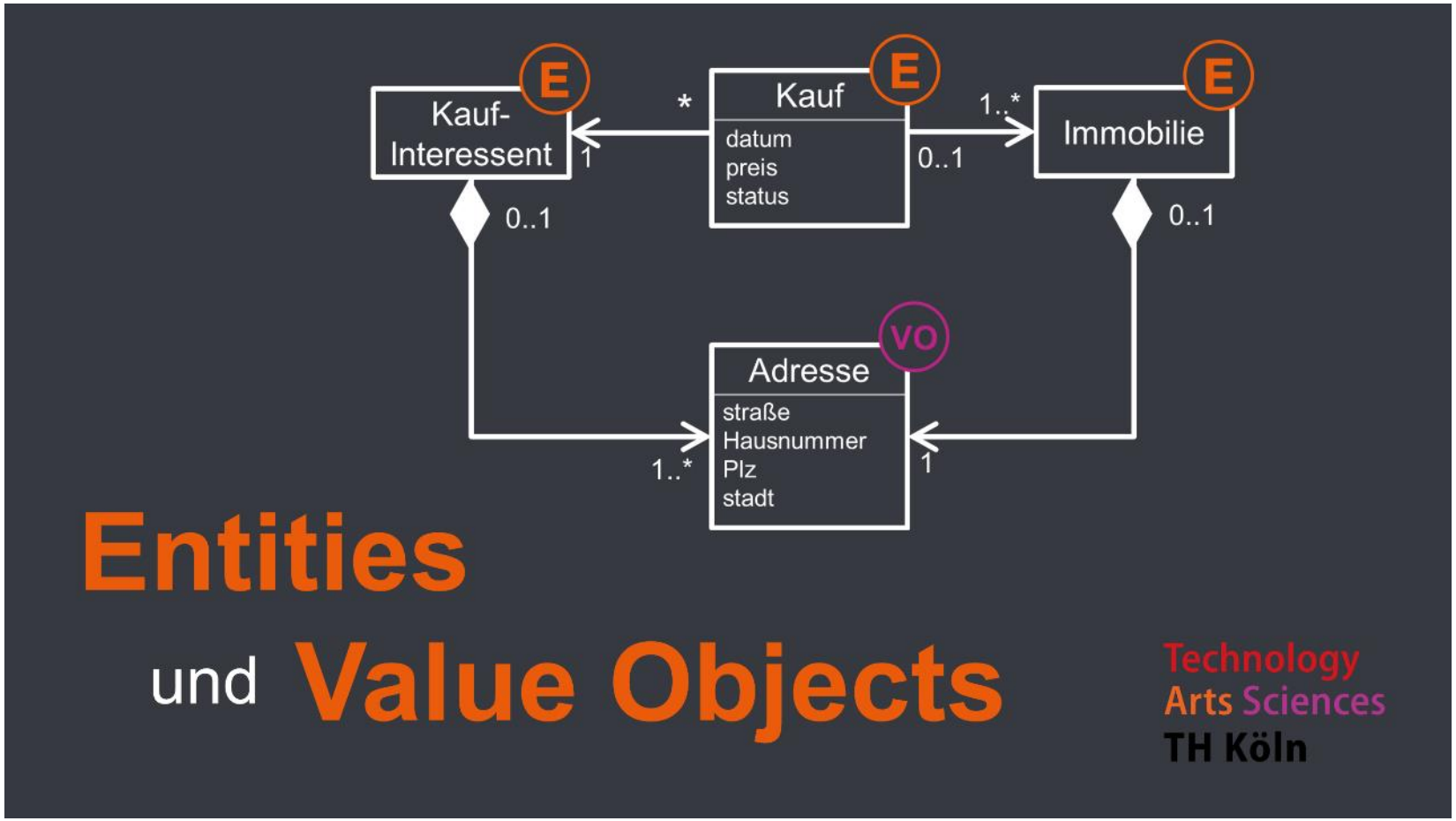
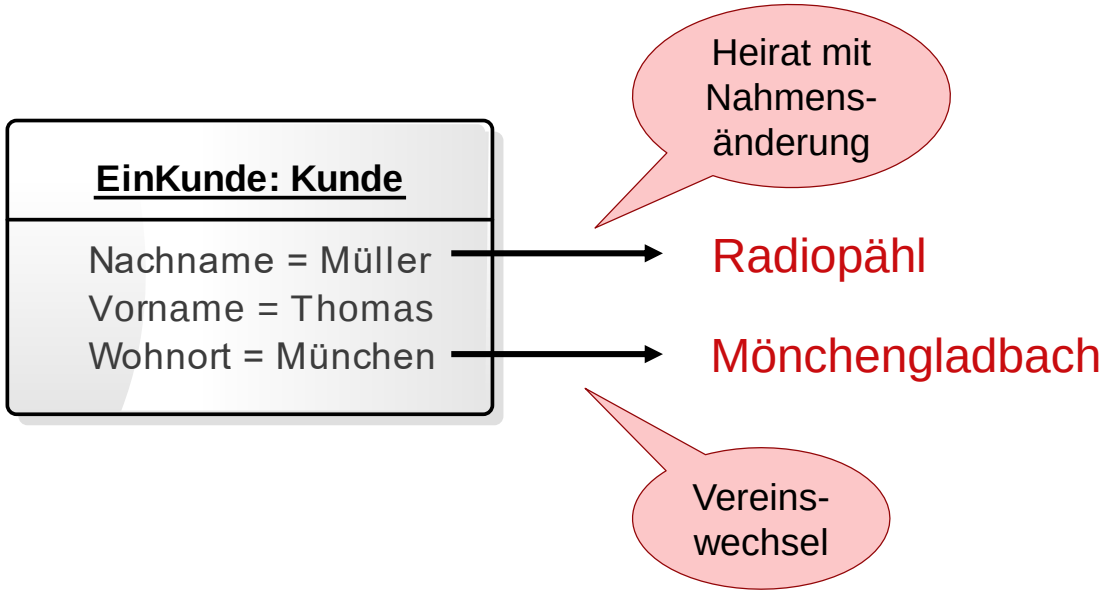


