Objektorientes Design & Programm

- Ziele
  - Motivation für OO-Design
  - Softwarekomponenten mit Objektbeschreibungen
  - Klassen und Objekte
  - Konstruktoren zum Erzeugen von Objekten

Motivation

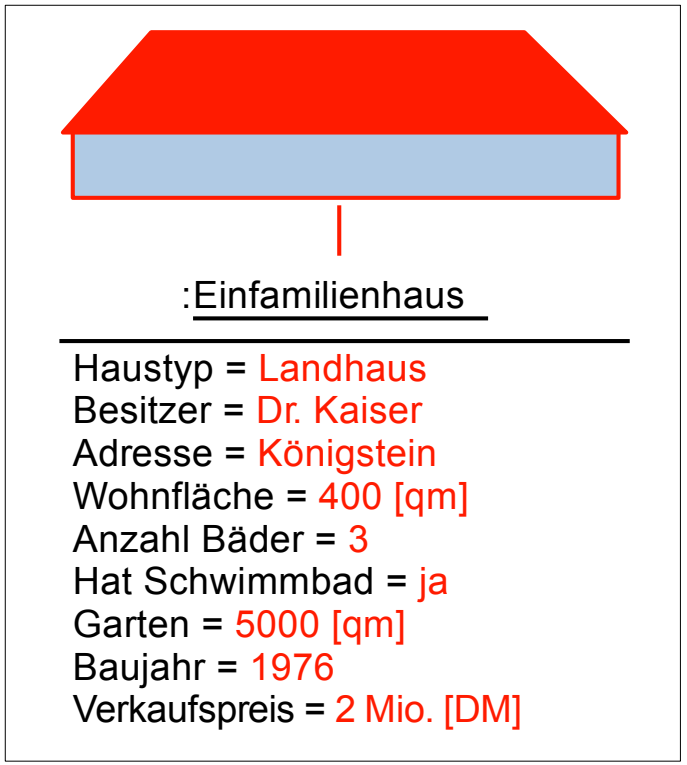
- Vermischung von Variablendeklarationen und Methoden (Prozeduren) verschiedener Themen in einer Datei führte zu Spaghetticode
  - Nicht wartbar, Nicht debugbar, Nicht erweiterbar
  - Nicht für Außenstehende verstehbar
- In OO-Design
  - Geheimnis- & Kapselungsprinzip
  - Seperation nach Themen
  - Sehr hohe Wiederverwertbung durch Erweiterung (Recycling) bestehender Dateien

Objekte

- Objekt
  - Allgemein:
    - Gegenstand des Interesses, insbesondere einer Beobachtung, Untersuchung oder Messung
  - Objekt-Orientiert:
    - Objekt ist ein individuelles Exemplar von Dingen:
      - Roboter, Auto
      - Personen (z.B. Kunde, Mitarbeiter)
      - Begriffe der realen Welt (z.B. Bestellung)
      - Begriffe der Vorstellungswelt (z.B. juristische und natürliche Personen).

Beispiel Haus

- Beispiel: Immobilienfirma Nobel & Teuer
  - Firma Nobel & Teuer vermittelt exklusive Einfamilienhäuser
- Objekt
  - Jedes Einfamilienhaus ist ein Objekt
  - Ein Objekt besitzt Variablen in denen objektspezifische Zustände gespeichert werden



Beispiel Haus

- Operation (Methode)
  - Bilden Verhalten ab
  - Zum Manipulieren und Abfragen von Objektvariablen
  - z.b. zur Auskunft des Verkaufspreises

:Einfamilienhaus

Haustyp = Landhaus
Besitzer = Dr. Kaiser
Adresse = Königstein
Wohnfläche = 400 [qm]
Anzahl Bäder = 3
Hat Schwimmbad = ja
Garten = 5000 [qm]
Baujahr = 1976
Verkaufspreis = 2 Mio. [DM]

anfragen Verkaufspreis

Task Klasse Einfamilienhaus

- Erstellen Sie die Java-Klasse Einfamilienhaus und deklarieren Sie die benötigten Zustandsvariablen (Definition nicht notwendig)

