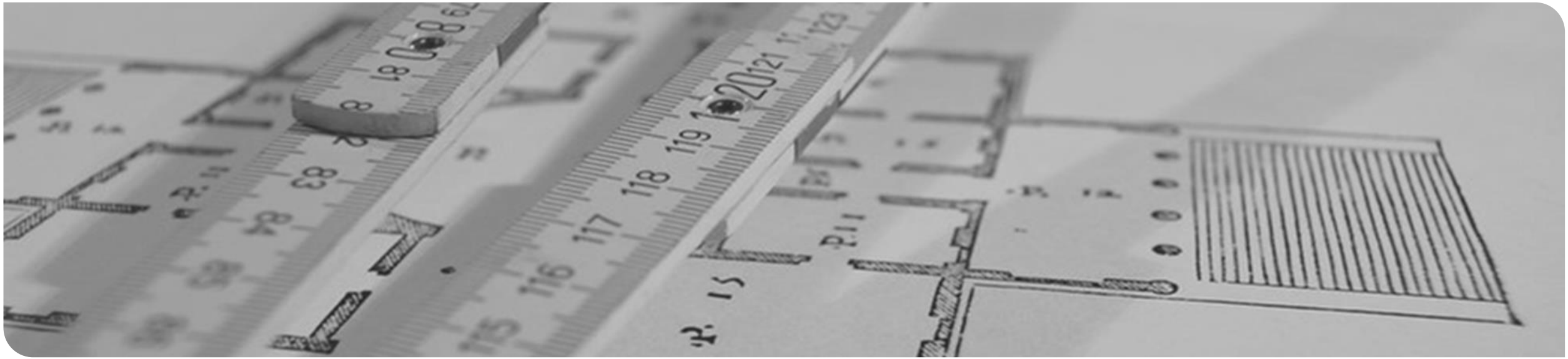
Karlsruher Institut für Technologie

Vorlesung Programmieren

1.2 Objekte und Klassen

Prof. Dr.-Ing. Anne Koziolek



Fragen?

