

# Java

Prof. Dr. Stefan Enderle  
NTA Isny

## 6. Klassen (Einführung, UML)



Was der Kunde erklärte



Was der Projektleiter verstand



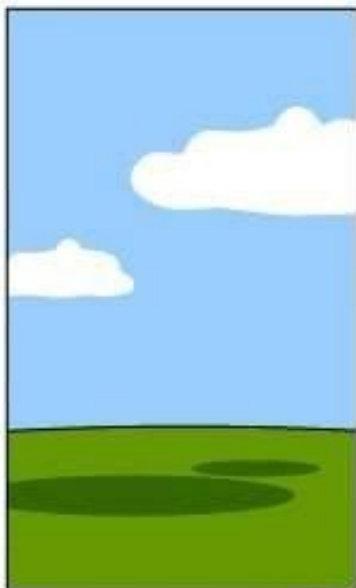
Wie es der Analytiker entwarf



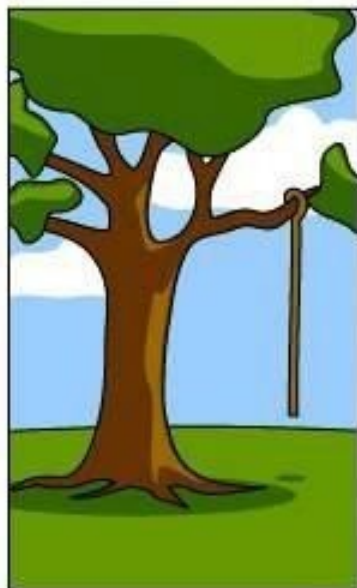
Was der Programmierer programmierte



Was der Berater definierte



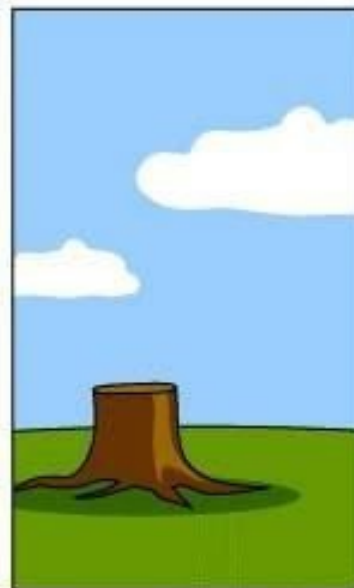
Wie das Projekt dokumentiert wurde



Was installiert wurde



Was dem Kunden in Rechnung gestellt wurde



Wie es gewartet wurde



Was der Kunde wirklich gebraucht hätte

# Objekte

- In einem Software-System können, je nach Anwendung, viele "Objekte" vorkommen.
- Beispiele:
  - Beteiligte oder betroffene Personen  
(*Kunde, Verkäufer, Ansprechpartner, Administrator,...*)
  - Zustände eines Prozesses  
(*Transaktion, Buchung, Reparatur, Abflug, Ankunft,...*)
  - Sachgegenstände eines Prozesses  
(*Vertrag, Rechnung, Memo,...*)
  - Alltagsobjekte der Anwendungsdomäne  
(*Auto, Haus,...*)
  - Infrastruktur  
(*Zimmer, Zimmerplan, Firmenhierarchie,...*)

# Objekte

- Ein Objekt enthält
  - Attribute ( = Daten oder Zustand )
  - Methoden ( = Funktionen )
  
- Graphische Darstellung (unabhängig von Programmiersprache) durch UML Klassendiagramm.