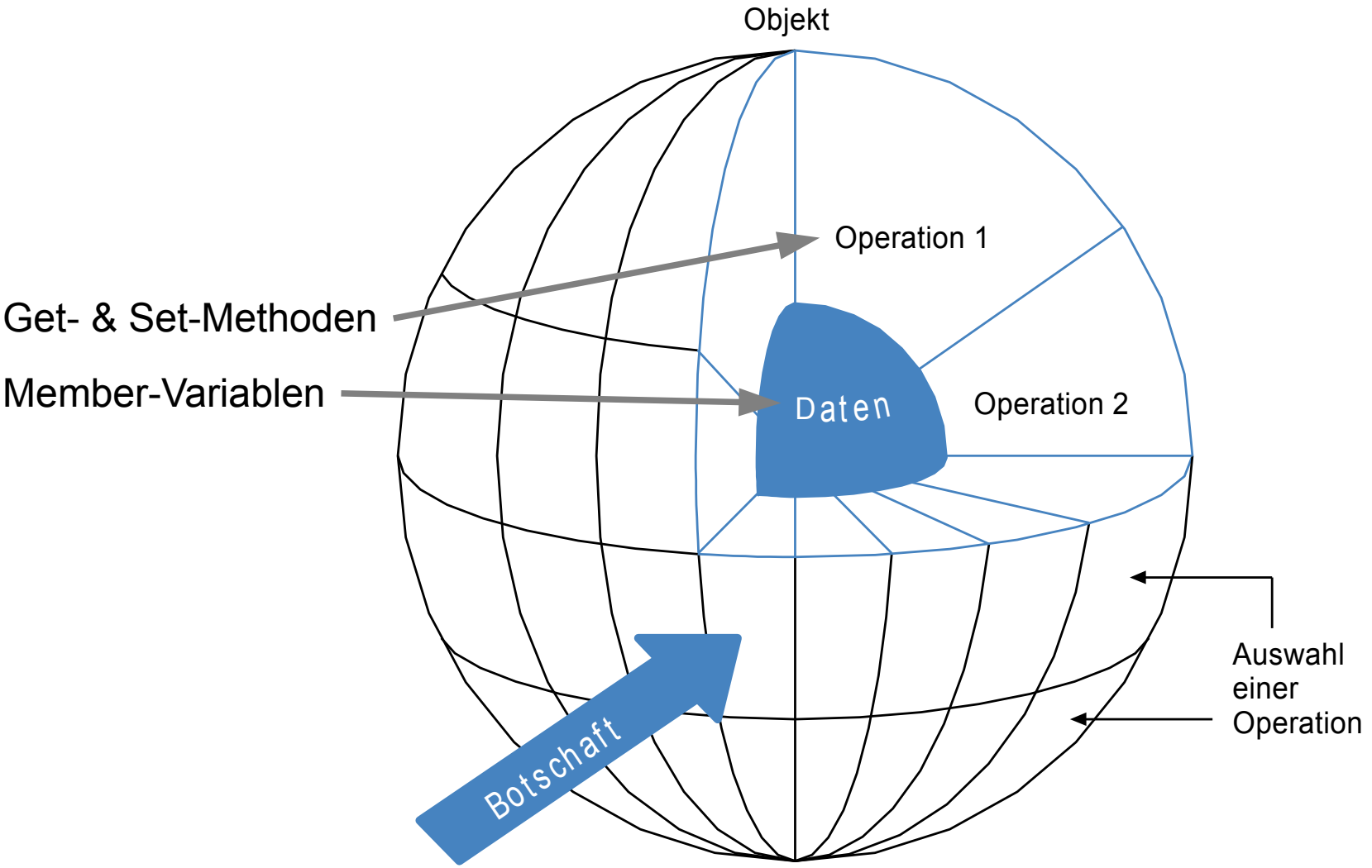
```
public class Einfamilienhaus
{
	public String mHaustyp;
	public int mAnzahlBaeder;
	….

}
```

Beispiel Haus Klasse & Verhalten

- Klasse
  - ...definiert die Attribute und Operationen ihrer Objekte
  - Verhalten: Ausführung der Operation »anfragen Verkaufspreis« für ein spezielles Objekt »Landhaus«
- Verhaltensmethode / Funktion
  - ...aktiviert eine Operation
  - Die gewünschten Ausgabedaten werden an den Aufrufer der Funktion zurückgegeben

