Universität Stuttgart

PSE:
Einfache Programmstrukturen

Vorlesung 3

Beitragsfreie GI-Mitgliedschaft

- Für Studierende im Bachelor- und Masterstudium sowie Auszubildende
- (Für Studierende im Dual- oder Promotionsstudium gilt weiterhin die ermäßigte Mitgliedschaft von 21,50 Euro.)
- Infos zur Gesellschaft für Informatik e. V. (GI)
  - http://www.gi.de
  - Oder bei mir

Bitte zurücksenden an:

Gesellschaft für Informatik e. V. (GI) – Wissenschaftszentrum -
Ahrstraße 45, 53175 Bonn (DEUTSCHLAND
Telefon: + 49 (0)228-302-151/-149 Telefax: + 49 (0)228-302-167
E-Mail: mitgliederservice@gi.de / http://www.gi.de

GESELLSCHAFT FÜR INFORMATIK

AUFNAHMEANTRAG

BEITRAGSFREIE MITGLIEDSCHAFT (EINJÄHRIG)

Ich beantrage die beitragsfreie Mitgliedschaft als ordentliches Mitglied in der Gesellschaft für Informatik e. V. (GI) zum:

1. JANUAR _____ **1. APRIL** _____ **1. JULI** _____

Anträge können bis **15.09.** rückwirkend für das laufende Jahr bearbeitet werden, danach erfolgt die Aufnahme zum 1. Januar des Folgejahres.

Fachgruppenmitgliedschaften/Fachzeitschriften sind nicht in der beitragsfreien Mitgliedschaft enthalten und können kostenpflichtig bestellt werden.

Bitte die nachfolgenden Felder ausfüllen (in Blockbuchstaben) und Zutreffendes ankreuzen. Die mit (*) gekennzeichneten Felder sind Pflichtfelder (Angaben erforderlich).

☐ Frau* ☐ Herr*

Name*	Vorname*
Geburtsjahr*	
Firma / Institution (optional)	
Straße* / Postfach	PLZ* / Ort*
Mobil- / Telefon-Nummer	E-Mail-Adresse*
Regionalgruppe* (siehe https://gi.de/netzwerk)	

Ich bin (**Angabe erforderlich**):

☐ Schüler(in) ☐ Preisträger(in) ☐ BWINF-Teilnehmer(in) ☐ Tagungsteilnehmer(in) / Sonstiges

Die Mitgliedschaft wurde vergeben von / aufgrund von: _____

_____

In dieser Mitgliedschaft ist das Informatik Spektrum Digital enthalten, alternativ bieten wir die Zeitschrift LOGIN (für Lehrberufe) an.

☐ Ich wünsche die Zeitschrift LOGIN.

© 2018 - Gesellschaft für Informatik e.V. (GI) Stand September 2018 Seite 1von 2

Universität Stuttgart

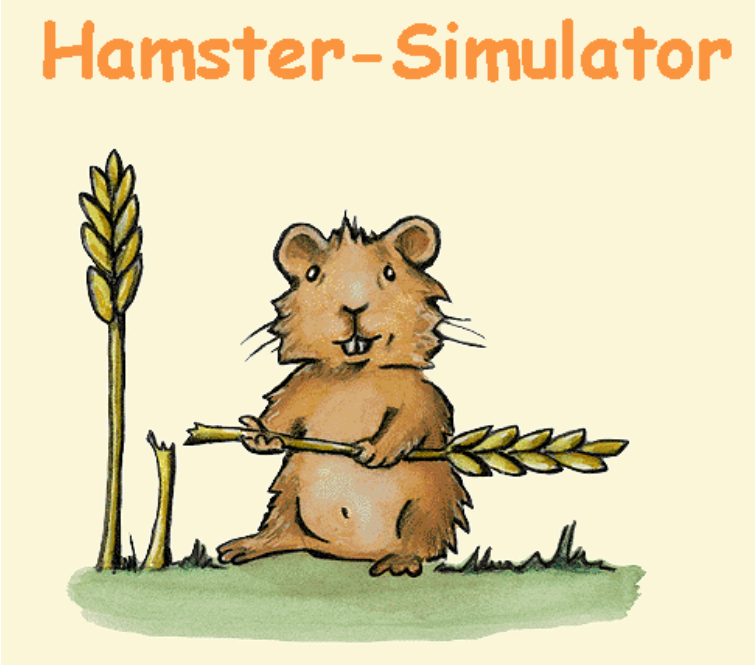
Recap

PSE:
Mit Objekten arbeiten

Prof. Dr.-Ing. Steffen Becker

Vorlesung 1

Welcome Paule!



Herzlich Willkommen, Paule!

Ausfüllen des Operationsrumpfs

```
class Example1 extends SimpleHamsterGame {

   void run() {
      game.displayInNewGameWindow();
      game.initialize();
      paule.move();
      paule.move();
      paule.pickGrain();
      paule.pickGrain();
      paule.turnLeft();
      paule.turnLeft();
   }

}
```

Example01.java

Formatieren des Programms

```
class Example1
extends SimpleHamsterGame {

   void run() {
      game.displayInNewGameWindow();
      game.initialize();
      paule.move();
      paule.move();
      paule.pickGrain();
      paule.pickGrain();
      paule.turnLeft();
      paule.turnLeft();
   }

}
```

Nutzen Sie Einrückungen, um Ihre Programme zu formatieren

White space Elemente füllen den Raum zwischen Ihrem Programmcode:
Spaces, Zeilenumbrüche, Tabs, ... (wobei Tabs den Spaces vorgezogen werden sollten)

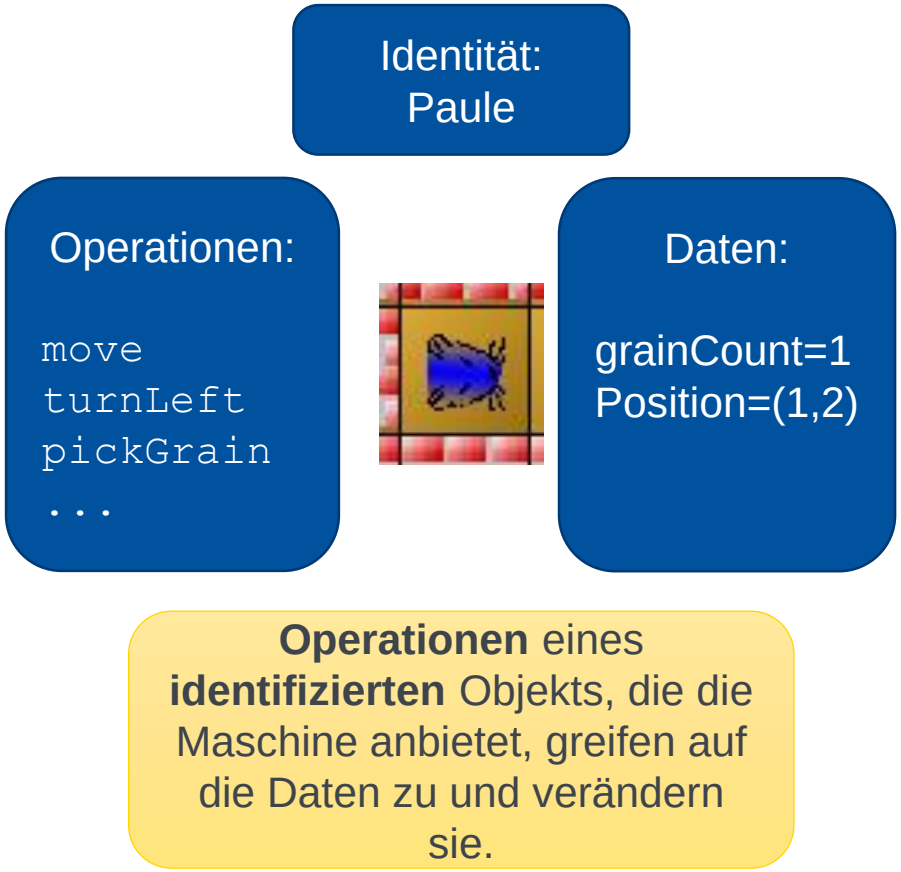
Sie werden alle ignoriert, sprich wirken sich nicht auf die Programmsemantik aus

Typographische Elemente verbessern die Lesbarkeit:
Farben, Kursiv, Fett, etc.