Beitreten über

slido.com

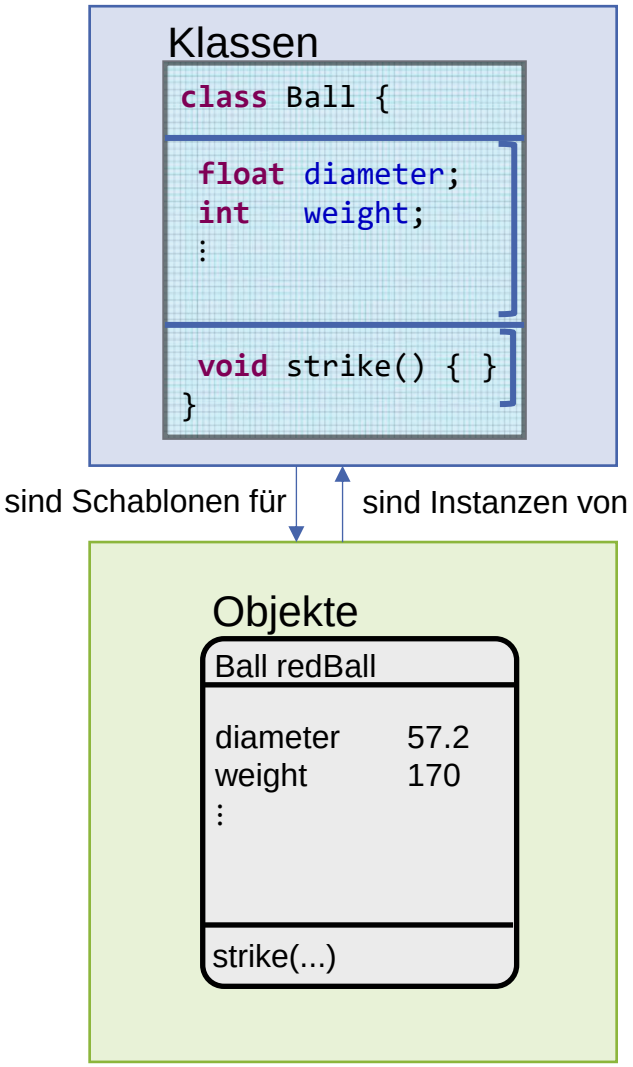
#Prog

Semesterplan für die Vorlesung

21.10.2024	Erstsemesterbegrüßung: Einführung
23.10.2024	Organisatorisches; Ein Einfaches Programm, Objekte und Klassen
30.10.2024	Typen und Variablen
06.11.2024	Kontrollstrukturen (+Scanner)
13.11.2024	Konstruktoren und Methoden
20.11.2024	Arrays; Konvertierung, Datenkapselung, Sichtbarkeit
27.11.2024	Listen und Abstrakte Datentypen
04.12.2024	Vererbung **~~Audimax~~ 50.34 Raum -101 und Raum -102**
11.12.2024	Exceptions; Interfaces
18.12.2024	Generics; Rekursion
08.01.2025	Java-API; Objektorientierte Design-Prinzipien
15.01.2025	Best Practices; Finden und Beheben von Fehlern
22.01.2025	Testen und Assertions
29.01.2025	JUnit; Parsen, Suchen, Sortieren
05.02.2025	Vom Programm zur Maschine; Ausblick auf zukünftige Lehrveranstaltungen
12.02.2025	Wrap-Up

Lernziele

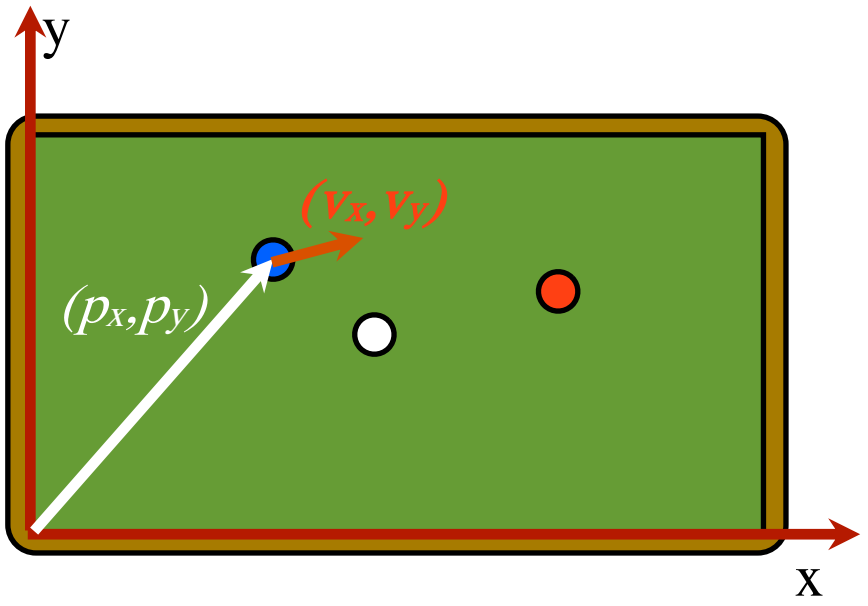
- Sie können für ein gegebenes Problem…
  - Objekte und Klassen identifizieren
  - Klassen beschreiben durch
    - Attribute
    - Methoden
- Sie können Klassen in Java ausdrücken
- Sie können neue Objekte einer Klasse erzeugen



