03-05-H
-709.53

Informatik für Nichtinformatiker (2)

Prof. Dr. Udo Frese
Tobias Hammer

Blitzeinstieg in Ruby (Fortsetzung)

Objekte und Klassen

Vererbung und Polymorphie

Datenkapselung

Was bisher geschah

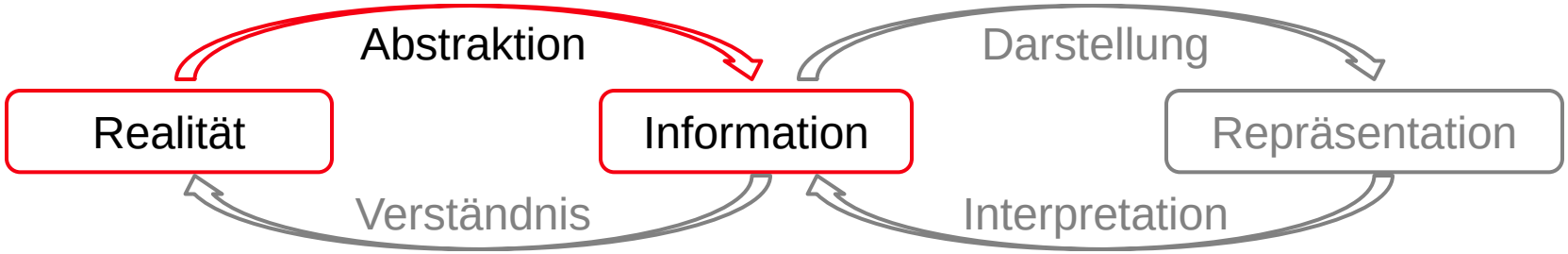
- **Ziel der Veranstaltung**
  - einfache Programme in Ruby erstellen können
  - bei Informatikern mitreden können
- **Was ist Informatik?**
  - Informatik ist die Disziplin der systematischen, automatisierten Verarbeitung von symbolischer Information.
  - Computer sind Maschinen zur Verarbeitung von Symbolen.
  - Programm ist eine formale Vorschrift wie Symbole zu verarbeiten sind
- **Realität, Information, Repräsentation**
- **Blitzeinstieg in Ruby (Programming Ruby, Kapitel 2)**
  - puts, Ausdrücke, object.method, def, Array, Hash, if, while, /…/, Blöcke, File, gets

Blitzeinstieg Ruby

- **(inifrese0901_chapter2Example.rb)**

Objekte und Klassen

Objekte und Klassen



- **Objektorientierter Ansatz zur Softwareentwicklung**
  - Analyse: Was soll warum mit welchen Objekten geschehen?
  - Entwurf: Wie soll es im Prinzip geschehen?
  - Implementierung: Wie soll es im Detail passieren?
- **Wir betrachten im Moment die Analyse:**
  Realität ⇒ symbolische Objekte

Quelle: Küchlin, Weber, Einführung in die Informatik, Springer, 2003, Kapitel 4

Objekte und Klassen

Philosophie

- **Objektorientierte Sichtweise ist, die Realität als Ansammlung von Objekten und deren Interaktion zu sehen.**
- **Objektorientierte Softwareentwicklung bedeutet, von der Realität objektweise zu abstrahieren, d.h. jedes relevante realen Objekt hat eine abstrahierte Entsprechung als symbolische Information.**
- **„Ein Gegenstand [Objekt] ist etwas, auf das wir verweisen können und einen Namen geben können."**
  [H. Seifert, Einführung in die Wissenschaftstheorie", Beck München, 1993]

Quelle: R. Weber, Philosophische Grundlagen der objekt-orientierten Softwareentwicklung,
`www.infnet.verein.de/oophil/index.html`

Objekte und Klassen

Praxis

- **Die Nomen in einer Problembeschreibung sind Kandidaten für Objekte.**
- **Nicht nur physische Dinge, sondern auch Begriffe.**
- **Beispiele:**
  - Kfz-Werkstatt: die Autos, die Kunden, die Mitarbeiter, die Teile, die Werkzeuge
  - Bank: die Kunden, die Mitarbeiter, die Kontos, die Transaktionen
  - Buchhandlung: die Buchexemplare, die Bücher, die Verlage, die Lieferungen
  - Computerspiel: der Spieler, die Gegnerfiguren, die Landschaft, die Gegenstände, die 3D Darstellungen der Figuren, der Joystick, der Spielstand
- **„*die* Autos“, weil jedes einzelne Auto ein einzelnes Objekt ist**
- **Stehen 10 Autos auf einem Parkplatz, gibt es 10 Objekte die ihnen symbolisch entsprechen, eins für jedes Auto.**