Example01.java

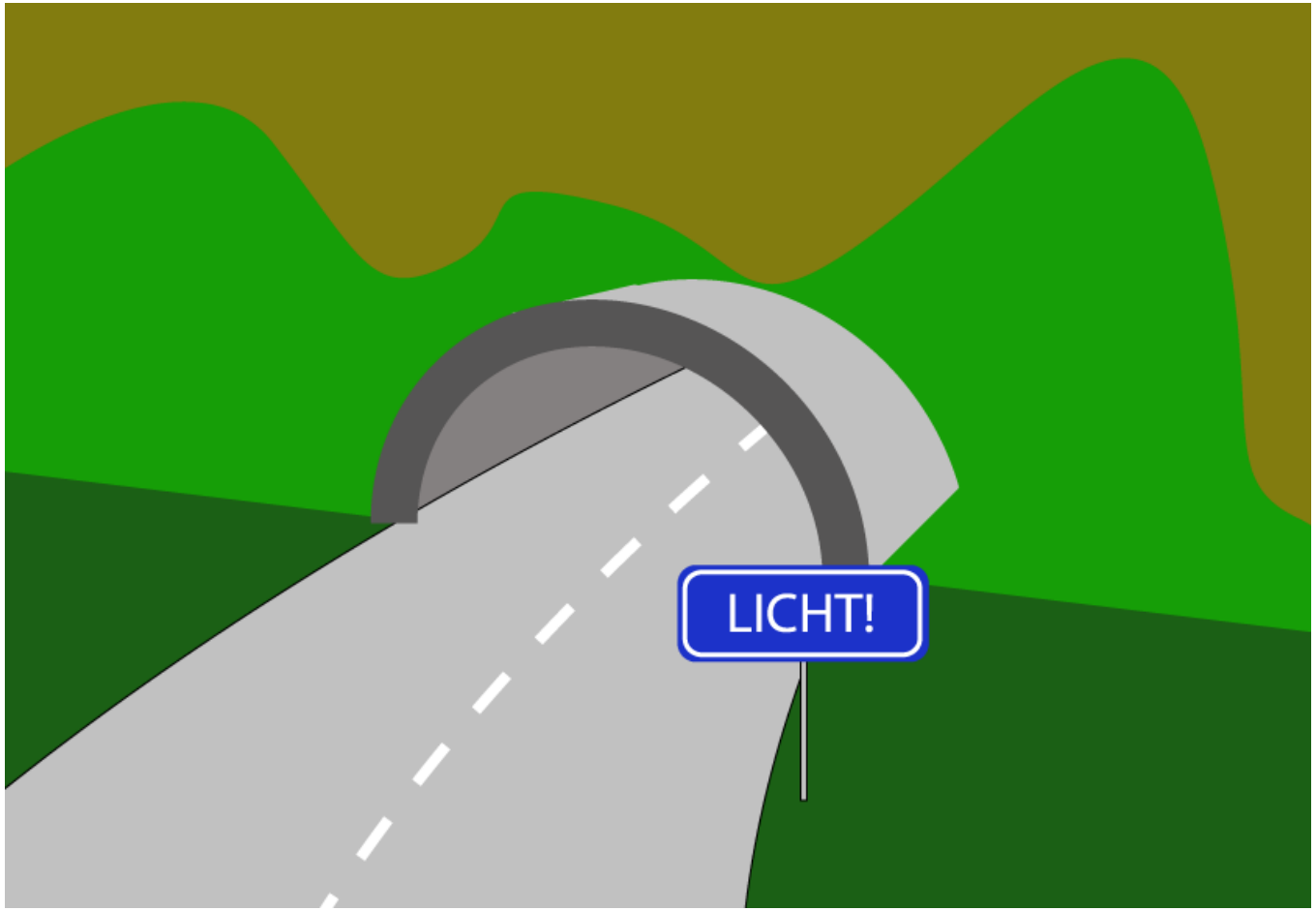
Eigenschaften von Objekten

1. Ein Objekt verwaltet **Daten**, die im Speicher liegen
2. Ein Objekt ist eine Maschine, die **Operationen** anbietet
3. Ein Objekt hat eine **Identität**

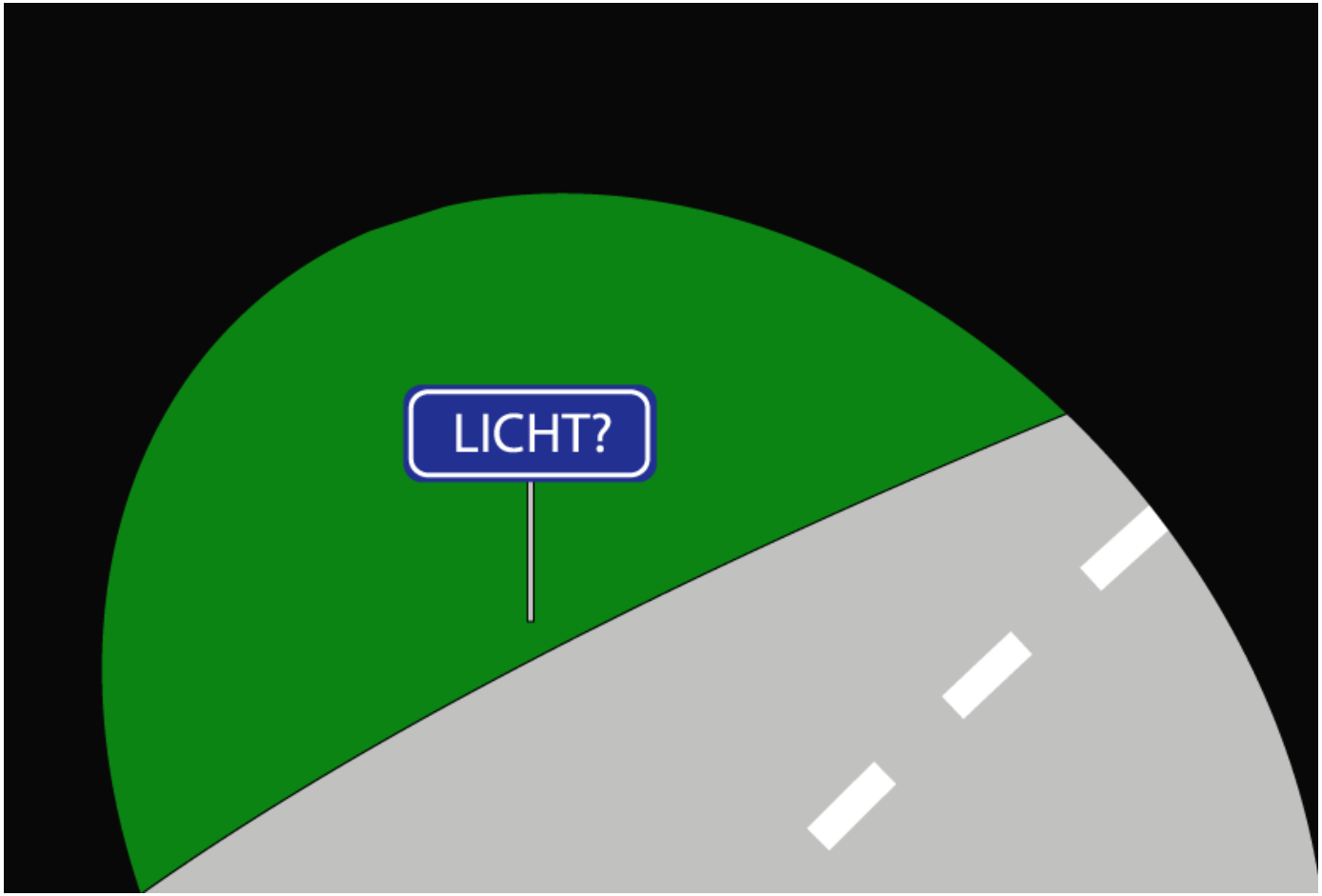


Operationen eines **identifizierten** Objekts, die die Maschine anbietet, greifen auf die Daten zu und verändern sie.