Billard-Spiel

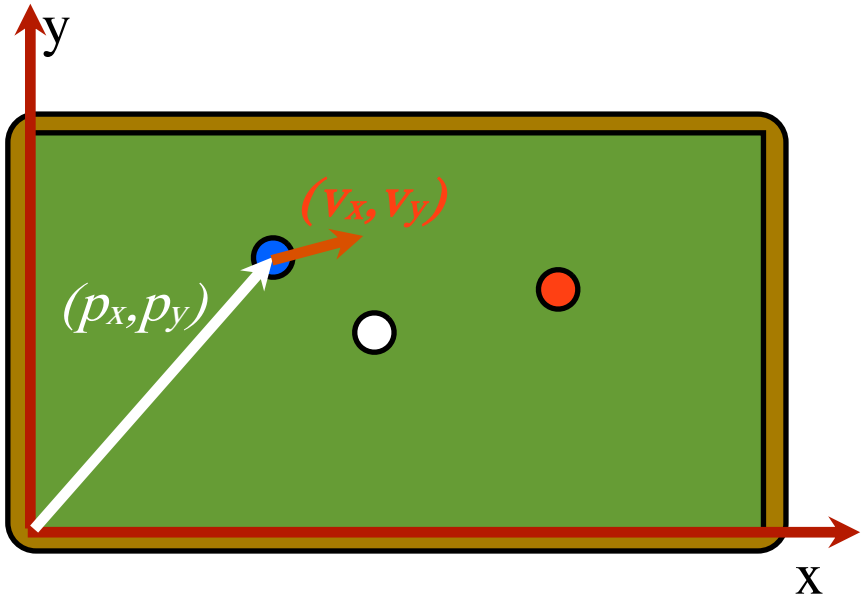
- Programmieren ist nur im Endergebnis die Definition von Befehlen und deren Ausführungsreihenfolge.

- Vorher kommt die **Modellierung** des Problems
  - Welche Objekte sind wichtig?
  - Welche Eigenschaften haben diese Objekte?
  - Welches Verhalten sollen die Objekte haben?



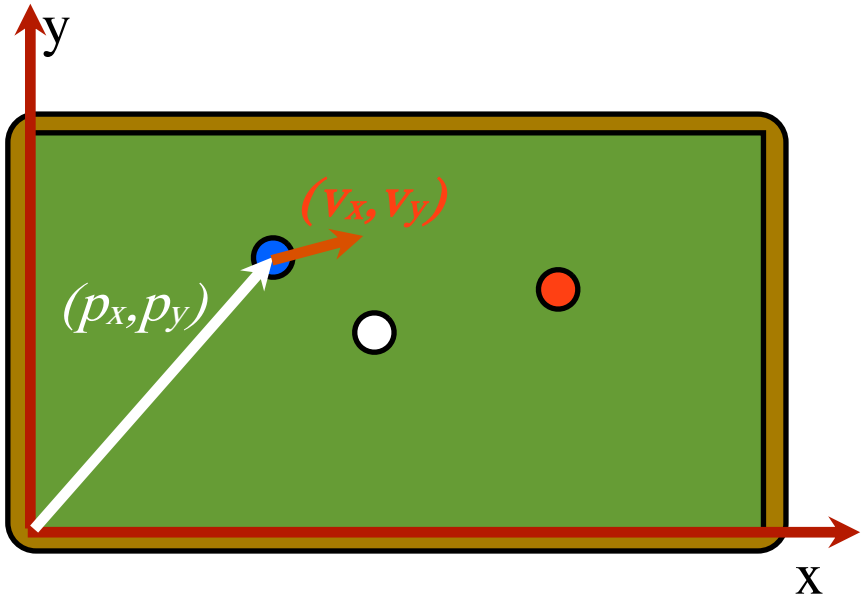
Billard-Spiel

- Angenommen, wir wollen ein Billard-Spiel programmieren
- Erster Teil: Simulation der Bewegung der Kugeln
- Was muss modelliert werden?
  - Tisch
    - Größe
  - Kugeln: jede Kugel hat (zu jedem Zeitpunkt):
    - Eine Position auf dem Tisch
    - Eine Bewegungsrichtung und Geschwindigkeit
    - Evtl. Effet (Drall, engl.: Spin)
    - Kugeln können aneinanderstoßen und ändern dann die Bewegungsrichtung