Objekte und Klassen

- **Philosophie**
- **Eine Klasse ist eine Menge von Objekten mit gemeinsamen Eigenschaften.**
- **Der griechische Philosoph Plato (427-347 v.Chr) unterscheidet in seiner Ideenlehre Form und Stoff:**
  - Form , Schema, Klasse ist zeitlos und unvergänglich
  - Stoff, Ausprägung ist das konkrete Objekt
- **Plato: Klasse $\Rightarrow$ Objekt**
- **Aristoteles: Objekt $\Rightarrow$ Klasse**

Objekte und Klassen

Informatik ist “platonisch”

- **Klasse ⇒ Objekt**
- **Programmierer definiert die Klasse**
  - Zustand: Daten, die ein Objekt der Klasse ausmachen
  - Funktionalität: Was Objekte der Klasse machen können (und wie)
- **Computer führt Programm der Klasse für jedes Objekt aus**
  - Funktionalität wird realisiert, in dem das Programm der Klasse auf dem Zustand des Objektes arbeitet
- **Klasse wird zur Entwicklungszeit definiert**
- **Objekt wird zur Laufzeit *instantiiert***

Objekte und Klassen

Praxis

- **Der archimedische Abstraktionsschritt von den vielen konkreten Autos (Objekt) zur Klasse Auto ist in der Regel Teil des Alltagswissens.**
- **In der Analyse werden die Klassen definiert. Dies legt fest, welche Typen von Objekten im Programm auftreten können.**
- **Beispiele:**
  - Kfz-Werkstatt: Auto, Kunde, Mitarbeiter, Teile, Werkzeug
  - Bank: Kunde, Mitarbeiter, Konto, Transaktion
  - Buchhandlung: Buchexemplar, Buch, Verlag, Lieferung
  - Computerspiel: Spieler, Gegnerfigur, Landschaft, Gegenstand, 3D Darstellung, Joystick, Spielstand

Objekte und Klassen

Beispiel Verkehrssimulation

- **Wir wollen ein Programm schreiben, das Verkehr simuliert, mit dem man z.B. die Auswirkungen von Ampelschaltung austesten kann**
- **Frage an das Auditorium (Schnellschuss):**
  Welche Klassen kommen vor?

Objekte und Klassen

Hilfsmittel Use-Cases

- **Ein Beispiel, wie ein zu analysierendes oder zu entwerfendes System sich verhält**
- **Konkretes Beispiel für Daten**
- **Frage an das Auditorium: Gebt einen einfachen Use-Case für die Verkehrssimulation**

Objekte und Klassen

CRC-Cards

- **Brainstorming für Objektorientierte Analyse**
- **Jede Karte**
- **Klasse (Class)**
- **Verantworlichkeiten (Responsibilities)**
- **Klassen mit denen er zusammenarbeitet (Collaboration). Gemeint ist Auftrag an ein Objekt der Klasse**

Klasse: Auto	
- sich entlang einer Straße bewegen - fährt zu Ziel - Geschwindigkeit abhängig von Straßentyp bestimmen - einer Route folgen	Straße Route

Objekte und Klassen

CRC-Cards

- **Use-Case as Rollenspiel durchspielen**
- **Entwickler übernimmt die Rolle von einer / mehrerer Klassen**
- **Schritt für Schritt des Use-Cases durchgehen**
- **Entwickler der für eine Klasse zuständig ist**
  - sagt, was die Klasse macht
  - erteilt Kollaborationsklassen Aufträge
  - meldet Fertigstellung
- **Dabei wird überprüft, ob das Konzept stimmig ist.**

Vererbung und Polymorphie

Vererbung und Polymorphie

<u>Frage an das Auditorium</u>: Hier fehlt eine wichtige Struktur! Welche?

PKW | LKW | Bus | Fahrradfahrer | Fußgänger

Vererbung und Polymorphie

<u>Frage an das Auditorium</u>: Hier fehlt eine wichtige Struktur! Welche?