Ein Kommando

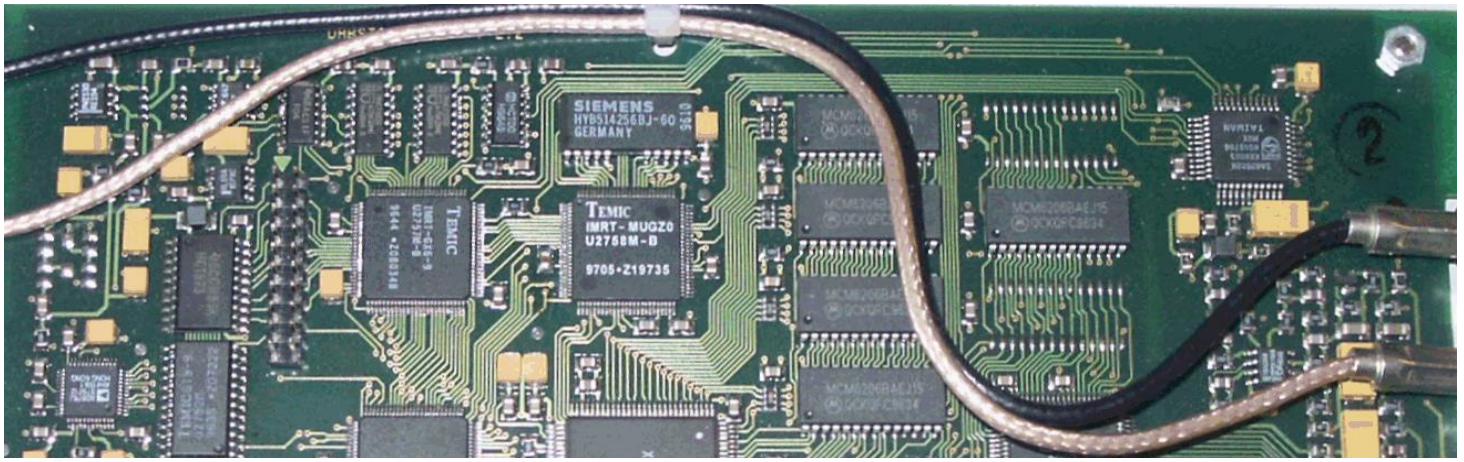


Eine Abfrage

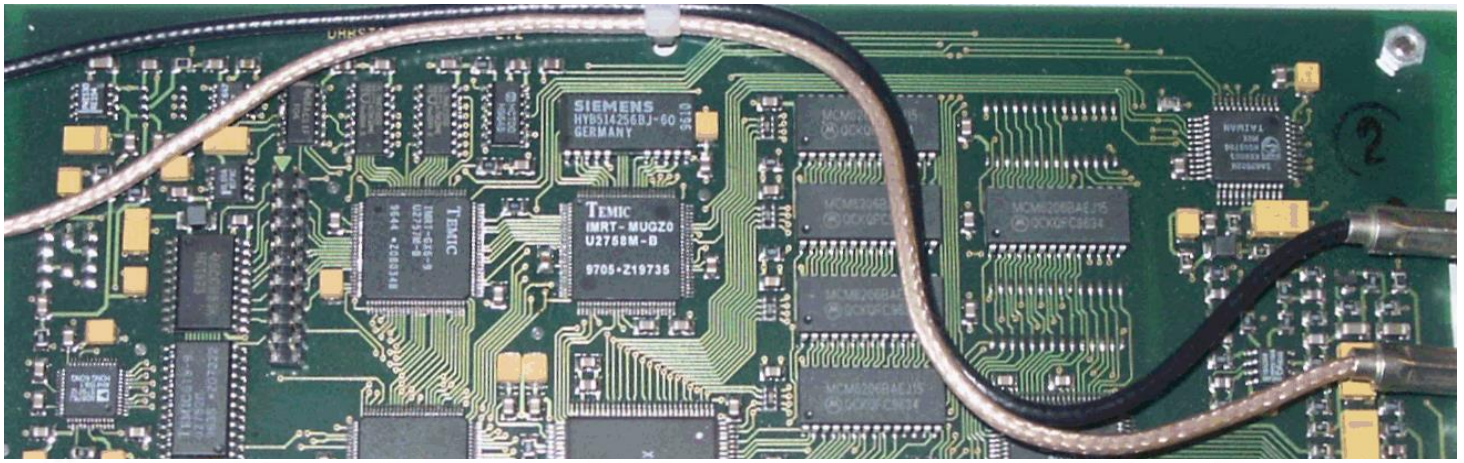


Ein Objekt hat eine Schnittstelle

Ein Objekt hat eine *Implementierung (Realisierung)*



Verbergen der Implementierungsdetails (Inf. Hiding)