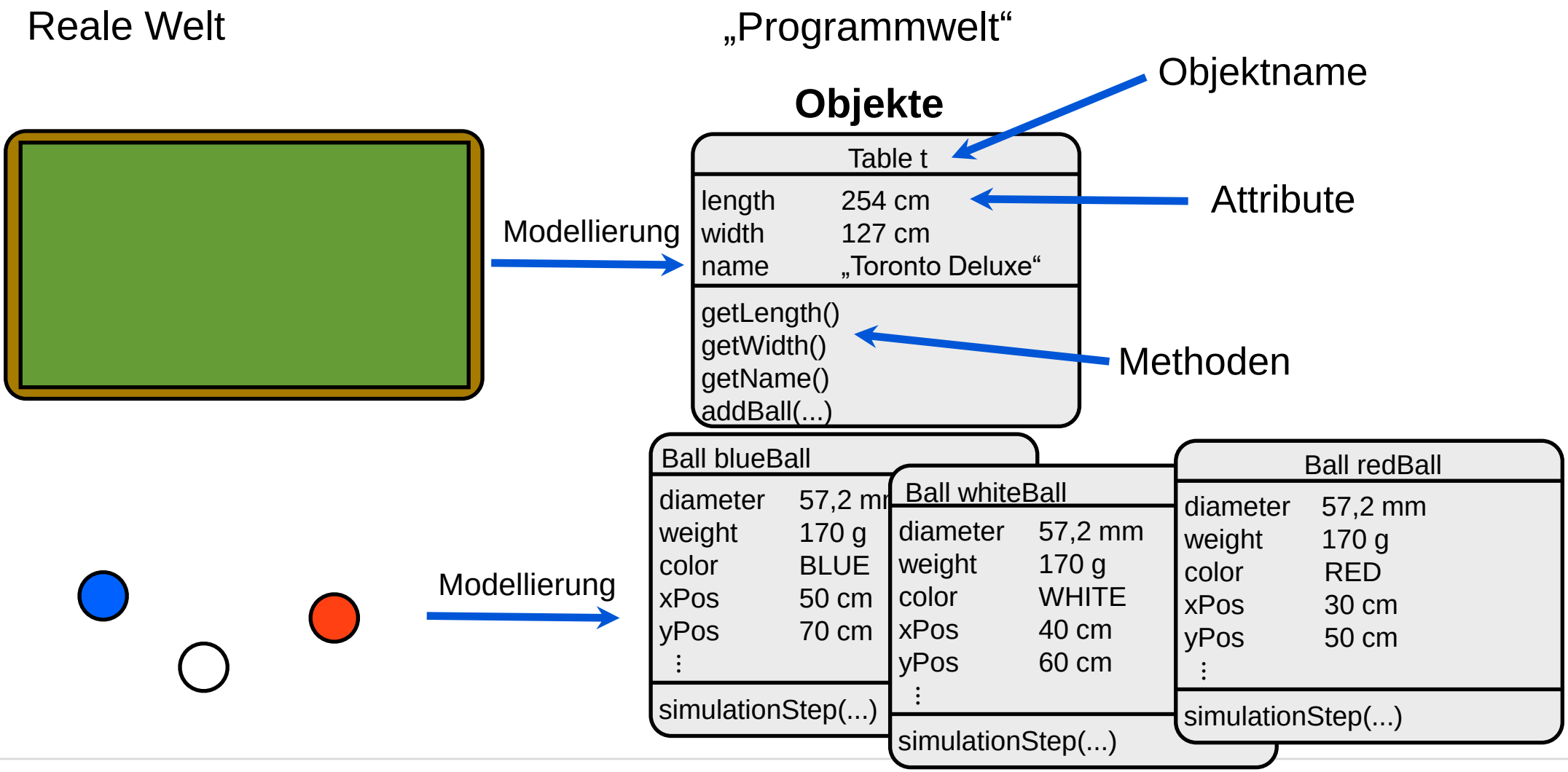


Objektorientierte Programmierung (OOP)

