        (String[] args) {  
  
        Raccoon fluffy = new Raccoon();  
  
        fluffy.sleep();  
        fluffy.eat(5);  
        String color =  
            fluffy.getColor();  
    }  
}
```

Eine eigene Klasse definieren

- ▶ Zur Definition von Klassen gibt es das Schlüsselwort `class`.
- ▶ Eine Klasse hat einen Namen. Der Name beginnt mit einem Großbuchstaben und folgt sonst den gleichen Regeln wie Variablennamen (keine Leerzeichen!).
- ▶ Eine Klasse `<Name>` muss in der Datei `<Name>.java` definiert werden, z. B. Klasse `Raccoon` in Datei `Raccoon.java`.
- ▶ Syntax: `public class <Klassenname> { ... }`

```
public class Raccoon {  
    // Code für die Klasse einfügen  
}
```

Attribute / Methoden definieren

- ▶ Attribute werden wie Variablen deklariert.
- ▶ Methoden einer Klasse werden durch ihre Signatur beschrieben.
- ▶ Syntax: <Rückgabetyp> <Name>(<Parameter>*) { ... }

```
public class Raccoon {  
  
    public String color = "gray";  
  
    public void sleep() {  
        // Code für die Methode einfügen  
    }  
    public void eat(int times) {  
        // Code für die Methode einfügen  
    }  
    public int getColor() {  
        // Code für die Methode einfügen  
    }  
}
```

Objekte erzeugen

- ▶ Um ein Objekt einer Klasse zu erzeugen, verwendet man das Schlüsselwort `new`.
- ▶ Eine Klasse kann Parameter haben, für die bei der Erzeugung Argumente übergeben werden müssen.
- ▶ Syntax: `new <Klassenname>(<Argument>*)`

```
Raccoon fluffy = new Raccoon();
```

Methodenaufruf

- ▶ Wenn Sie ein neues Objekt erzeugt haben (z.B. `fluffy`), dann können Sie für dieses Objekt Methoden aufrufen.
- ▶ Syntax: `<Objektname>. <Methodename>(<Argumente>*)`

```
fluffy.sleep();
fluffy.eat(5);
String color = fluffy.getColor();
```

void vs. andere Rückgabetypen

Wir unterscheiden Abfragen und Kommandos.

- ▶ Abfragen: Methoden mit einem Rückgabetyp, der nicht `void` ist, sind Fragen, auf die eine Antwort zurückgegeben wird:

- ▶ `int getColor()`

Frage: Welche Farbe hast Du?

Antwort: grau

void vs. andere Rückgabetypen

Wir unterscheiden Abfragen und Kommandos.

- ▶ Abfragen: Methoden mit einem Rückgabetyp, der nicht `void` ist, sind Fragen, auf die eine Antwort zurückgegeben wird:

- ▶ `int getColor()`

Frage: Welche Farbe hast Du?

Antwort: grau

- ▶ Kommandos: Methoden mit dem Rückgabetyp `void` stellen Befehle dar. Sie führen eine Aktion aus, geben aber keine Rückantwort:

- ▶ `void sleep()`

Befehl: schlafe!

- ▶ `void eat(int times)`

Befehl: iss ... mal!

Outline

- 1 Klassen, Objekte, Methoden
- 2 Objekte
- 3 Klassen
- 4 Klassen und Methoden in Java
- 5 Klassendokumentation
- 6 Strings
- 7 Zusammenfassung

Klassendokumentation

In der Dokumentation einer Klasse ist genau beschrieben, über welche Methoden eine Klasse verfügt und wie sich diese verhalten.

- ▶ Dokumentation ist extrem wichtig, um Wiederverwendbarkeit von Code sicherzustellen.
- ▶ Es geht meist schneller, die Erklärung zu lesen was der Code macht, als den Code komplett nachzuvollziehen.
- ▶ Oft gibt die Dokumentation wichtige Hinweise auf erlaubte Argumente oder zu erwartende Rückgabewerte.

Kommentare und Klassendokumentation

- Zeilenkommentar (für Kommentare im Code):

```
1 int a; // Der Rest dieser Zeile ist Kommentar
```

- Blockkommentar (für Kommentare im Code):

```
1 /* Ein Blockkommentar kann ueber  
2 mehrere Zeilen gehen */
```

- Dokumentationskommentar (zu einer Klasse oder Methode):

```
1 /**  
2 * Ein Dokumentationskommentar ist ein  
3 * spezieller Blockkommentar.  
4 */
```

Klassendokumentation erstellen