https://www.youtube.com/watch?v=pYg7I3UJ3Ls

Entitäten (Entities)

- **Entities** (Entitäten): Objekte mit unveränderlicher Identität
  - auch wenn Attributwerte sich ändern, bleibt Objekt "es selbst"



Bilder:
- https://www.unitedcharity.de/PROMINENTE/Thomas-Mueller
- Sascha Kohlmann, CC BY-SA 2.0, https://www.flickr.com/photos/skohlmann/8844560880

Entities haben Identität, auch bei gleichen Attributwerten

EinKunde: Kunde

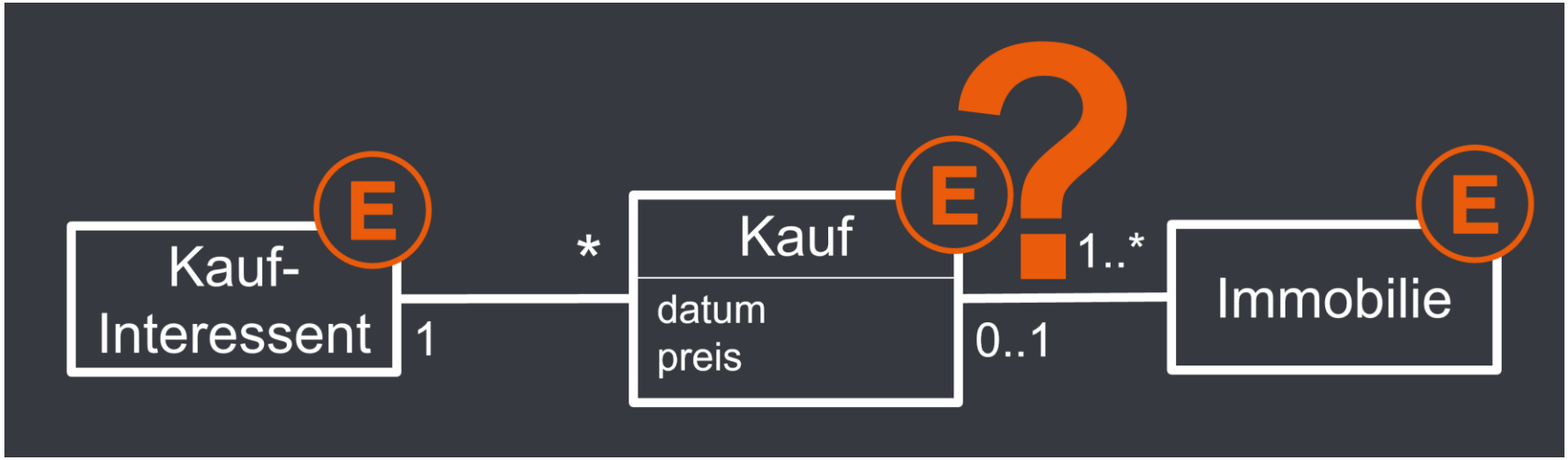
Nachname = Müller
Vorname = Thomas
Wohnort = München