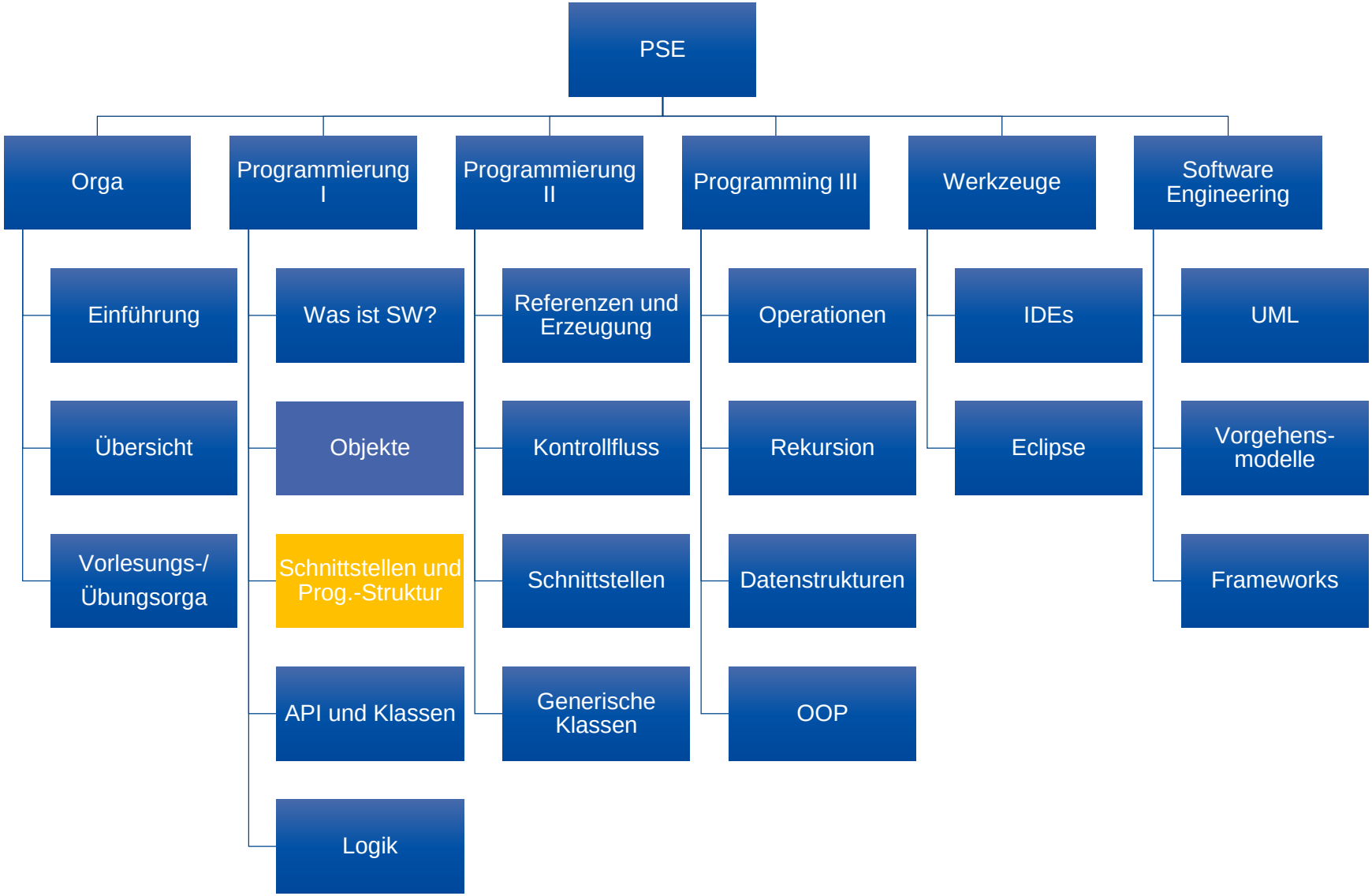


Universität Stuttgart

PSE:
Einfache Programmstrukturen

Vorlesung 3

Vorlesungsübersicht



Lernziele dieser VL

Verschiede Arten von Sprachen

Eigenschaften von Sprachen

- Syntax, Semantik, Pragmatik

Programmstruktur

Syntaxbäume benutzen

Arten von Fehlern, die der Compiler ausgibt

- u.A. syntaktische und semantische Fehler

Hauptquelle: Touch of Class, B. Meyer, Kapitel 3

Didaktik dieser VL

Vermittlung von Fakten und historischen Zusammenhängen

Vom Beispiel zu allgemeinen Konzepten → Induktives Vorgehen

Aktivierung der Studierenden:

- Anwendung der Konzepte (Syntax, Semantik, Pragmatik)
- Gemeinsames Entwickeln einer Lösung (AST)

Motivation für die VL

“Fehlerfreie“ Programme schreiben ist oft recht schwer. Wieso?

- Fehler werden von Werkzeugen erkannt und gemeldet. Als Entwickler muss man diese verstehen und beheben
- Fehler gibt es aus verschiedenen Arten von Gründen. Oft wird nicht der richtige Grund genannt. Als Entwickler muss man die Fehler richtig einordnen, um sie zu beheben
- Als Entwickler muss man einige Fehler auch ohne Werkzeuge erkennen können – weil Werkzeuge es nicht können. Dazu muss man die Fehlerklassen und ihre Erkennung erlernen

→ Sprachtheorie und Hintergrundwissen nötig, um dies zu können

Beispiel: Welcher Code ist fehlerfrei?

```
class 666Devil
extends SimpleHamsterGame {
   void run() {
      paule.move();
   }
}
```

A

```
class Example2
extends SimpleHamsterGame {
   void run() {
      paule.mvoe();
   }
}
```

B

```
class Example9
extends SimpleHamsterGame {
   void run() {
      paule.move();
   }
}
```

C

Welcher Code (A-C) ist fehlerfrei?
Welche Fehler haben die anderen?
Woher wissen Sie, dass die anderen Fehler haben?

Hintergrundinformationen: Programmiersprachen und die Motivation für Java

Programmiersprachen

Programmiersprachen sind die Notationen, die **Syntax** und **Semantik** von Programmen definiert.

Wie ein korrektes Programm hinschreiben?

Was bedeutet und macht das Programm?

Unsere Programmiersprache in PSE ist **Java, Version 11**

Es gibt viele Programmiersprachen, einige „allgemein“ andere “spezialisiert“

Programmiersprache sind **künstliche Notationen**, die für einen bestimmten Zweck entworfen wurden (Programmierung).

Objekttechnologien

Ursprung: Simula 67, Oslo, Mitte der Sechziger, langsam verbreitet in den Siebzigern

Smalltalk (Xerox PARC, 1970er) hat OO hip gemacht, indem es OO mit visuellen Techniken verbunden hat

Die erste **OOPSLA** 1986 hat OO massentauglich gemacht, wonach es in den 90ern auf breiter Front angenommen wurde

- OO Sprachen: Objective C, C++, Eiffel, Java, C#...
- OO Werkzeuge, OO Datenbanken, OO Analyse...

Heutzutage weitestgehend akzeptierte Technologie. Sprachen, die nicht OO sind bezeichnet man als “**prozedural**”.

Über Java

- Entwickelt als **Oak** im „The Green Project“ von 1991-1992 für Sun Microsystems
- Ziel: Betriebsumgebung inkl. virtueller CPU
- 1995 Alpha Release von Java 1.0a2 für Internet Anwendungen
  - WebRunner/HotJava für den **Mosaic** Webbrowser
- 1997 JDK 1.1.4 inkl. Integration in den **Netscape Navigator**
  - Java wurde Mainstream
- 2000er: Java wurde zur bestimmenden Sprache auf Servern
- 2010 Übernahme von Sun durch **Oracle**, Weiterentwicklung auf Basis des Java Community Prozesses

Einige Erfolgsfaktoren von Java

Breite Werkzeug–unterstützung

Bibliotheken

Laufzeit–umgebungen

IDEs

Laufzeit–generierung

Plattform–unabhängigkeit

Einige Beispielprojekte

IDEs

- Eclipse
- Netbeans

Server

- Java EE Server (JBoss, Netsphere, ...)
- Spring / Spring Boot

Datenbanken

- Cassandra
- MongoDB

Wieso also Java in PSE?

Vorteile

- Mainstream Sprache, seit einigen Jahren regelmäßig Top 1 unter den meistgenutzten Sprachen
- Hervorragende Entwicklungsumgebungen und Werkzeuge
- Portabilität (Linux, Mac, Win)
- Kosten (Die meisten Werkzeuge und Bibliotheken sind kostenlos)
- Konzepte → Viele moderne OO-Konzepte gibt es in Java

Nachteile

- Java ist nicht „reine Lehre“, Kompromisse existieren
- Einige Konzepte sind nicht oder nur eingeschränkt in Java
- Wir versuchen, das durch Drittwerkzeuge zu kompensieren

Konzepte von Sprachen: Beispiele anhand unserer Hamsterminisprache

Instruktionen

```
class Example1 extends SimpleHamsterGame {

   void run() {
      game.displayInNewGameWindow();
      game.initialize();
      paule.move();
      paule.move();
      paule.pickGrain();
      paule.pickGrain();
      paule.turnLeft();
      paule.turnLeft();
   }

}
```

Elementare Befehle → Instruktionen

8 Instruktionen hier...

Instruktionssequenzen: eine nach der anderen, getrennt durch ;

Stilregel: Eine Instruktion pro Zeile

Example01.java

Ausdrücke

Ein **Ausdruck** ist ein Programmelement, welches für eine Menge möglicher Laufzeitwerte steht

Beispiele:

```
telephone.ring_several(2 * 5, Loud)
```

Unter anderem auch normale, mathematische Ausdrücke:

2 * 5, a + b, ...