# UML Klassenmodell

- **Detailgrade:**
  - Nur Klassenname
  - Klassenname und Attribute
  - Klassenname, Attribute und Methoden

Aufführungssaal
-----------------

Aufführungssaal
-----------------

Ort Bezeichnung Art Anzahl_Plätze Kosten_pro_Tag
--

Aufführungssaal
-----------------

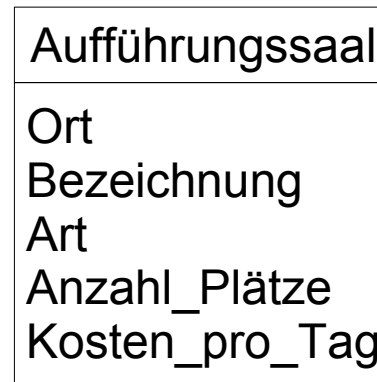
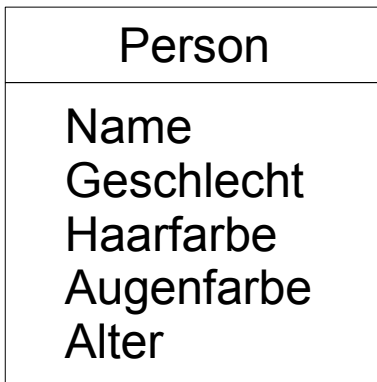
Ort Bezeichnung Art Anzahl_Plätze Kosten_pro_Tag
--

definieren suchen
----------------------

# UML Klassenmodell

- **Attribute:**

- „Gedächtnis“ eines Objektes
- Eigenschaften
- Alle Attribute-Werte zusammen: *Zustand*



- Attribute, die z.B. in einer Datenbank gespeichert werden, um das Objekt später zu rekonstruieren, heißen *persistent*.

# UML Klassenmodell

- **Methoden:**
  - Fähigkeiten einer Klasse
  - Unterscheidungsmöglichkeit:
    - Konstruktoren und Destruktoren
    - Speichern und Laden von Objektzuständen
    - Attribute oder Gesamtzustand ändern
    - Attribute oder Gesamtzustand auslesen
    - Berechnung ausführen, basierend auf aktuellem Zustand

# UML Klassenmodell

- **Methoden**

Person
Name Geschlecht Haarfarbe Augenfarbe Alter
definieren suchen Alter_festlegen Haarfarbe_ändern löschen laden speichern

Aufführungssaal
Ort Bezeichnung Art Anzahl_Plätze Kosten_pro_Tag
definieren suchen

# UML Klassenmodell

- **Vererbung:**

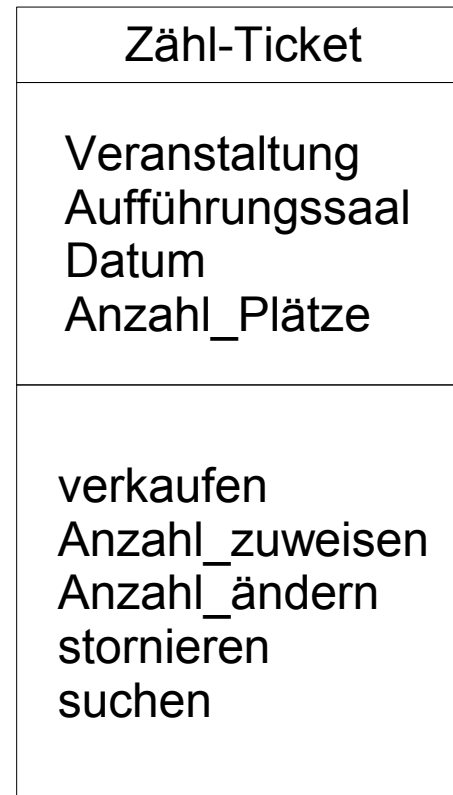
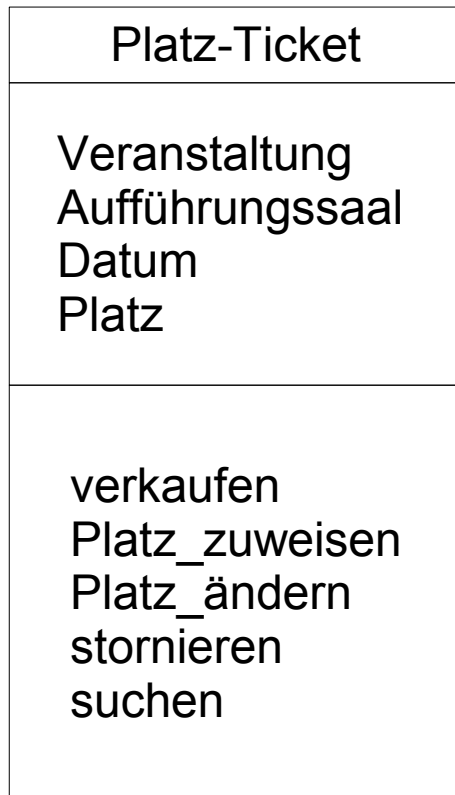
In Klassen kann es inhaltliche Übereinstimmung geben:

- Attribute können gleich sein  
(Typ und Wert)
- Methoden können gleich sein  
(Parameter, Rückgabewert, Code)

- Gibt es genügend Übereinstimmungen, so können diese über die Vererbungsbeziehung verbunden werden:  
*Generalisierung / Spezialisierung (GenSpec)*

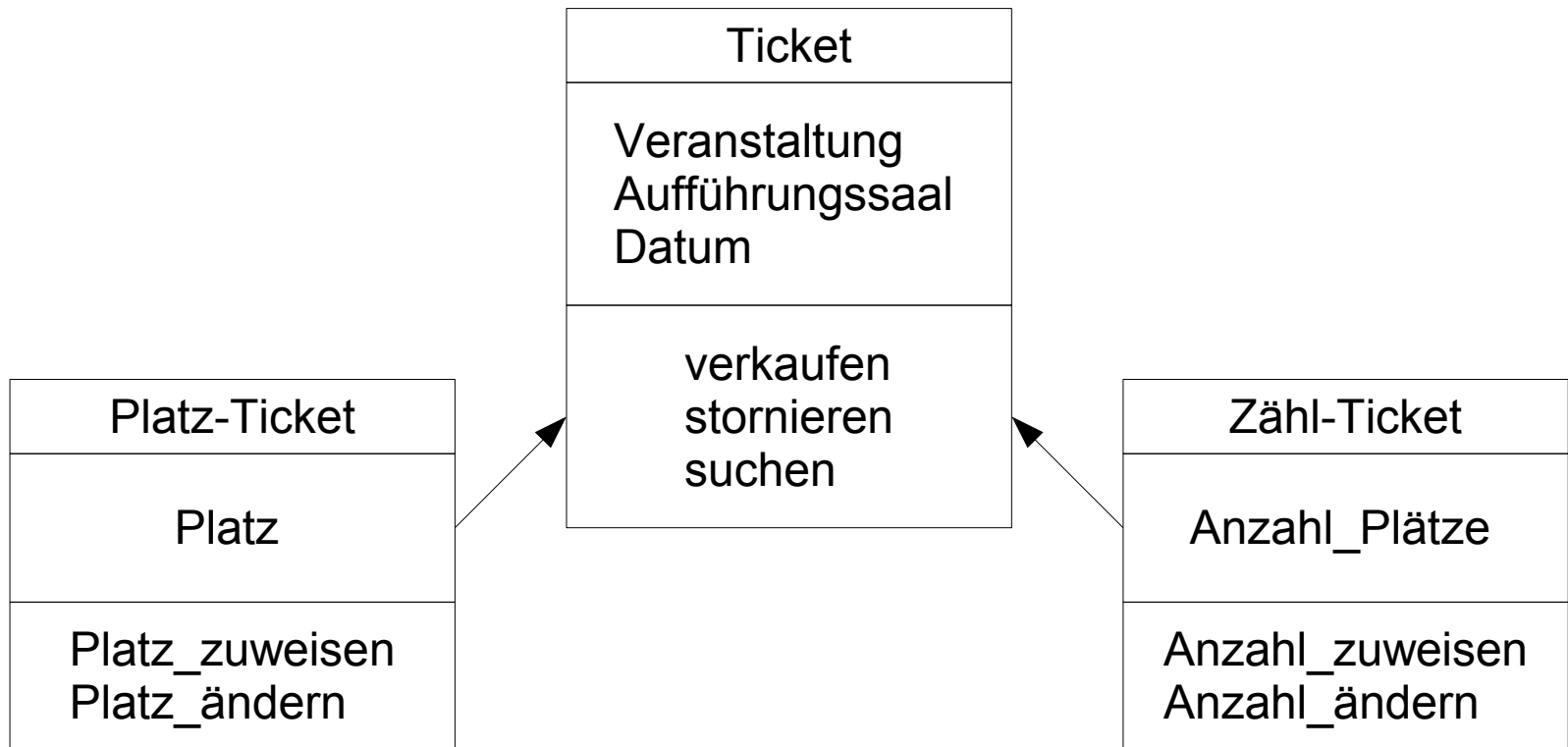
# UML Klassenmodell

- **Vererbung:**  
*Generalisierung / Spezialisierung (GenSpec):*



# UML Klassenmodell

- **Vererbung:**  
*Generalisierung / Spezialisierung (GenSpec):*