NochEinKunde: Kunde

Nachname = Müller
Vorname = Thomas
Wohnort = München

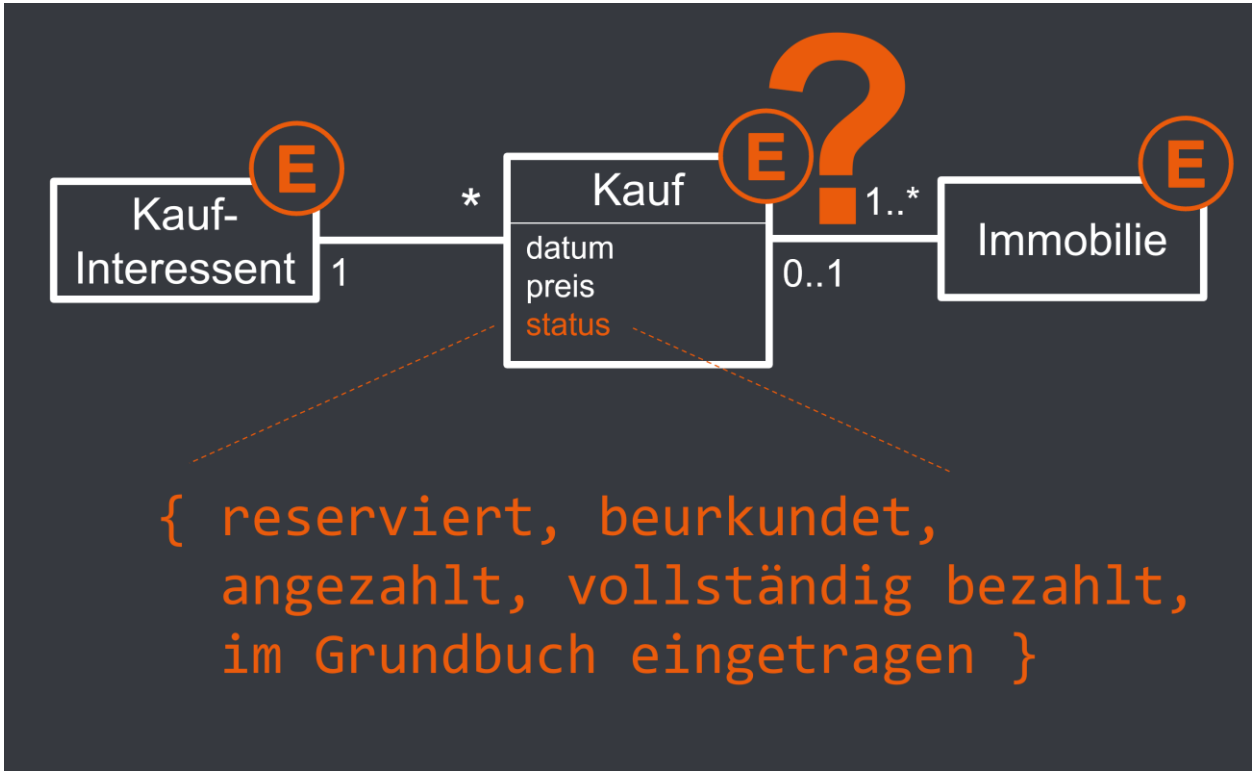
Bilder:
- https://www.unitedcharity.de/PROMINENTE/Thomas-Mueller
- Sascha Kohlmann, CC BY-SA 2.0, https://www.flickr.com/photos/skohlmann/8844560880

Ist alles ein Entity?