- Grundidee der Objektorientierten Programmierung:
  - Jedes Objekt der Realität hat ein virtuelles Gegenstück
  - Objekte kooperieren durch Datenaustausch
- Im Zentrum: Modellierungsfrage
  - „Wer macht was?“
    (Im Gegensatz zu: „Wie wird’s gemacht?“)
- Gebräuchliche Benennungen für Datenaustausch:
  - „Objekt A sendet Nachricht an Objekt B“
  - „Objekt A ruft eine Methode von Objekt B auf“

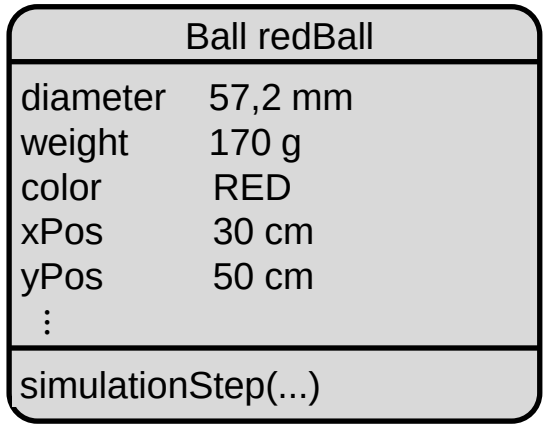


OOP am Beispiel „Billard-Spiel“



Klassen: Der „Bauplan“ von Objekten

Objekt



Klasse (Java)

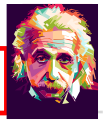
```java
class Ball {
  // Attribute
  float diameter; // in mm
  int weight; // in Gramm
  ...
  float xPos; // in cm
  float yPos; // in cm
  ...
  // Methoden
  void simulationStep() { }
}
```

- Aus einer Klasse können beliebig viele Objekte („Objekt-Instanzen“, „Instanzen der Klasse“) dieses „Bauplans“ erzeugt werden.

Definition: Objekte und Klassen

- Ein Objekt wird durch drei Aspekte charakterisiert:
  - Identität
    Objekteigenschaften können sich ändern, die Identität des Objekts bleibt aber gleich
    - Durch Objektnamen sichergestellt
  - Zustand
    Menge der Eigenschaften des Objekts zu einem Zeitpunkt
    - Durch Attribute realisiert
  - Verhalten
    Ausführen von Aktionen, die den Zustand ändern können
    - Durch Methoden realisiert
- Eine **Klasse** ist ein „Bauplan“ für gleichartige Objekte und legt fest
  - welche Attribute die Objekt-Instanzen der Klasse haben
  - welche Methoden die Objekt-Instanzen der Klasse haben

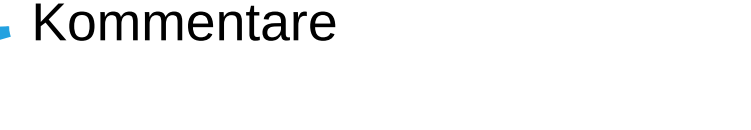
Grady Booch, Object Oriented Analysis and Design



Klassen in Java

```
class Ball {
  // Attribute
  float diameter; // in mm
  int weight; // in Gramm
  ...
  float xPosition; // in cm
  float yPosition; // in cm
  float xVelocity; // in m/s
  float yVelocity; // in m/s

  // Methoden
  ...
}
```



Zwei Arten von Kommentaren in Java:

```
// Zeilenkommentar

/*
längerer Kommentar
über mehrere
Zeilen
*/
```

- Schema: `class Name { Klassenrumpf }`
- „Klassen-Deklaration“

Ist die Klassendefinition gut so?

```
class Ball {
  // Attribute
  float diameter; // in mm
  int weight; // in Gramm
  ...
  float xPosition; // in cm
  float yPosition; // in cm
  float xVelocity; // in m/s
  float yVelocity; // in m/s

  // Methoden
  ...
}
```

Wir haben zweimal fast dasselbe Konstrukt. Lässt sich das vereinfachen?