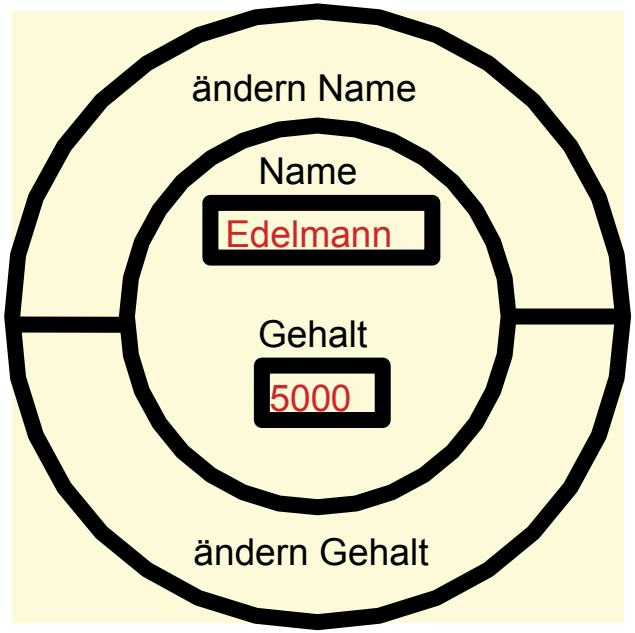
Kapselung der Daten

- Geheimnisprinzip



Reale Objekte

- Beispiel Mitarbeiterkartei
  - Die Werte für die Zustandsspeicher Name und Gehalt dieses Objekts werden durch die Attributwerte „Edelmann“ und „5000“ bestimmt
  - Das Verhalten (ändernName und ändernGehalt) wird durch 2 Operationen bestimmt