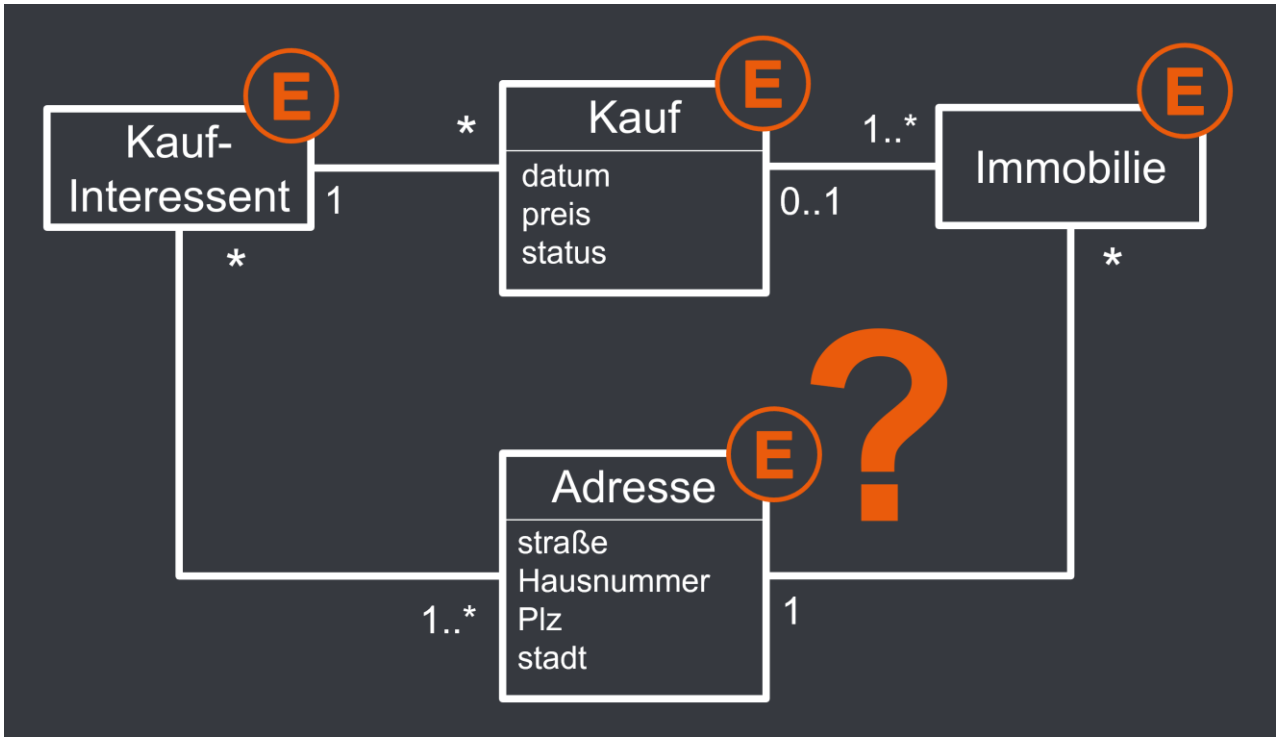
- **KaufInteressent** ist eine Person. Aus dem Beispiel von eben kann man ableiten, dass das ein Entity ist.
- Gleiches gilt auch für eine **Immobilie**. Ich kann z.B. eine Garage an die Immobilie anbauen. Dadurch verändern sich Attributwerte wie “Größe”. Es bleibt aber dieselbe Immobilie.
- Für den Kauf müssen wir uns dieses genauer ansehen – auf der Basis der beiden Attribute “datum” und “preis” kann man die Frage noch nicht gut beantworten.

Ist „Kauf“ ein Entity?