Im Quellcode:

Eine **Instruktion** steht für eine elementare Operation, die während der Programmausführung ausgeführt wird.

Ein **Ausdruck** steht für einen Wert, der von einer Instruktion zur Ausführung benutzt wird.

Etwas Sprachtheorie: Syntax, Semantik, Pragmatik

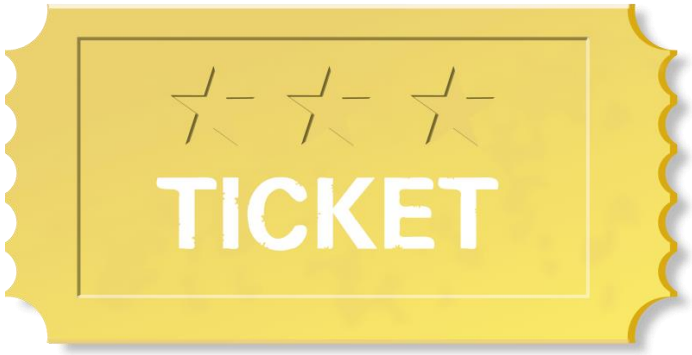
Sprache

- Im Allgemeinen, viele versch. Bedeutungen des Worts „Sprache“
- Im Allgemeinen ☞ Das „gesprochene“ Wort
- In der Informatik ☞ Jede Konvention für Kommunikation
- Kommunikation Mensch « Maschine
- Mensch ⟶ Mensch, Mensch ⟶ Maschine,
  Maschine ⟶ Mensch, Maschine ⟶ Maschine
- Kommunikation in geschriebener Form, in Bildern oder gesprochenen Wörtern

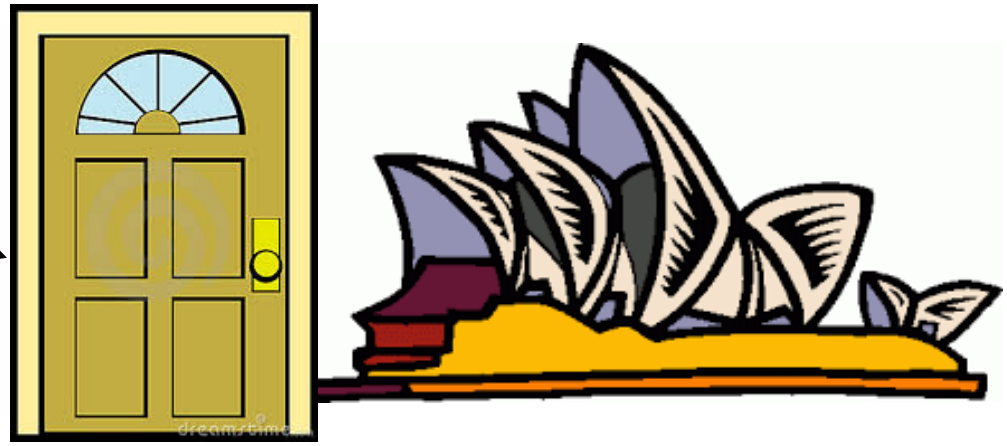
Syntax, Semantik, Pragmatik: Einführung

- **Syntax** ≈ Gültige Strukturen von Sätzen
  - Die Art und Weise, Programme aufzuschreiben
  - Gültigkeit wird vom **Compiler** geprüft
- **Semantik** ≈ Bedeutung gültiger Sätze
  - Erwarteter Effekt eines IT Programms in seiner Umgebung
  - Der generierte Maschinencode repräsentiert diese Bedeutung
- **Pragmatik** ≈ Auswirkung gültiger Sätze
  - Auswirkung des Programms auf die reale Welt
  - Oft nicht Teil der Informatik, sondern der Anwendungsdomäne

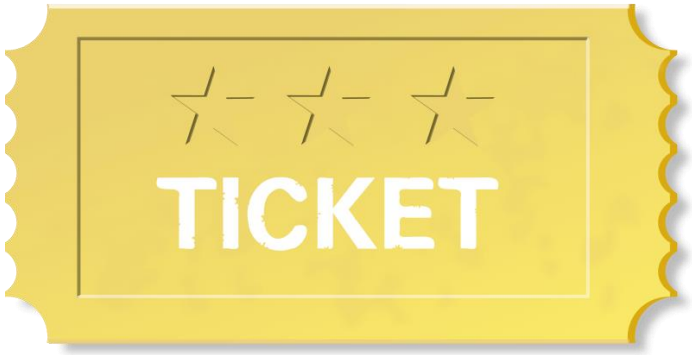
Syntax

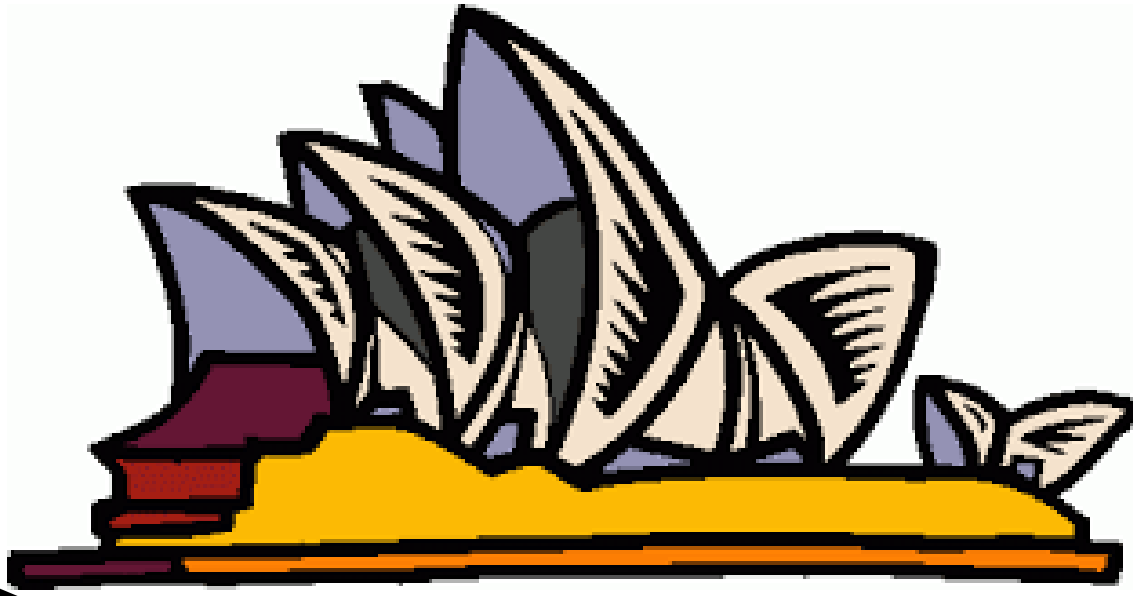


Gültiger Tag?
Gültige Zeit?