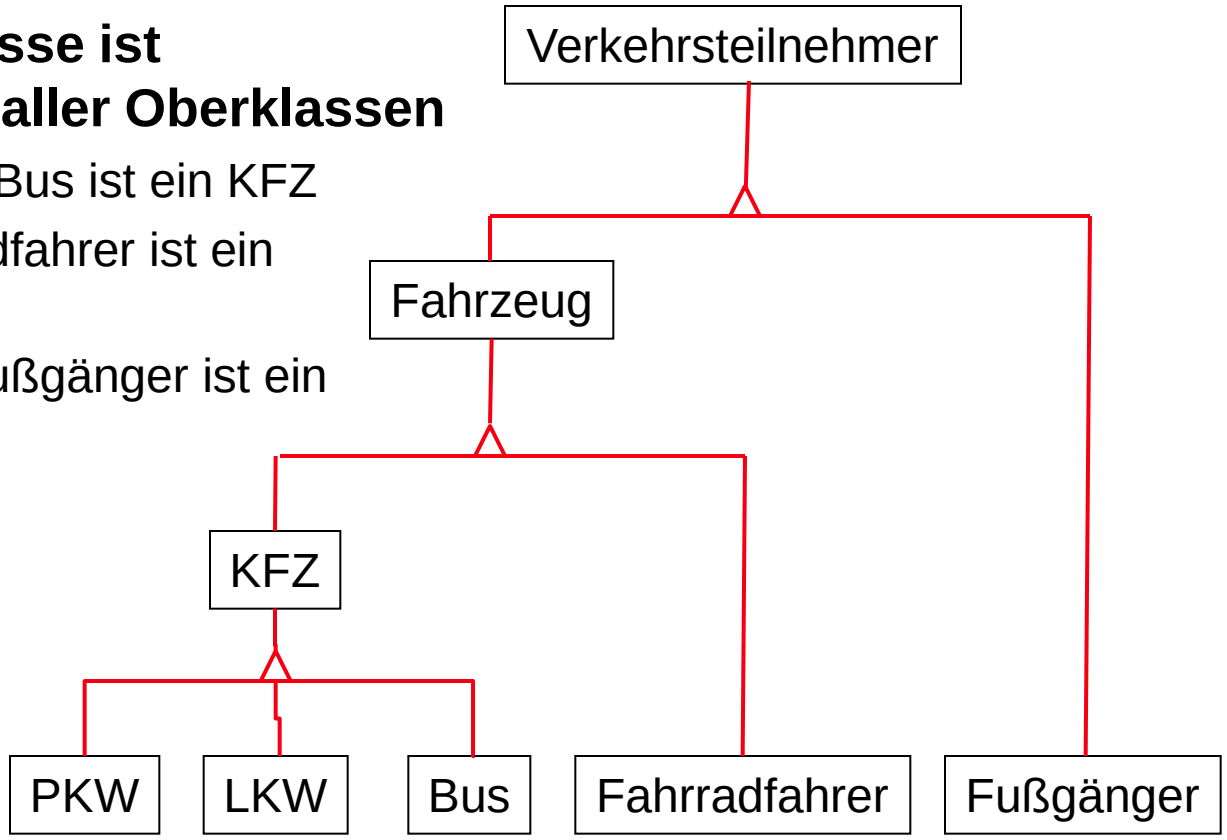
- **Objekte aller Klassen haben Zustand / Funktionalität gemeinsam.**
  - Position, Ziel
  - können sich bewegen
  - können einer Route folgen
- **PKW, LKW und Bus haben gemeinsam**
  - Kennzeichen
  - Führerschein nötig
  - Tankstand
- **PKW, LKW, Bus und Fahrrad haben gemeinsam**
  - Insassen

PKW | LKW | Bus | Fahrradfahrer | Fußgänger

Vererbung und Polymorphie

Hierarchie von abgeleiteten Klassen

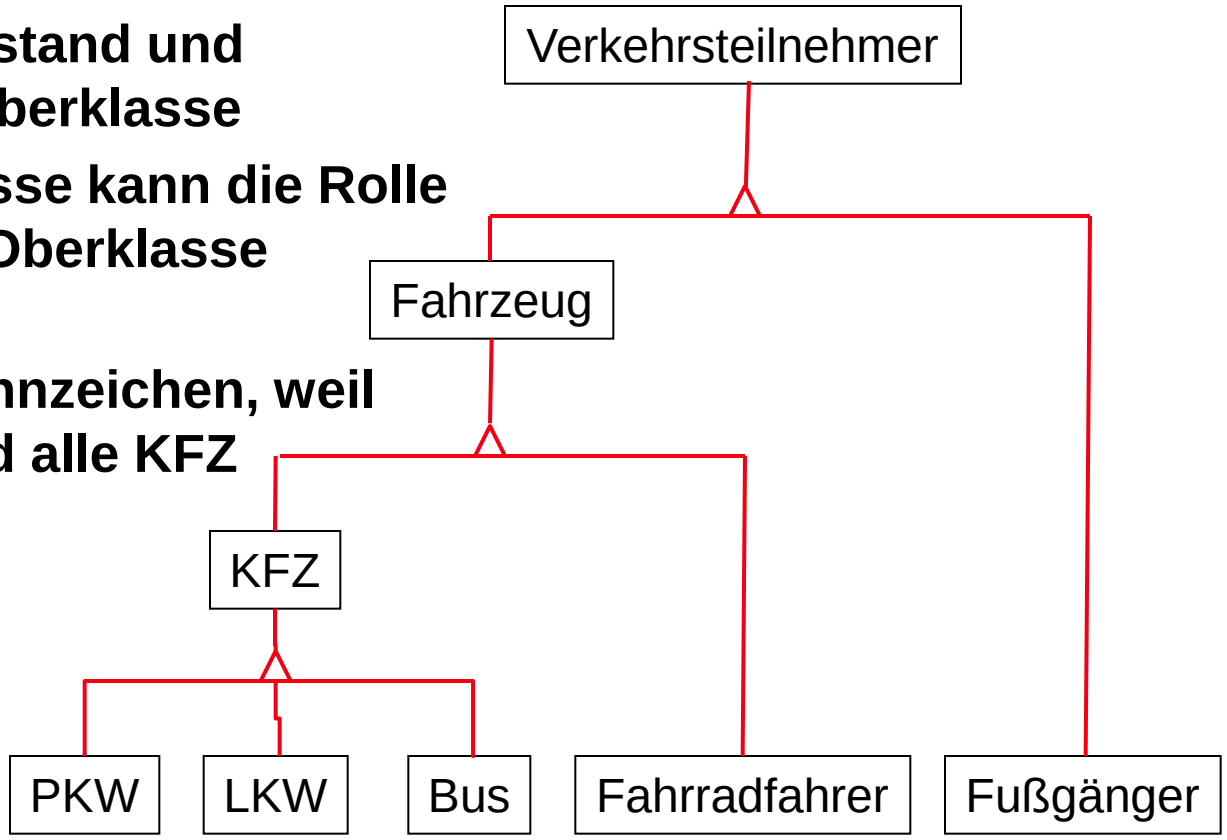
- **Objekt von Unterklasse ist automatisch Objekt aller Oberklassen**
  - ein PKW, LKW oder Bus ist ein KFZ
  - ein KFZ oder Fahrradfahrer ist ein Fahrzeug
  - ein Fahrzeug oder Fußgänger ist ein Verkehrsteilnehmer