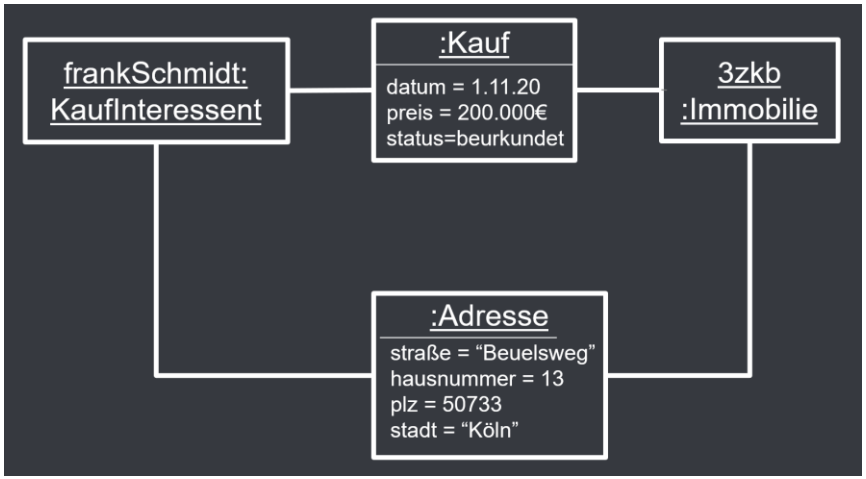
- Ein Immobilienverkauf ist kein *punktueller* Vorgang, sondern durchläuft verschiedene Stadien. Wir führen hierfür ein Statusfeld ein. (*Anmerkung*: die Statuswerte sind nicht groß recherchiert, sondern eher grob geschätzt – das reicht für unser Beispiel.)
- Damit erkennt man schon, dass sich der Status von “angezahlt” auf “vollständig bezahlt” ändern kann, und es bleibt doch derselbe Kauf(-vorgang).
- **Also ist Kauf auch ein Entity.**

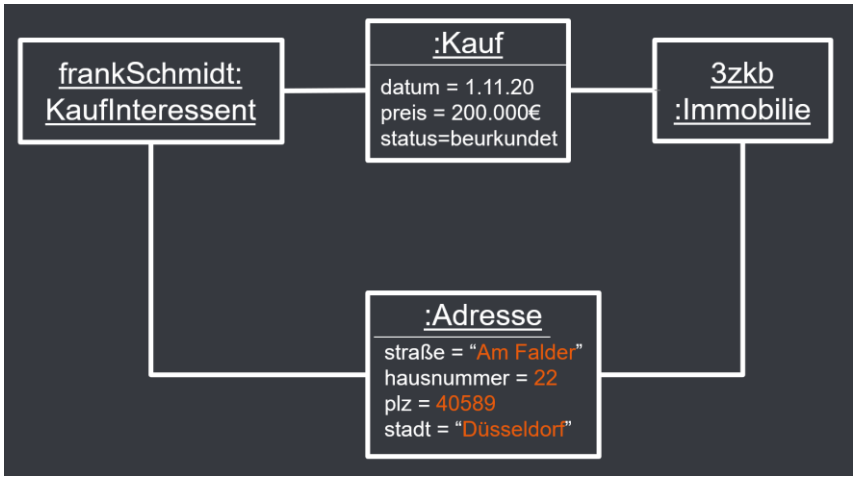
Ist Adresse ein Entity?



- Nehmen wir nun weiter an, dass wir für KaufInteressent und die Immobilie **Adressen** nachhalten wollen.
- Stellen wir dieselbe Frage für Adresse – auch ein Entity?

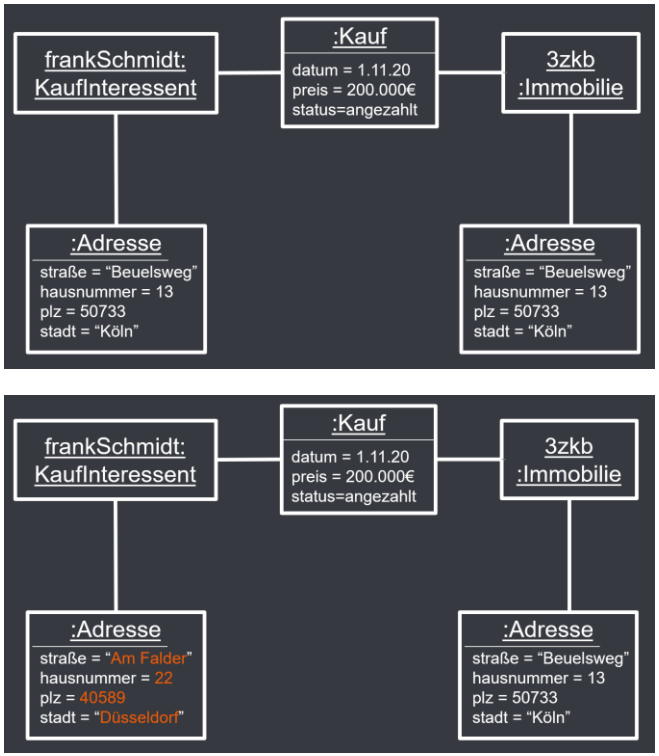
Ist Adresse ein Entity? => Objektdiagramm zur Klärung