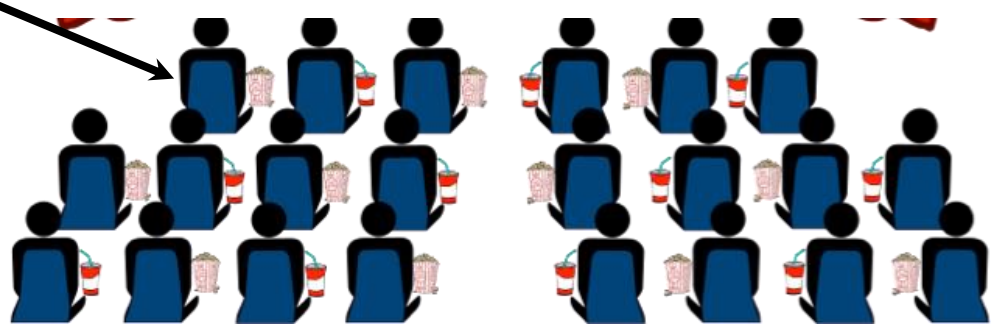


Semantik

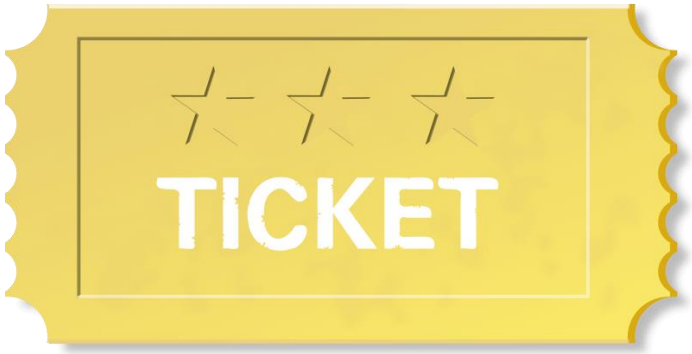




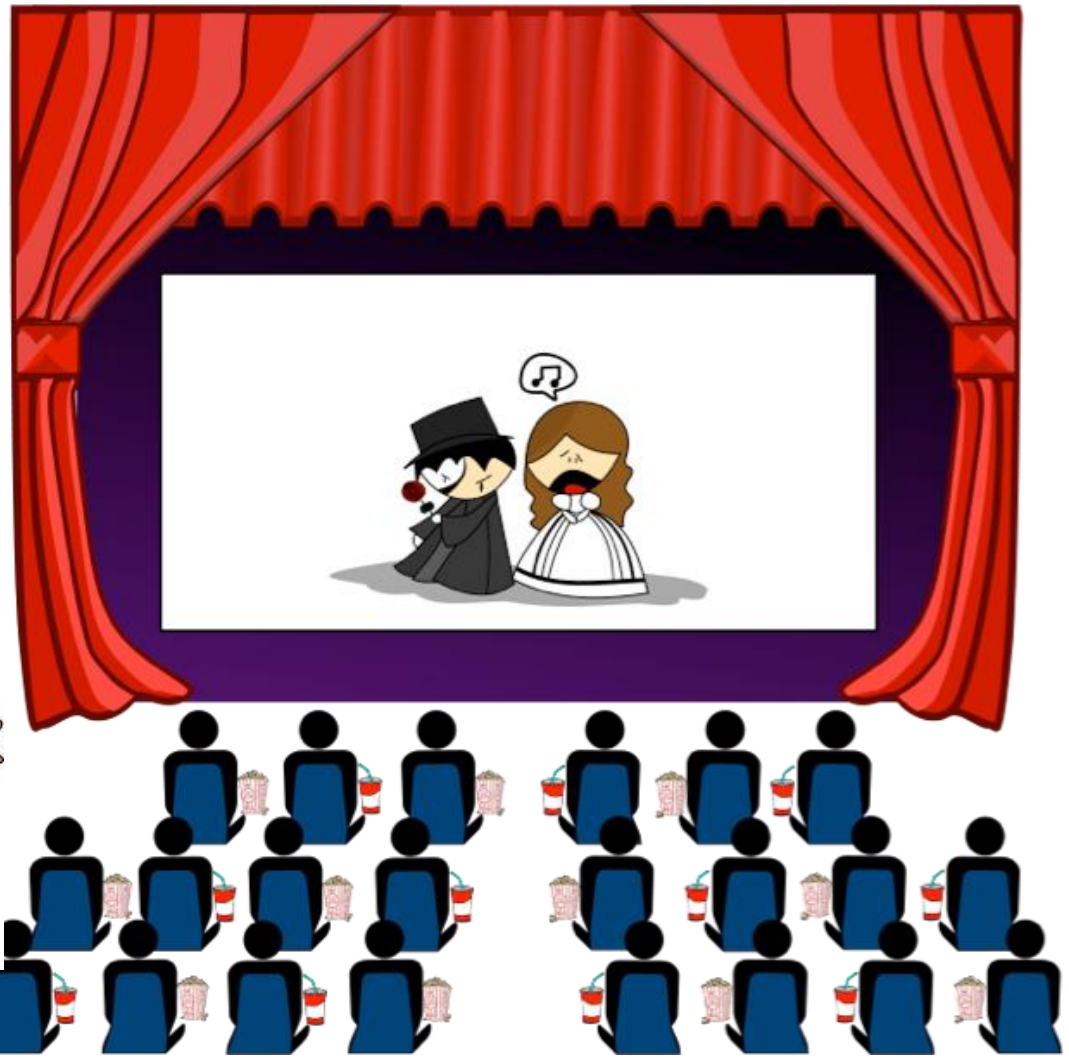
Das Recht auf
einen best. Sitz



Pragmatismus



Freude, die beim Ansehen des Stücks aufkommt



Beispiel „Deutsch“ als Sprache

Syntax

Syntax:

- Worte werden durch Buchstabensequenzen geformt
- Worte werden zusammen mit Satzzeichen gem. der deutschen Grammatik zu Sätzen aneinandergereiht
- Beispiele für **Gültigkeitsregeln** von Sätzen
  - Ein Satz hat Subjekt, Prädikat und Objekt und enden mit einem Punkt “Klaus programmiert Java11.“
  - Nebensätze mit um ... zu, werden immer mit Komma abgetrennt „Klaus lernt Java, um Softwareentwickler zu werden.“

Beispiel „Deutsch“ als Sprache

Semantik

Semantik:

- Worte sind gültig, wenn Sie
  - Im Duden vorkommen
  - Eine gebeugte Variante eines Worts im Duden sind
  - Eigennamen sind
  - ...
- Sätze sind gültig, wenn ihre Bedeutung sinnvoll ist. Gegenbeispiel:

Dunkel war’s, der Mond schien helle,
schneebedeckt die grüne Flur,
als ein Wagen blitzesschnelle,
langsam um die Ecke fuhr. [..]

Aus *Gertrud Züricher: Kinderlied und Kinderspiel im Kanton Bern. Verlag der Schweizerischen Gesellschaft für Volkskunde, Zürich 1902*

Beispiel „Deutsch“ als Sprache

Pragmatik

Pragmatik:

- Die Wirkung, die gesprochener oder geschriebener Text erzielt
- Zeigt, ob der Text verstanden und korrekt verarbeitet wurde
- Gegenbeispiel:

Gehen Sie alle nach Hause!
Jetzt!
Sofort!

Syntax und Semantik von Ausdrücken und Instruktionen

Ein Ausdruck, z.B. `X * Y`, ist kein Wert sondern mögliche **zukünftige Laufzeitwerte**

Eine Instruktion, z.B. `gamefield.display()` steht für eine **Operation, die zur Laufzeit ausgeführt werden soll**

Syntax	Semantik
Instruktion	Kommando
Ausdruck	Abfrage

Die **Syntax** eines Programms bestimmt die **Struktur** und die **Form** des Quellcodetexts

Die **Semantik** eines Programms ist die **Menge der Eigenschaften** seiner **potentiellen Ausführungen**

Syntaktische Struktur einer Java Klasse

```
class Example01 extends SimpleHamsterGame {

    void run() {
        gamefield.show();
        territory.initialize();


        paule.move();
        paule.move();
        paule.pickGrain();
        paule.pickGrain();
        paule.turnLeft();
        paule.turnLeft();
        paule.move();
        paule.move();
        paule.turnLeft();
        paule.turnLeft();
    }


}
```

Name der Klasse

Operations–deklaration

Operationsrumpf

Instruktionen

Operations–bezeichner

Example01.java

Mathematische Sprache

Denken Sie an Polynome (z.B., $f(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_0$). Beschreiben Sie Regeln, die die korrekte Syntax von Polynomen beschreiben sollen. Was ist die Semantik und der Pragmatismus von Polynomen?