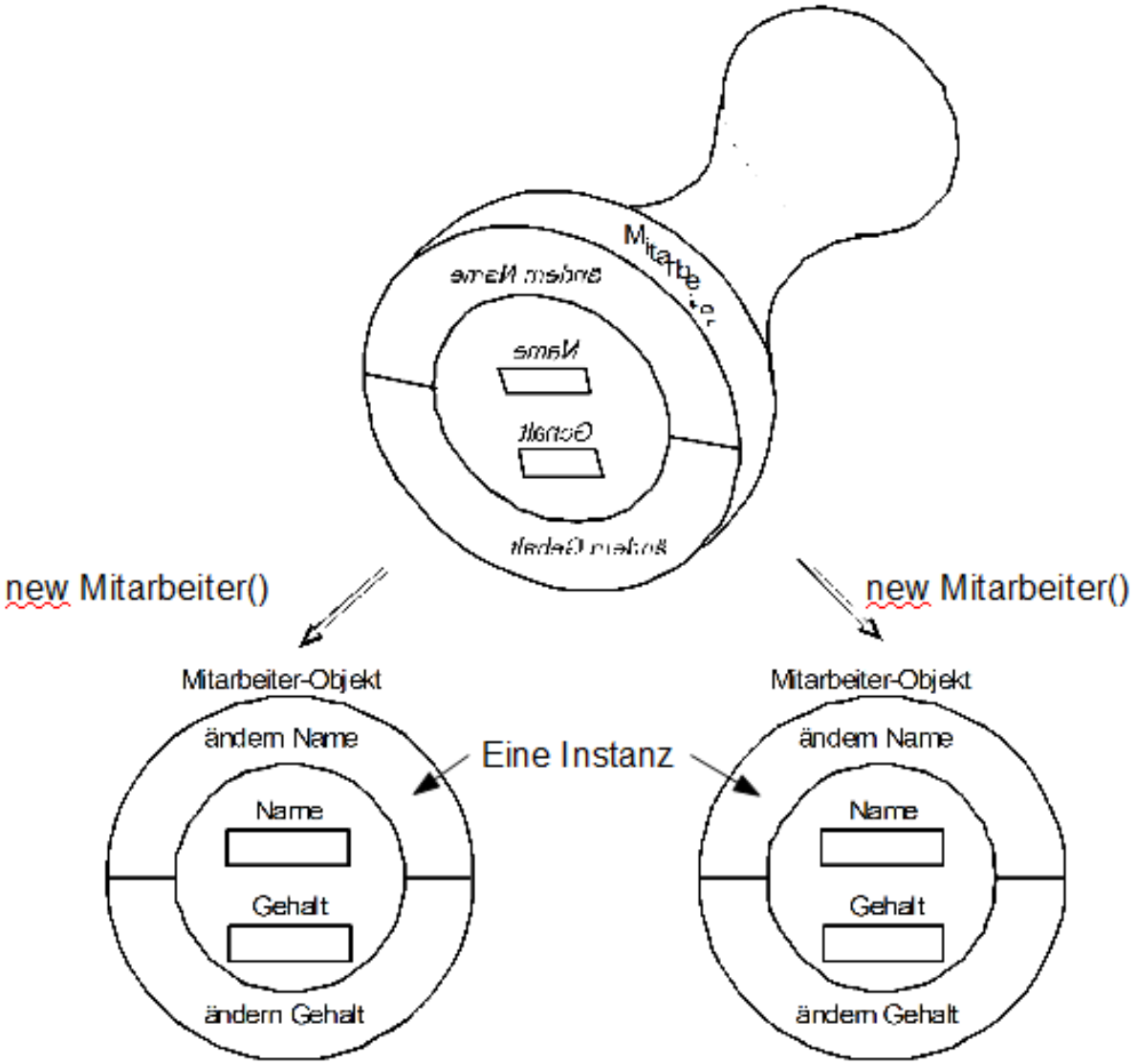


Unterschied Objekt versus Klasse

- Klasse ist ein Bauplan und besteht nur als nicht-reale Beschreibung von Verhalten und Zustandsspeichern
- Ein Objekt ist eine realisierte Instanz der Klasse
  - Die zum gehörenden Verhalten / Zustandsspeicher können gefüllt, geleehrt, genutzt werden

Klassen

- Klasse als Schablone



Klassen Aufbau

- Klassenname
- Klassenrumpf
  - Attributdeklarationen und -Definitionen
  - Methodendeklarationen
    - Constructor-Methode
    - Weitere Verhaltensmethoden

Klassen Constructor

- Objekterzeugung
  - Jede Klasse muß durch Konstruktoren festlegen, wie Objekte von ihr erzeugt werden können
- Konstruktor
  - Operationsname = Klassenname
- `public`
  - Muß vor dem Operationsnamen angegeben werden damit von anderen Java-Programmen der Konstruktor aufgerufen werden kann
- Initialisierung
  - Parameterliste mit Initialisierungswerten (optional)

Klassen einfache Deklaration

Beispiel mit Defaultconstructor

```
public class Kunde
{
      public String Name;


public static void main(String[] args)
{
//Instanz erzeugen
    Kunde meinKunde=new Kunde();
}
}
```

Klassen Constructor

- Beispiel mit explizitem Constructor

```
public class Kunde
{
    public String mFirmenname;

 public Kunde(String aName)
 {
   mFirmenname = aName;
 }
}
//Instanz erzeugen
Kunde meinKunde=new Kunde(„Hans“);
```

Klassen Constructor

Objekte durch Konstruktor-Aufruf erzeugen

- Gewünschte Konstruktor wird mit vorangestellten Operator `new` aufgerufen

```
new Kunde()
new Kunde("Karl")
new Kunde("Heinz")
```