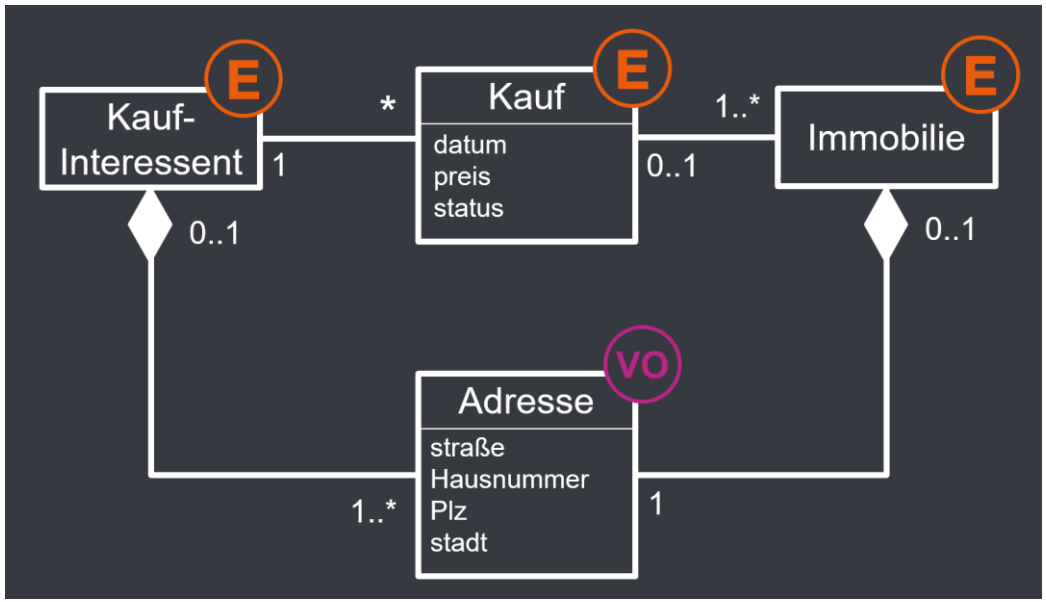




- Zur Klärung schauen wir uns ein *Objektdiagramm* an.
  - Im Objdiagramm werden (im Gegensatz zum Klassendiagramm) keine Klassen, sondern *Objektinstanzen* dargestellt. "*frankSchmidt : KaufInteressent*" ist also ein Objekt namens "frankSchmidt" von der Klasse "KaufInteressent".
- Hier sehen wir die Situation, dass Frank Schmidt in Köln am Beuelsweg 13 wohnt. Er will an derselben Adresse auch eine Immobilie kaufen – nämlich die Wohnung über der, in der er selbst wohnt.
  - Wenn Adresse ein Entity ist, dann müssten wir das so wie hier dargestellt modellieren können – Frank Schmidt verweist mit seiner Wohnadresse auf dasselbe Entity wie die zu kaufende Immobilie.
- Wenn Frank aber der Liebe wegen nach Düsseldorf umzieht (rechts), dann sieht man die Schwächen. Modelliert man so naiv wie oben dargestellt, dann würde sich auch die Adresse seiner zu kaufenden Immobilie ändern.