Dauer: 7min

Ergebnis: Sammlung an der Tafel

Natürliche vs. Programmier- (artifiziellen) Sprachen

Man hat eine Leiter zur Sonne gestellt
Die Sonne ist schwarz
Die Mühle blüht
...
Du deiner dir dich
Du deiner dich dir.

Kurt Schwitters: Der Bahnhof (1918)

```
22156
7
3452
245
42
```

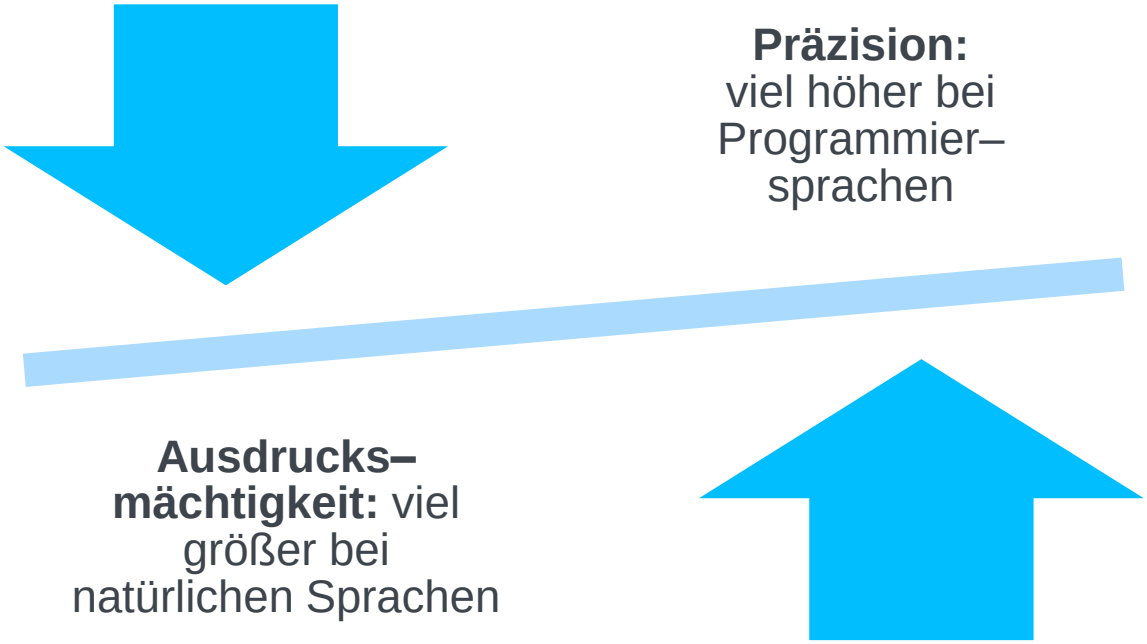
Java ist eine Programmier- (artifizielle) Sprache

```
class Example1 extends SimpleHamsterGame {

   void run() {
      game.displayInNewGameWindow();
      game.initialize();
      paule.move();
      paule.move();
      paule.pickGrain();
      paule.pickGrain();
      paule.turnLeft();
      paule.turnLeft();
   }

}
```

(Java) Satz

Example01.java

Programmier- vs. natürliche Sprachen: Unterschiede



Programmiersprachen sind Erweiterungen “mathematischer Notationen”.

Beziehung: Kommentare sind Schnipsel natürlicher Sprache, die im Quellcode von Programmen vorkommen.

Stilregel

Nutzen Sie Wörter der natürlichen Sprache
(vorzugsweise Englisch)
für alle **Bezeichner**, die Sie definieren

Beispiele:

```
paule, gamefield
```

Operationsnamen:

```
display, move
```

Java **Schlüsselwörter** sind englische Wörter

```
extends, do, for, void, ...
```

Darstellung der Syntax: Abstrakte Syntaxbäume (ASTs)

Syntaktische Struktur einer Java Klasse

```
class Example01 extends SimpleHamsterGame {

    void run() {
        gamefield.show();
        territory.initialize();


        paule.move();
        paule.move();
        paule.pickGrain();
        paule.pickGrain();
        paule.turnLeft();
        paule.turnLeft();
        paule.move();
        paule.move();
        paule.turnLeft();
        paule.turnLeft();
    }


}
```

Name der Klasse

Operations–deklaration

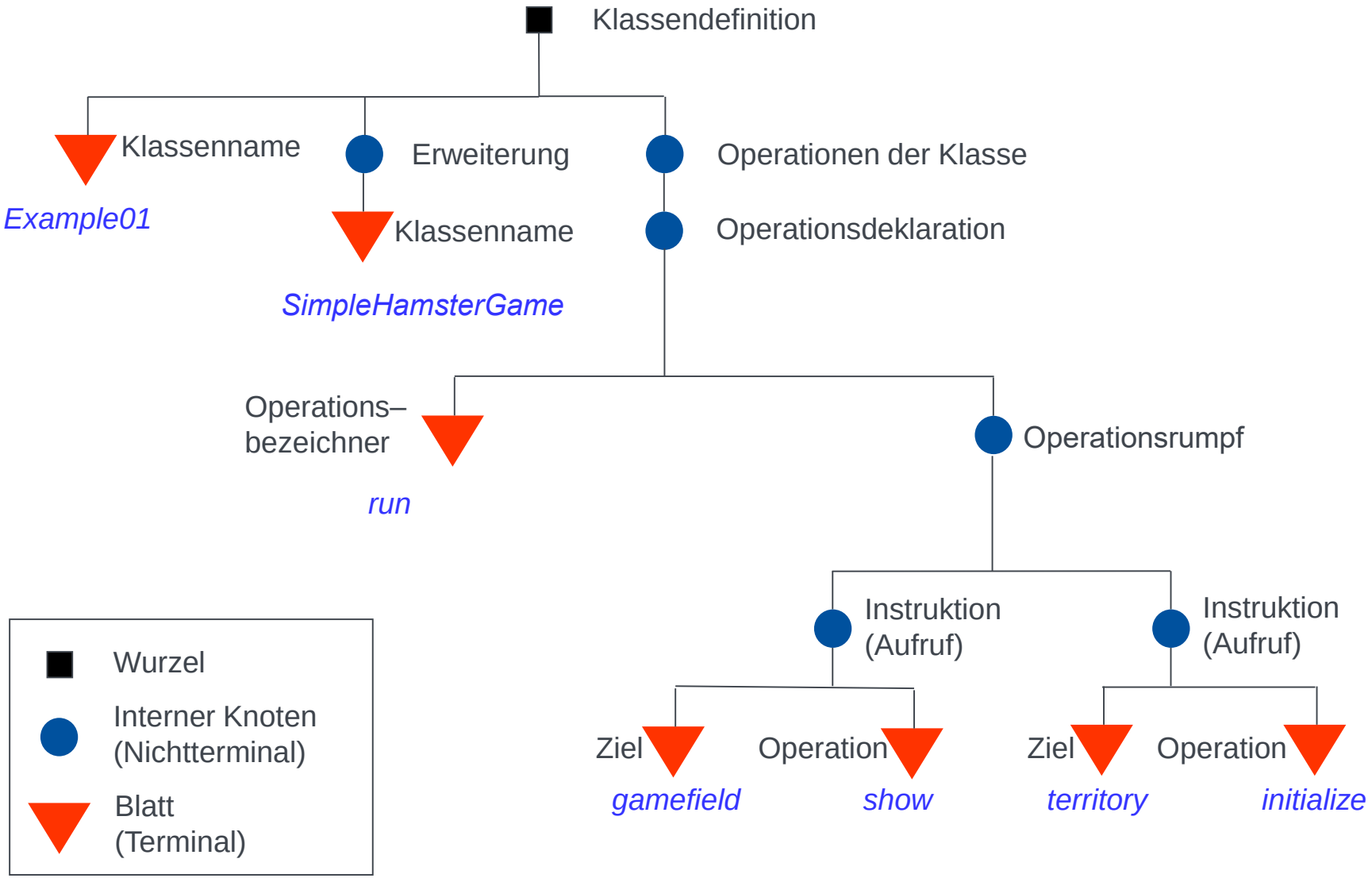
Operationsrumpf

Instruktionen

Operations–bezeichner

Example01.java

Alternative Darstellung: Abstract Syntax Tree (AST)



Abstrakter Syntaxbaum

- Zeigt die syntaktische Struktur auf
- **Nur Exemplare**: keine Schlüsselwörter oder andere Trennzeichen (daher ist der Baum auch **abstrakt**)
- Nutzt die Darstellung als Baum, analog zu Organigrammen von Unternehmen.