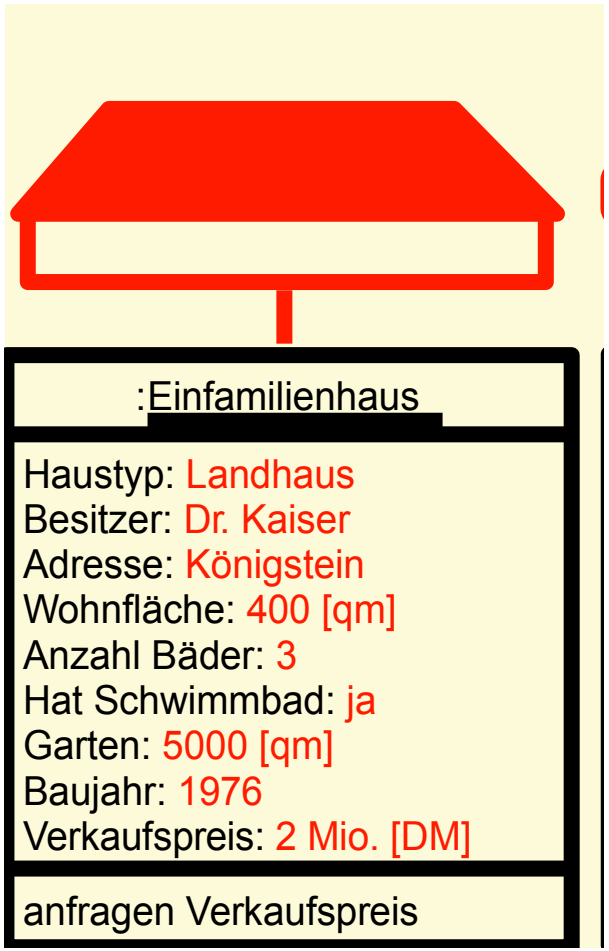
Speichern der Objektreferenz

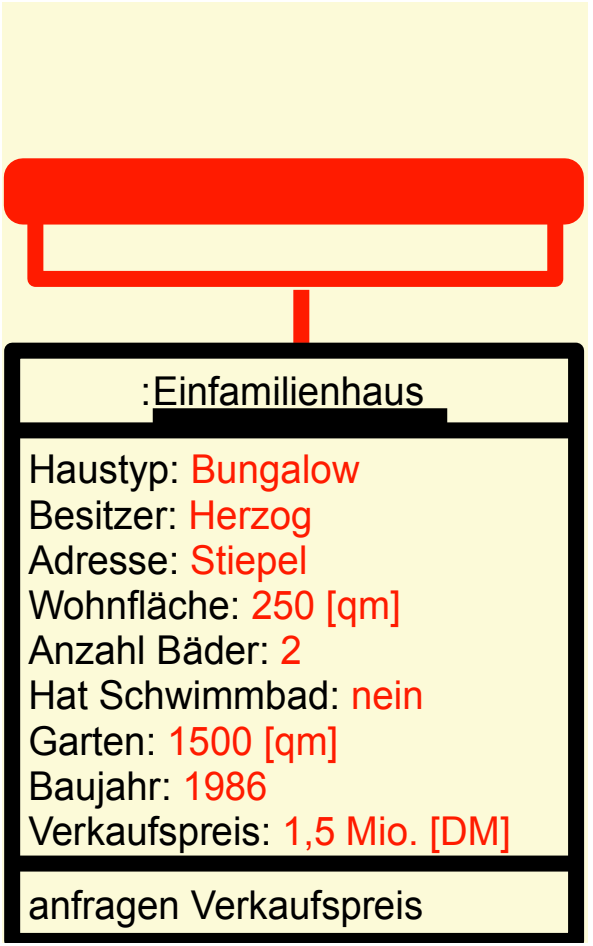
```
ersterKunde = new Kunde("Tom");
```

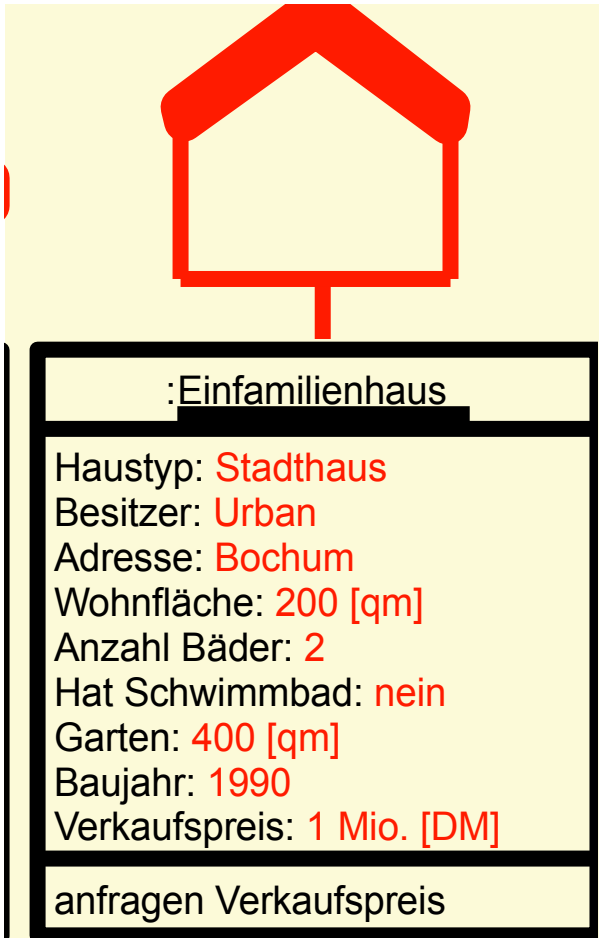
Die zurückgegebene Speicheradresse wird in der Speicherzelle `ersterKunde` gespeichert

Task Objekte vom Typ Einfamilienhaus

- Erstellen/Instantiieren Sie 3 Objekte vom Typ Einfamilienhaus







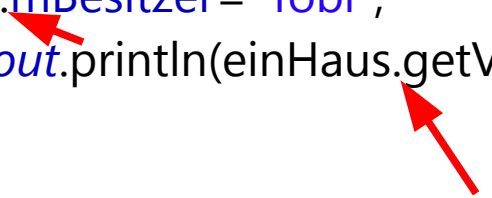
Objekte

- Charakteristika
  - Jedes Objekt besitzt eine Objekt-Identität, die es von allen anderen Objekten unterscheidet
  - Keine 2 Objekte besitzen gleiche Identität, selbst bei identischen Attributwerten
    - Unterscheidungsmerkmal ist Position im Speicher
  - Zustand *(state)* eines Objekts
    - Bestimmt durch Attributwerte bzw. Daten und Verbindungen zu anderen Objekten
  - Verhalten *(behavior)* eines Objekts
    - Beschreibung durch Menge von Operationen
- Synonyme für "Objekt"
  - Exemplar, Instanz, Klasseninstanz

Zugriff auf Objekteigenschaften

- Der Punktoperator gestattet den Zugriff auf Attribute und das Aufrufen von Objektmethoden

```
public class Einfamilienhaus {

    public String mBesitzer;

    public int getVerkaufspreis()
    {
        return 200000;
    }

    public static void main(String[] args) {