Ja!

Abstraktion der x/y-Paare in eine neue Klasse `Vector2D`.

Abstraktion: Klasse für Koordinatenpaare

```
class Ball {
  // Attribute
  float diameter; // in mm
  int weight; // in Gramm
   ...
  Vector2D position; // in cm
  Vector2D velocity; // in m/s

  // Methoden
  ...
}
```

```
class Vector2D {
   // Attribute: x/y-Koordinaten
   float x;
   float y;

   // Methoden
   ...
}
```

- Was haben wir gewonnen?
  - Kürzere, übersichtlichere Klassendeklaration
  - Strukturierung, „Kapselung“ von Daten

Wie erzeugt man Objekte?

- Objekte werden mit dem **new**-Operator erzeugt

Beispiele:

```
Vector2D position = new Vector2D(); // erzeugt Vektor „position“
Vector2D velocity = new Vector2D(); // erzeugt Vektor „velocity“
Ball ball = new Ball(); // erzeugt Billardkugel
```

Jetzt sind Sie gefragt!

- Wir modellieren eine Bibliothek für ein Ausleihsystem für Medien

„Bauplan“ für gleichartige Objekte

- Welche **Klassen** benötigen wir?
  - Welche **Attribute**?
  - Welche **Methoden**?

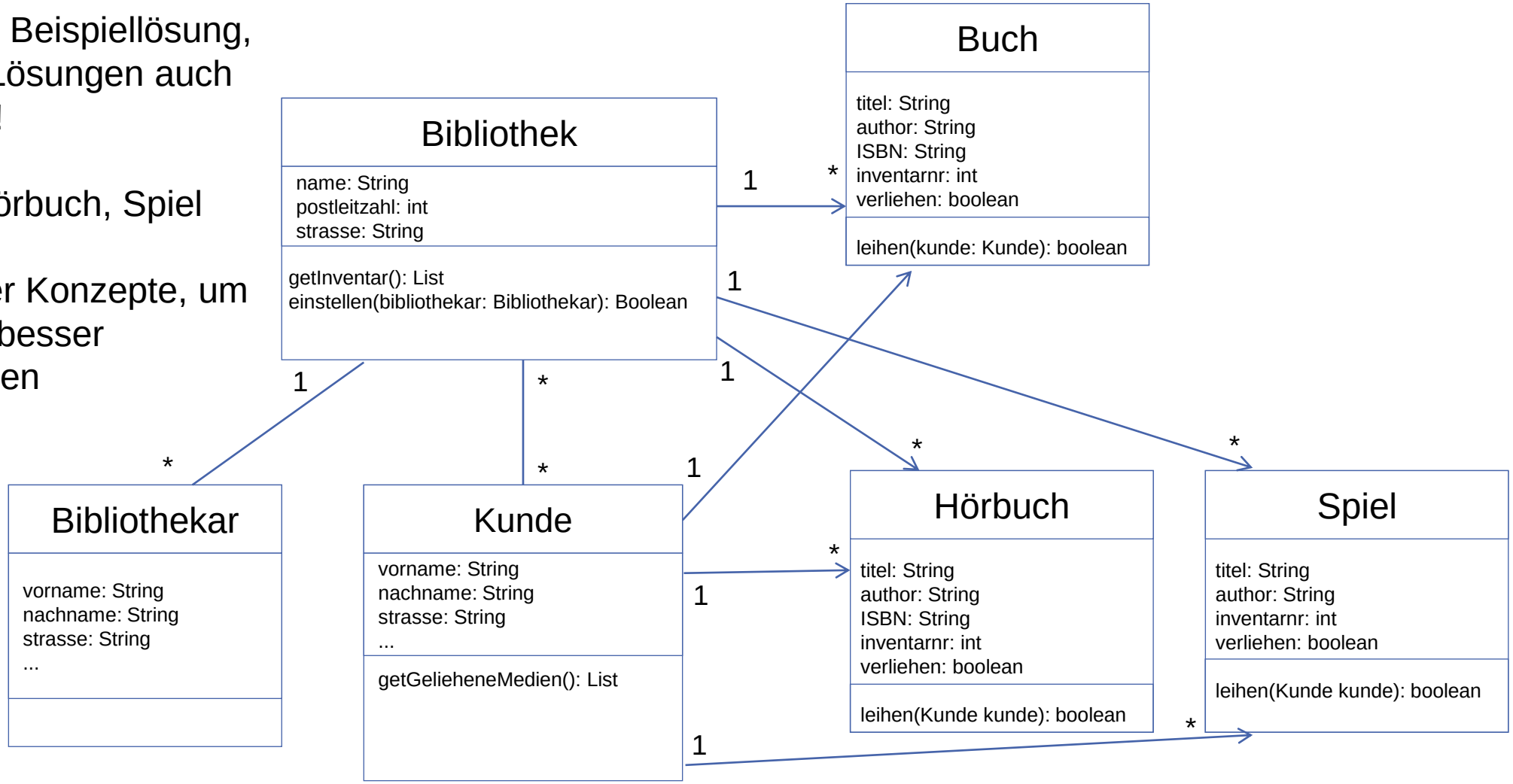
Eigenschaften eines Objekts (Zustand)