Abstrakter Syntaxbaum

- **Wurzel**: repräsentiert das Gesamtexemplar (äußerstes Rechteck)
- **Innere Knoten** (Nichtterminale): repräsentieren Unterstrukturen, die wiederrum Exemplare enthalten
- **Blätter** (Terminale): repräsentieren Exemplare ohne weitere Untergliederung

Die Syntax einer Programmiersprache ist durch die **Menge der Konstrukte** (z.B. Klasse, Operation, Operationsaufruf, ...) und die **Struktur dieser Konstrukte** (z.B. Operationsdeklarationen sind Teil der Klassendefinition) definiert.

AST

Zeichnen Sie den AST der folgenden Klassendefinition:

```
class CallingMethodsInSameClass extends ConsoleInteraction {
   void run() {
      printOne();
      printOne();
      printTwo();
   }

   void printOne() {
      console.println("Hello World");
   }

   void printTwo() {
      printOne();
      printOne();
   }
}
```

Dauer: 10min

Ergebnis: AST an der Tafel

Weitere Detailierung von Syntax und Semantik

Untere (Syntax-)ebene: Lexik

Die elementaren Bestandteile des Programmtexts sind die Lexikale **(Token)**:

- **Terminale**
  - **Bezeichner:** Namen; vom Programmierer gewählt, z.B., `paule` oder `display`
  - **Konstanten:** Selbsterklärende Werte, z.B., `34`
- **Schlüsselwörter**, z.B. **`class`**
- **Spezialsymbole:** z.B. Punkt, "." genutzt in Operationsaufrufen

Lexikale definieren die lexikalische Struktur der Sprache

Drei Ebenen der Beschreibung

Semantische Regeln definieren, was der **Effekt** von Programmen ist, die die syntaktischen Regeln erfüllen

Syntaktische Regeln definieren, wie man eine **AST Struktur** aus den gültigen Lexikalen ableiten kann

Lexikalische Regeln definieren, wie eine Zeichenkette in Lexikale **zerlegt** wird

Lexikalische Regeln für Bezeichner

Bezeichner

Im Allgemeinen: Ein Bezeichner beginnt mit einem Buchstaben gefolgt von keinen oder mehr Zeichen, die entweder Buchstaben oder Zahlen sein dürfen (0 bis 9).

In Java: Jeder Bezeichner muss mindestens aus einem Zeichen bestehen. Das erste Zeichen muss aus folgender Menge stammen: Buchstaben, Unterstrich oder Dollarzeichen. Das erste Zeichen darf daher keine Zahl sein. Die weiteren Zeichen dürfen aus der folgenden Menge sein: Buchstaben, Zahlen, Unterstrich, oder Dollarzeichen

Sie können ihre eigenen Bezeichner wählen, wie Sie mögen. Allerdings dürfen Schlüsselwörter nicht benutzt werden und Stilregeln müssen eingehalten werden!

Stilregeln für Bezeichner

Wählen Sie immer Bezeichner, die die Rolle des Elements in der Domäne klar identifizieren.

Bezeichner sind in Englisch zu verfassen.

Bei Operationen, nutzen Sie immer ganze Namen, keine Abkürzungen: **`turnLeft`**

Bei Bezeichnern aus mehreren Wörtern wird CamelCase benutzt: **`busStation`**

Für Klassen werden großgeschriebene Bezeichner im Singular benutzt: **`SimpleHamsterGame`**

Eine zusätzliche Detailebene der Semantik…

Statische Semantikregeln definieren Gültigkeitsregeln, die nicht in der Syntax erfasst wurden

Zum Beispiel:

```
gamefield.display()
```

Aber nicht:

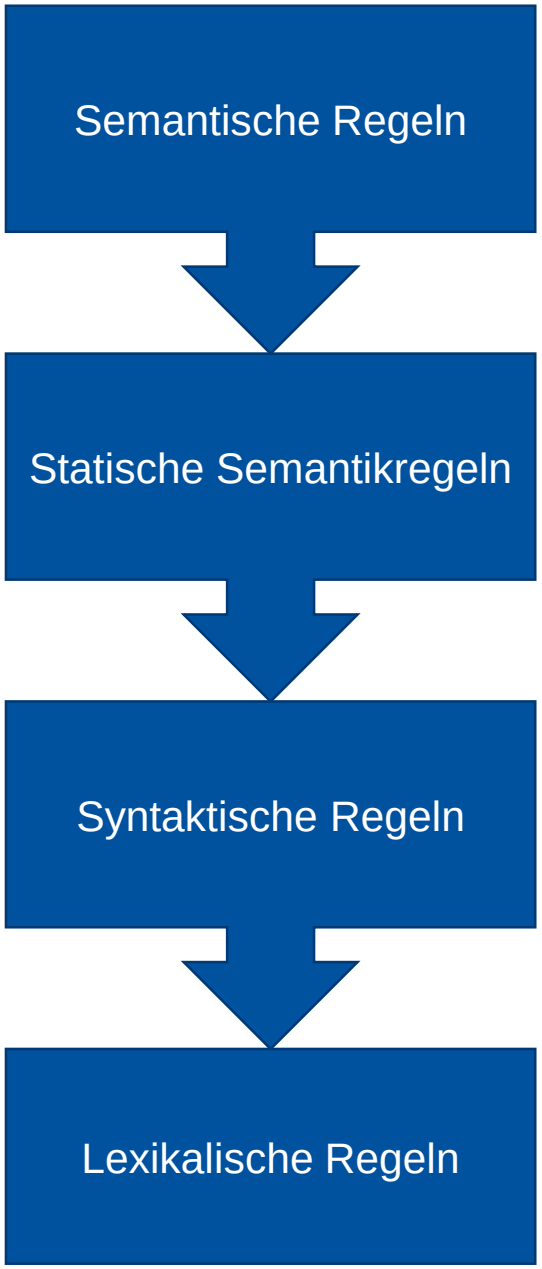
```
display.gamefield()
```

(Vergleiche in Deutsch:

Ich sehe das Meer

Aber nicht z.B.

Ich meere das Seh)



Zusammenfassung

Was sollten Sie nun können?

- Unterschiede zwischen Instruktionen und Ausdrücken kennen und erklären können
- Instruktionen und Ausdrücke in Programmcode identifizieren können
- Syntax, Semantik und Pragmatik erklären können
- Abstrakte Syntaxbäume herleiten können
- Kompilierfehler erkennen, analysieren, klassifizieren und erklären können
- Stilregeln für Namensgebung kennen und anwenden können

Universität Stuttgart

PSE:
Die Schnittstelle einer Klasse

Prof. Dr.-Ing. Steffen Becker

Vorlesung 4