```
/**  
 * Ein Dokumentationskommentar steht bei der  
 * Definition einer Klasse oder einer Methode, daraus  
 * kann automatisch eine Dokumentation erstellt werden.  
 *  
 * @param ...  
 * @return ...  
 */
```

- ▶ Aus Dokumentationskommentaren kann mit dem im JDK enthaltenen Programm JavaDoc automatisch eine Dokumentation erstellt werden: javadoc *.java
- ▶ Das Resultat sind HTML-Dokumente, die mit dem Code zusammen weitergegeben werden können.

Beispiel: String API

- ▶ Beispiel für ein HTML-Dokument:

Method Summary

All Methods	Static Methods	Instance Methods	Concrete Methods	Deprecated Methods
Modifier and Type		Method and Description		
char		<code>charAt(int index)</code> Returns the char value at the specified index.		
int		<code>codePointAt(int index)</code> Returns the character (Unicode code point) at the specified index.		
int		<code>codePointBefore(int index)</code> Returns the character (Unicode code point) before the specified index.		
int		<code>codePointCount(int beginIndex, int endIndex)</code> Returns the number of Unicode code points in the specified text range of this String.		
int		<code>compareTo(String anotherString)</code> Compares two strings lexicographically.		
int		<code>compareToIgnoreCase(String str)</code> Compares two strings lexicographically, ignoring case differences.		
String		<code>concat(String str)</code> Concatenates the specified string to the end of this string.		
boolean		<code>contains(CharSequence s)</code> Returns true if and only if this string contains the specified sequence of char values.		

- ▶ Diese Dokumente gibt es auch für alle bereits existierenden Java-Klassen.

Outline

- 1 Klassen, Objekte, Methoden
- 2 Objekte
- 3 Klassen
- 4 Klassen und Methoden in Java
- 5 Klassendokumentation
- 6 Strings
- 7 Zusammenfassung

Die Klasse String

- ▶ `String` ist ein Objektdatentyp.
- ▶ Das bedeutet: `String` ist eine Klasse. Ein spezifischer Text ist ein Objekt dieser Klasse.
- ▶ Die Klasse `String` bietet einige Methoden (die schon fest implementiert sind, d.h. Sie können sie direkt verwenden), z. B.
 - ▶ `int length()`
 - ▶ `boolean isEmpty()`
 - ▶ `boolean equals(String anotherString)`
 - ▶ `String substring(int beginIndex, int endIndex)`

Methodenaufrufe

- ▶ Um eine Methode eines Objekts aufzurufen schreibt man
`<Objektname>.<Methodename>(<Argumente>*)`
- ▶ Wenn die Methode Parameter erwartet, müssen Argumente für diese Parameter beim Aufruf übergeben werden, in der Reihenfolge und mit den Datentypen, die die Signatur definiert.
- ▶ Wenn die Methode einen Wert zurückgibt, kann dieser Wert in einer Variable des passenden Datentyps gespeichert werden.

```
String text = new String("Hello World!"); // Objekt der Klasse String

int length = text.length(); // 10

boolean test1 = text.equals("Test"); // false
boolean test2 = text.equals(text); // true

String hello = text.substring(0,5); // "Hello"
```

Java-Klassenbibliotheken

Java-Klassenbibliotheken

Die Java-Klassenbibliotheken bilden eine umfangreiche Sammlung von vordefinierten Klassen, die untrennbar mit dem Java-System verbunden sind.

- ▶ Zu allen wichtigen Klassen in Java gibt es sogenannte Klassendokumentationen
- ▶ In diesen finden sich alle Methoden, die ein Objekt dieser Klasse haben kann und was diese bewirken.

Klassendokumentation online:

<http://docs.oracle.com/javase/8/docs/api/>

Outline

- 1 Klassen, Objekte, Methoden
- 2 Objekte
- 3 Klassen
- 4 Klassen und Methoden in Java
- 5 Klassendokumentation
- 6 Strings
- 7 Zusammenfassung

Zusammenfassung: Konzepte

- ▶ Eine **Klasse** ist ein Bauplan für **Objekte**. Objekte haben **Methoden**, bei deren Aufruf eine Aktion ausgeführt wird.
- ▶ Methoden werden unterschieden in **Kommandos** (`void`) und **Abfragen** (mit Rückgabewert).
- ▶ In der **Dokumentation** einer Klasse ist beschrieben, über welche Methoden eine Klasse verfügt und was diese tun.

Zusammenfassung: Syntax

- ▶ Klasse definieren:

```
public class <Klassenname> { ... }
```

- ▶ Methode einer Klasse definieren (ohne Rückgabewert):

```
void <Name> (<Parameter>*) { ... }
```

- ▶ Methode einer Klasse definieren (mit Rückgabewert):

```
<Rückgabetyp> <Name> (<Parameter>*) { ... return <Wert>; }
```

- ▶ Objekt erzeugen:

```
new <Klassenname>(<Argument>*)
```

- ▶ Methodenaufruf auf Objekt:

```
<Objektname>. <Methodename>(<Argument>*)
```

- ▶ Dokumentationskommentar:

```
/** ... */ direkt vor Klassen- oder Methodendefinition.
```



Vielen Dank!



Frank Schweiner

E-Mail frank.schweiner@mint-kolleg.de
Telefon +49 (0) 711 685-84326
Fax —

Universität Stuttgart
MINT-Kolleg Baden-Württemberg
Azenbergstr. 12
70174 Stuttgart