Aktionen, welche den Objekt-Zustand ändern können (Verhalten)

Wir modellieren eine Bibliothek!

Nur eine Beispiellösung, andere Lösungen auch denkbar!

Buch, Hörbuch, Spiel unschön
→ Später Konzepte, um Entwurf besser zu machen



Jetzt sind Sie gefragt!

- Schreiben Sie eine main-Methode für die Klasse Vector2D, die die Koordinaten eines Vektors ausgibt.

```
class Vector2D {
  // attributes:
  // x/y-coordinates
  float x;
  float y;
}
```

Jetzt sind Sie gefragt!

- Schreiben Sie eine main-Methode für die Klasse Vector2D, die die Koordinaten eines Vektors ausgibt.

```
class Vector2D {
  // attributes:
  // x/y-coordinates
  float x;
  float y;
}
```

```
class Vector2D {
  // attributes: x/y-coordinates
  float x;
  float y;

  // main method (for test)
  public static void main(String args[]) {
    Vector2D v = new Vector2D();
    System.out.println("v.x = " + v.x);
    System.out.println("v.y = " + v.y);
  }
}
```

Ursprung Objektorientierter Programmierung

- 1960er Jahre: Ole-Johan Dahl (1931-2002) und Kristen Nygaard (1926-2002) entwickeln die Programmiersprache **Simula**
  - genauer: erst Simula I, danach Simula 67
- Simula steht für SIMulation LAnguage, eine Programmiersprache für Simulationen
- Erste Programmiersprache, die zwischen Klassen und Objekten unterscheidet
- Viele weitere Konzepte aus Simula finden sich in heutigen objektorientierten Sprachen wieder