Besser: Adresse ohne Sharing





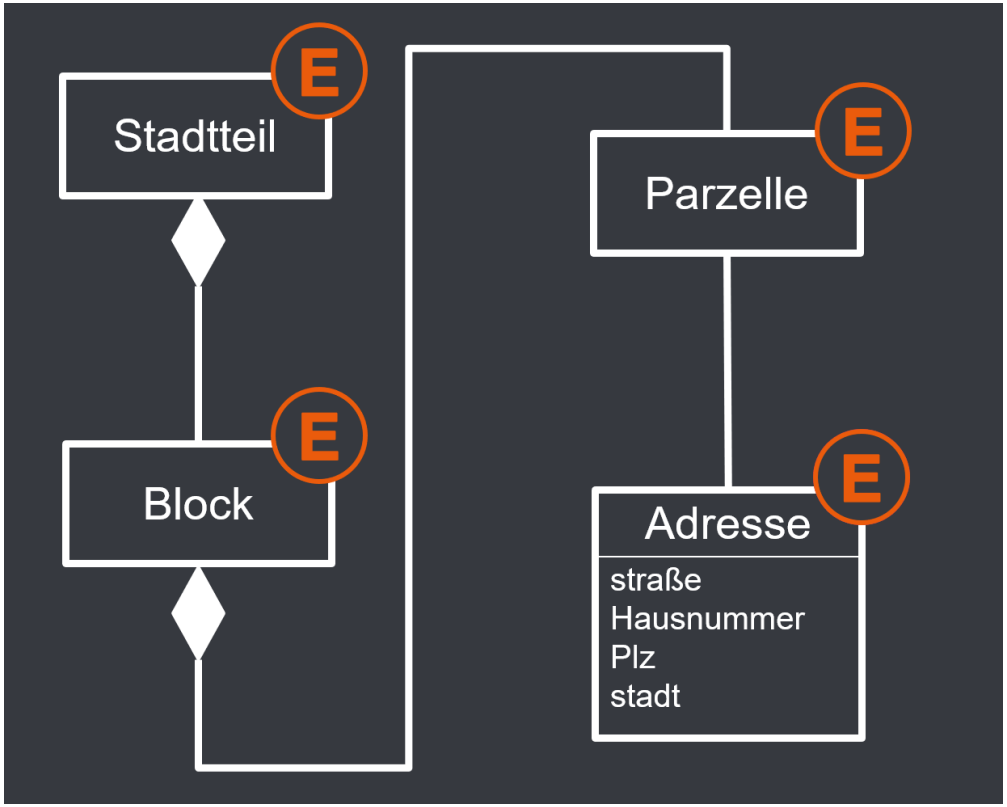
- Besser wäre es, die Modellierung so zu machen: Frank und die Immobilie haben von vornherein eigene Adressobjekte.
  - Dann mache ich keine Änderung an einem gemeinsam genutzten Adressobjekt. Vielmehr kann ich Franks alte Adresse "Beuelsweg in Köln" (links oben) wegwerfen und durch die neue Adresse "Am Falder in Düsseldorf" ersetzen (links unten).
- Damit wird Adresse zum sogenannten **Value Object** (rechts).

Value Objects

- Identität bestimmt durch Attributwerte
  - Andere Attributwerte => anderes Objekt
- Value Objects sind immutable
  - Attributwerte werden nur einmal gesetzt, und danach nicht mehr verändert.
  - Statt Änderung wird das Objekt gelöscht und neu erzeugt.
  - Damit vermeidet man Konfliktsituationen, die z.B. durch ein Sharing von Objektinstanzen entstehen, wie eben gesehen.

- Wie erkennt man Value Objekts? Vier Indizien:
  - ändert man nur einen Attributwert, wird es zu einem anderen Objekt
  - wird nicht mehr verändert, sondern eher gelöscht und neu angelegt
  - kein Sharing
  - stellt ein „komplexes Attribut“ eines anderen Geschäftsobjekts dar