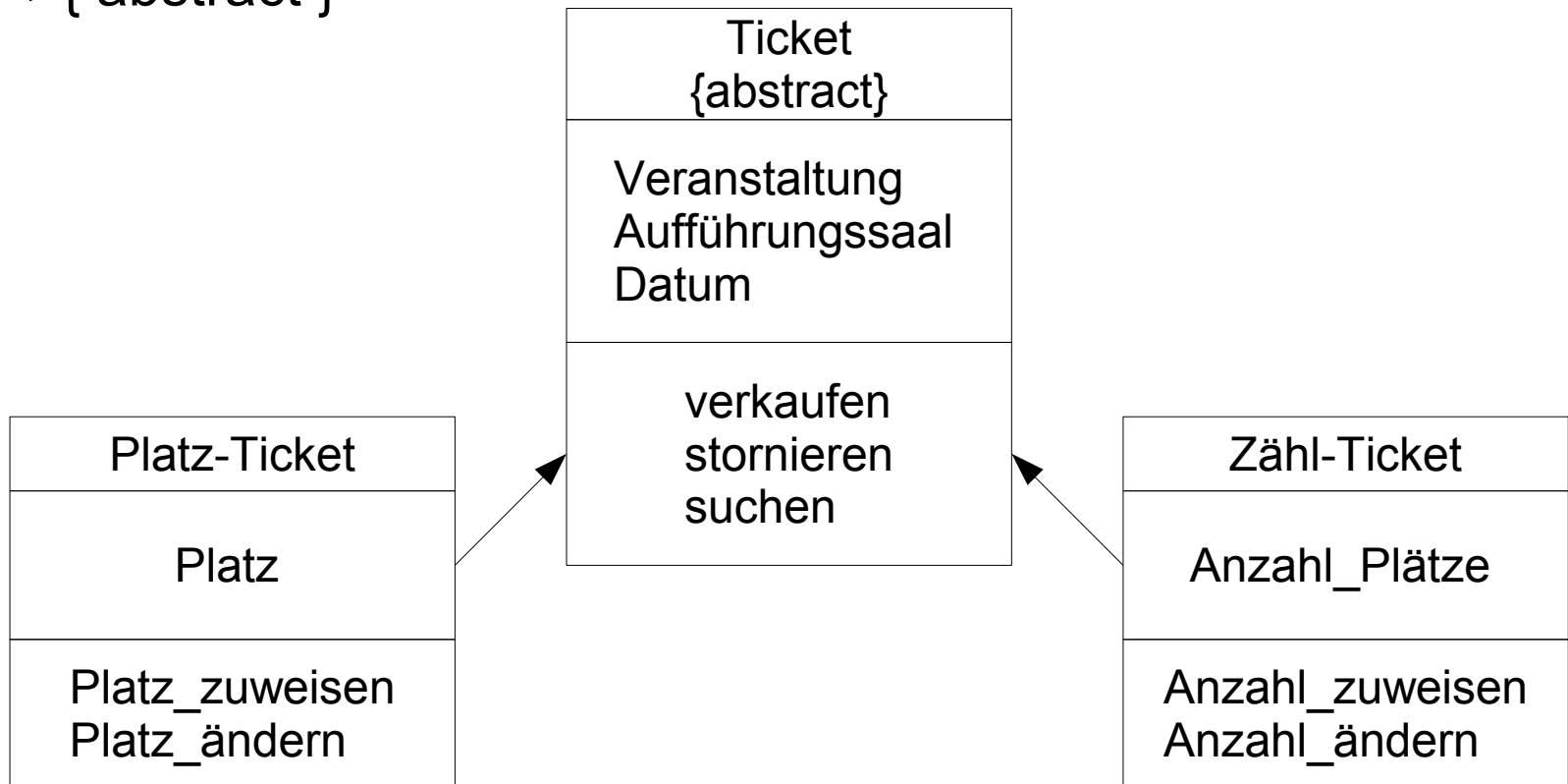
*Bem: Pfeile leer!*

# UML Klassenmodell

- **Abstrakte Oberklasse:**

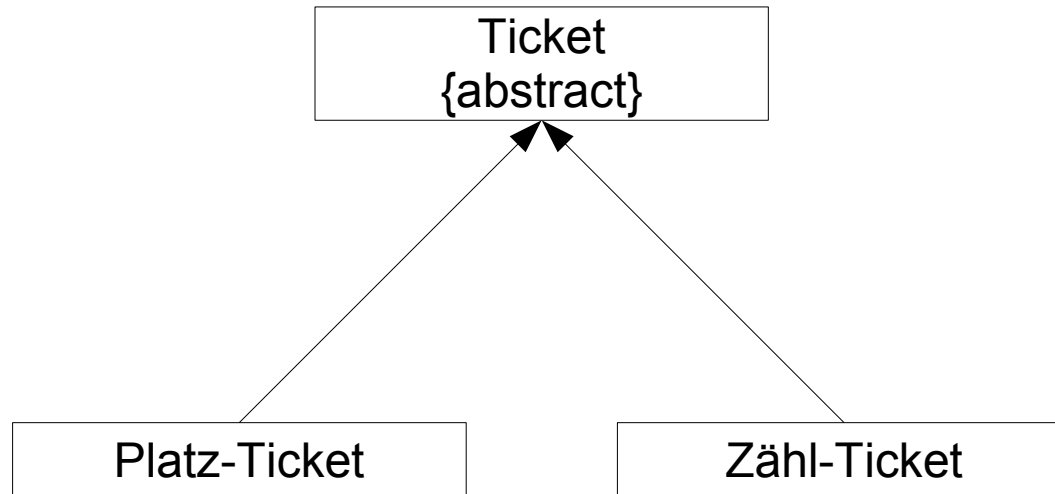
Von der Oberklasse Ticket kann kein Objekt erzeugt werden