        Einfamilienhaus einHaus=new Einfamilienhaus();
        einHaus.mBesitzer="Tobi";
        System.out.println(einHaus.getVerkaufspreis());
    }
}
```



Klassen und Ihre Objekte

- Objekt kennt seine Klasse
  - Ein Objekt kann i. allg. zur Laufzeit ermitteln, zu welcher Klasse es gehört
  - In Java: `getClass()`
  - Boolscher Vergleich `instanceOf`
  - Operationen sind der Klasse zugeordnet
    - Da alle Objekte zwar unterschiedliche Attributwerte, jedoch gleiche Operationen besitzen

Klassen und Ihre Objekte

- Klasse kennt Objekte nicht
  - Klasse »weiß« im allgemeinen nicht, welche Objekte sie »besitzt« bzw. welche Objekte von ihr erzeugt wurden
  - Wenn diese Eigenschaft notwendig ist, muß sie im Einzelfall »von Hand« hinzugefügt werden.

Objekte/Rollen in UML

- UML-Notation
  - Objekt: zweigeteiltes Rechteck
  - Oberer Teil: enthält den unterstrichenen Klassennamen, mit vorangestelltem Doppelpunkt, zu dem das Objekt gehört
  - Besitzt das Objekt einen eigenen Namen, dann steht dieser vor dem Doppelpunkt
    - Z.B. einLandhaus: Einfamilienhaus
  - Ist die Klasse aus dem Kontext ersichtlich, dann genügt auch der unterstrichene Objektname
  - Klassenname beginnt mit Großbuchstaben
  - Objektname beginnt mit Kleinbuchstaben.

UML Klassendiagram

- Von UML-Klassen zu Java-Klassen