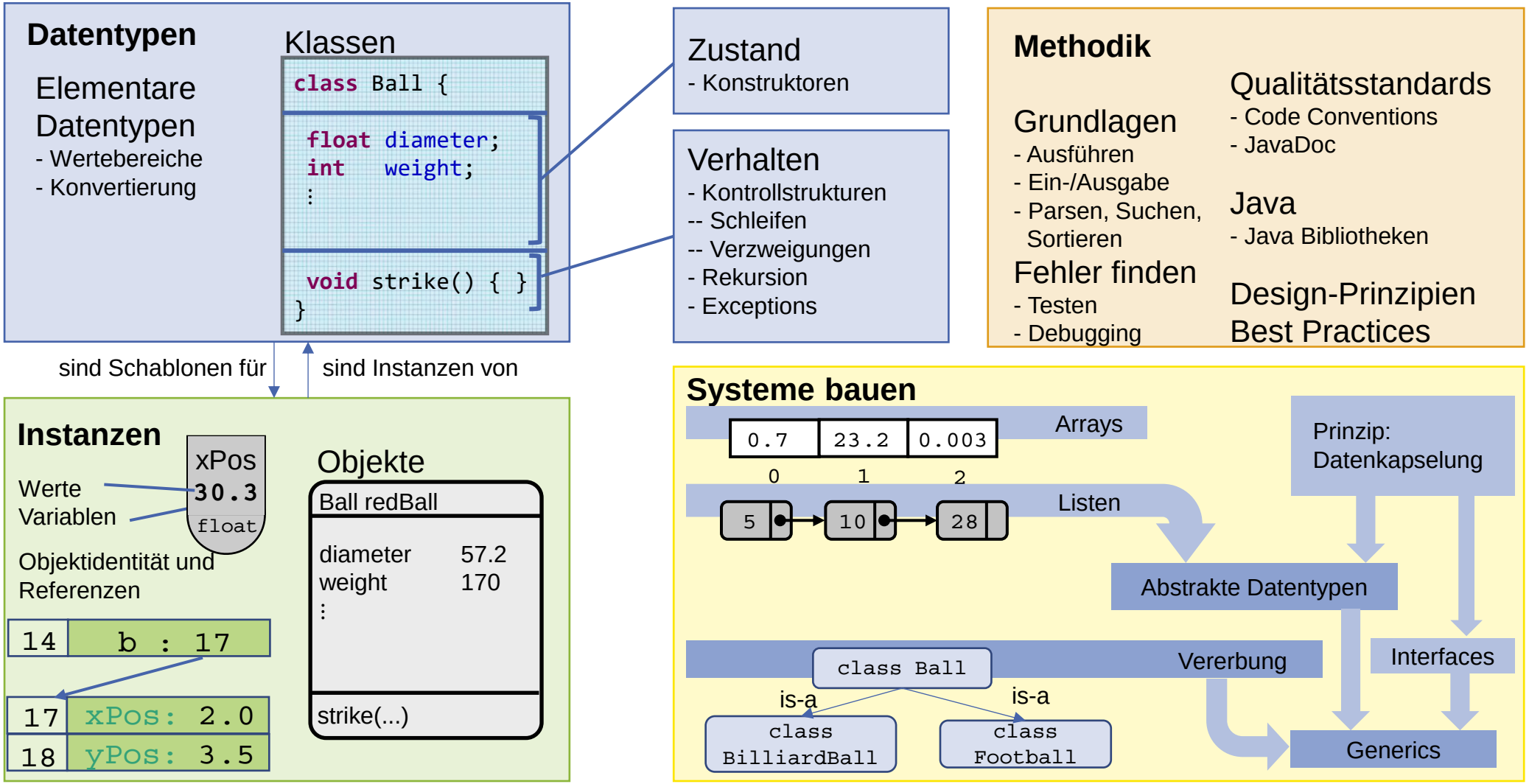
Zusammenfassung

- Objekte modellieren Gegenstände der realen Welt
- Klassen sind „Baupläne“ für Objekte
- Klassen in Java werden deklariert mit

```
class Name { Klassenrumpf }
```

- Neue Objekte einer Klasse werden mittels **`new`** erzeugt
- Klassen erlauben eine Erweiterung der eingebauten Java-Datentypen

Vorlesungsüberblick
Objektorientierte Programmierung in Java



ANHANG