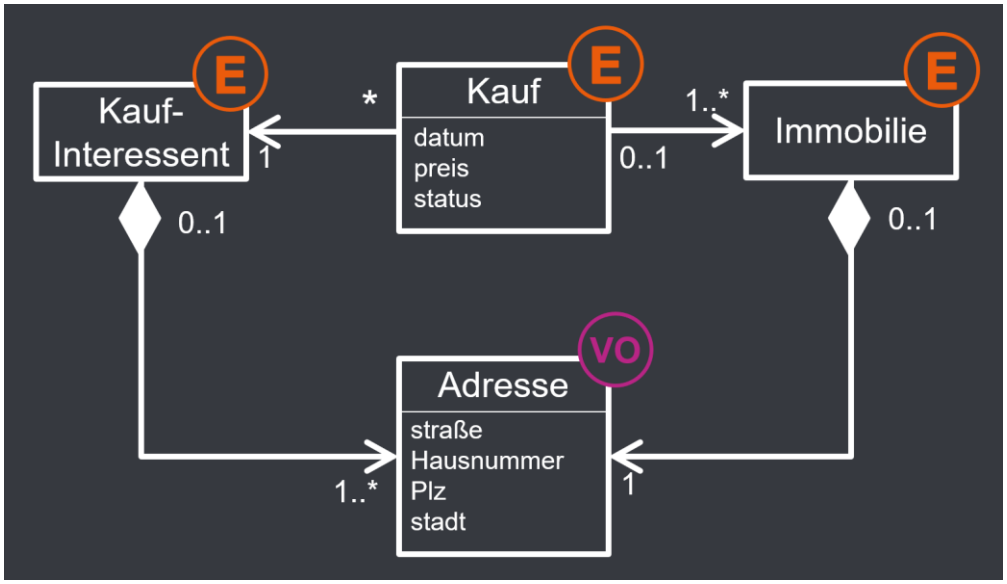
“Value Object oder Entity” ist kontextabhängig!



- In dem Maklerportal, im Gegensatz dazu, sind Änderungen an der Adresse **nicht im Fokus**. Man braucht die Daten, sie sind aber (anders als bei der Stadtplanung) nicht im Fokus des Softwaresystems.
- Genauso wie etwa eine Kontoverbindung ist sie ein komplexes Attribut, das man neu anlegt, wenn es sich ändert.

- Ob man eine Klasse als Entity oder Value Object modelliert, liegt nicht “in der Natur der Klasse”, sondern hängt vom Fokus und Kontext ab.
- Machen wir ein Gedankenexperiment als Beleg: nehmen wir an, wir würden kein Maklerportal, sondern eine **Stadtplanungs-Software** implementieren.
- Dann sähe unser Domain Modell vielleicht so aus links: Es gibt *Stadtteile*, die aus *Wohnblöcken* bestehen. Diese wiederum bestehen aus *Parzellen*. Eine Parzelle hat dann eine Adresse.
  - (Wahrscheinlich würde man die Adresse in dem Modell anders als links dargestellt modellieren, aber ich wollte die Adress-Klasse wiedererkennbar halten.)
- In diesem Beispiel würde man möglicherweise in einem zu planenen Neubaugebiet eine Straße noch einmal umbenennen. Oder man schneidet Parzellen neu zu und die Hausnummer ändert sich, weil es jetzt weniger oder mehr Parzellen in der Straße gibt.
- Hier wäre Adresse damit ein **Entity**!

Und am Schluss ☺ - wofür braucht man das?



- Entities und Value Objects sind wichtige Konzepte aus dem **Domain-Driven Design (DDD)**, das uns noch länger beschäftigen wird.
- Sie sind für uns aus zwei Gründen wichtig:
  1. Beim Übergang von *ungerichteten* zu *unidirektionalen* Beziehungen haben wir hier Ausnahme von der sonst geltenden Regel “speziell => allgemein”. Es gilt immer die **Richtung “Entity => Value Object”**.
     - Da VOs keine eigene Identität haben, sondern sich nur über ihre Attributwerte definieren, können wir sie nicht per ID aus einer Datenhaltung holen. Dadurch wäre die Gegenrichtung nicht sinnvoll.
     - Außerdem haben wir VOs als eine Art von “komplexem Attribut” eingestuft. Attribute gehören eng zu einem Geschäftsobjekt, so dass ich vom GO direkt dorthin navigieren können will.
  2. Sobald wir zur Programmierung mit Spring Data JPA übergehen, sieht man, dass Entities und VOs unterschiedlich behandelt werden. Als Vorgriff: Entities haben eine ID und ein Repository, VOs nicht.