Vererbung und Polymorphie

Vererbung

- **Unterklasse erbt Zustand und Funktionalität der Oberklasse**
- **Objekt der Unterklasse kann die Rolle eines Objektes der Oberklasse einnehmen**
- **Ein PKW hat ein Kennzeichen, weil PKW ein KFZ ist und alle KFZ Kennzeichen haben**



Vererbung und Polymorphie

Polymorphie

- **Alle Verkehrsteilnehmer müssen berechnen können, wie schnell sie sich auf einer bestimmten Straße bewegen.**
- **Aber verschiedene Klassen berechnen es anders:**
  - Fußgänger: immer 6km/h
  - PKW: 50 km/h, 100 km/h oder 180 km/h, je nach Straßentyp
  - Fahrrad: je nach Steigung (z.B. durch Formel)
  - LKW: 50 km/h, 80 km/h oder 100 km/h, aber höchstens je nach Steigung
- **auch, wenn LKW als Verkehrsteilnehmer betrachtet wird, muss die Verantwortlichkeit „Geschwindigkeit bestimmen“ nach dem LKW spezifischen Vorgehen ausgeführt werden**

Datenverkapselung

Datenverkapselung

- **Verantwortlichkeiten eines Objektes**
  - der Nutzer eines Objektes weiß *was* passieren soll
  - das Objekt weiß *wie etwas* passiert
- **Vergleichbar: Ich kaufe Brötchen beim Bäcker, der Bäcker backt sie.**
- **Wie es innen funktioniert, geht außen keinen an!**
- **Vorteil**
  - Klassen werden entkoppelt
  - Entwickler werden entkoppelt

Datenverkapselung

Beispiel Klasse Bruch

- **Zustand: Zähler, Nenner**
- **Aber nach aussen soll es einfach wie eine Zahl wirken**
- **Niemand darf von aussen Zähler und Nenner verändern**
- **Statt dessen Operationen, die für einen Bruch Sinn machen**
  - plus, minus, mal, geteilt
- **Die Klasse Bruch weiß, wie zwei Brüche addiert, subtrahiert, multipliziert und dividiert werden.**

Zusammenfassung

- **Objektorientierte Sichtweise ist, die Realität als Ansammlung von Objekten und deren Interaktion zu sehen.**
- **Objektorientierte Softwareentwicklung bedeutet, von der Realität objektweise zu abstrahieren, d.h. jedes relevante realen Objekt hat eine abstrahierte Entsprechung als symbolische Information.**
- **Eine Klasse ist ein Typ von Objekten, die im Programm gleich behandelt werden**
  - Zustand: Welche Daten beschreiben ein Objekt?
  - Funktionalität: Was kann das Objekt tun?
- **Vererbung: Objekte einer Unterklasse sind auch Objekte der Oberklasse, Zustand und Funktionalität wird übernommen**
- **Polymorphie: Es wird immer die Funktionalität genommen, die in der wirklichen Klasse eines Objektes definiert ist, auch wenn es die Funktionalität in einer Oberklasse schon gibt.**
- **Datenverkapselung: Das Objekt ist für das Wie verantwortlich.**