→ { abstract }



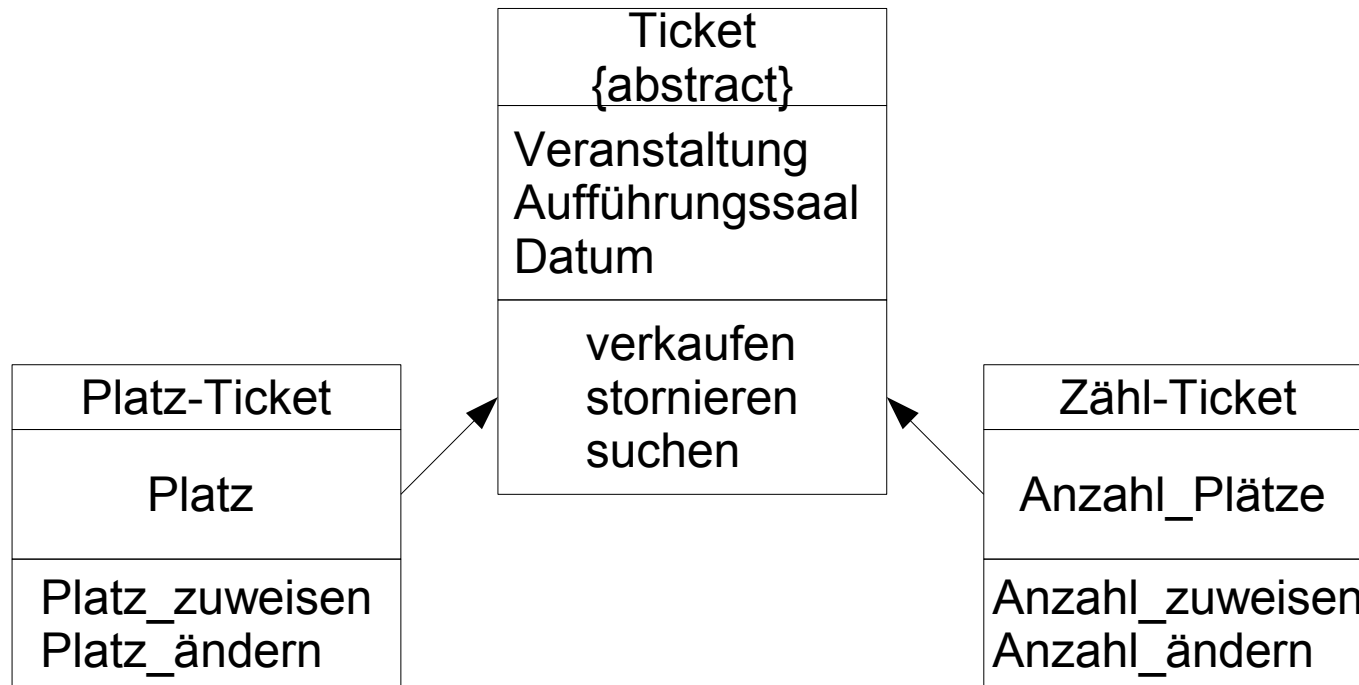
# UML Klassenmodell

- **Vererbungsbeziehung:**  
Wenn es auf Attribute und Methoden nicht ankommt:



# Datentypen, Sichtbarkeit

- **Beispiel:**



- Typen?
- Sichtbarkeit?

# Erweiterungen

- **Sichtbarkeit der Attribute:**

Angaben durch Zeichen *vor* dem Attributnamen:

- „+“ = public: Systemweiter Zugriff
- „-“ = protected: Zugriff nur durch Instanzen der Klasse
- „#“ = protected: Instanzen der Klasse und Unterklassen
- unterstrichen = class-scope (static):  
Systemweiter Zugriff ohne Instanz (Eigenschaft der Klasse)



# Erweiterungen

- **Typen der Attribute:**

Angabe des Typs *nach* „:“ *hinter* dem Attributnamen:

Fenster
<u>Standardgröße: Rechteck</u> +aktuelleGröße: Rechteck -tags: byte #owner: Fenster*

# Erweiterungen

- **Initialisierung der Attribute:**  
Initialisierungswerte werden nach „=“ hinter dem Typ angegeben:

Fenster
<u>Standardgröße: Rechteck = (200,100)</u> +aktuelleGröße: Rechteck -tags: byte = 128 #owner: Fenster* = NULL

# Erweiterungen

- **Sichtbarkeit der Methoden:**

Angabe von +, -, #, \_ wie bei Attributen:

Fenster
<u>Standardgröße: Rechteck = (200,100)</u> +aktuelleGröße: Rechteck -tags: byte = 128 #owner: Fenster* = NULL
-zeigeFenster +öffnen +anzKinder #minimiereAlleKinder

# Erweiterungen

- **Typen der Parameter und Rückgabewerte von Methoden:**  
Wie bei Attributen: *Name* „:“ *Typ*:

Fenster
<u>Standardgröße: Rechteck = (200,100)</u> +aktuelleGröße: Rechteck -tags: byte = 128 #owner: Fenster* = NULL
-zeigeFenster() +öffnen(owner:Fenster*):boolean +anzKinder():int #minimiereAlleKinder()

# Vereinbarungen

- Um das Klassendiagramm nicht zu überladen, wurde vereinbart, dass folgende Kategorien von Methoden weggelassen werden:
  - Konstruktoren: Initialisierung des Objekts
  - Destruktoren: Maßnahmen vor Beendigung des Objekts
  - Set-Methoden: Setzen (einzelner) Attributwerte
  - Get-Methoden: Lesen (einzelner) Attributwerte