Grundlagen der Programmierung

VL10: Objektorientierung

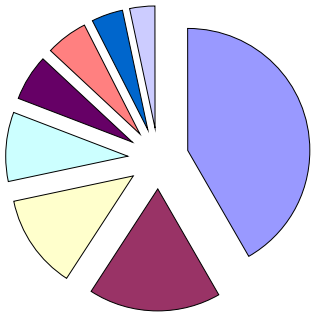
Prof. Dr. Samuel Kounev
M.Sc. Norbert Schmitt

Inhalt

- Treiber der Objektorientierung (historisch)
- Warum Objektorientierung?
- Was ist Objektorientierung?
- Die Philosophie und Grundprinzipien der Objektorientierung
  1. Abstraktion / Generalisierung (abstraction / generalization)
  2. Vererbung (inheritance)
  3. Kapselung (encapsulation)
  4. Polymorphismus (polymorphism)

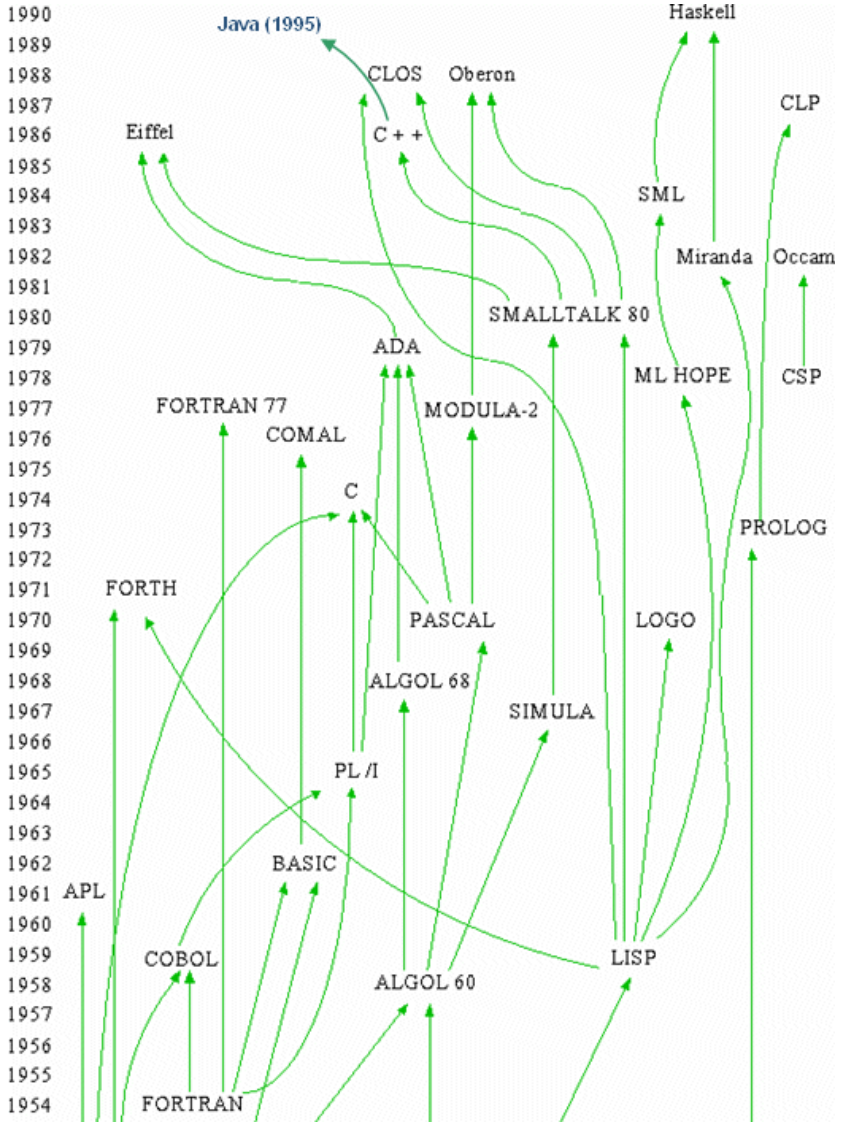
Die Treiber für die Objektorientierung (Ende der 1980er Jahre)

Grafische Benutzer-Oberflächen für den neuen PC-Massen-Markt (Programm-Komplexität vervielfacht sich)!



Die **Kosten für die Wartung** der Software sind zu hoch, Anpassungen dauern zu lange und kosten zu viel!

Softwarekrise führt zu immer größeren und imageschädigerenden Unglücken!



1973 Xerox PARC: Der „Alto“ erscheint

Erster Computer mit graphischer Benutzungsschnittstelle (GUI)
Vorbild für den legendären Apple Macintosh

- Monochromen Monitor mit einer Auflösung von 606 x 808 Pixel
- Unkodierte Tastatur mit 61 Tasten
- Drei-Tasten-Maus, nutzte Rastergrafik, Fenster, Menüs und Icons
- Über das Ethernet-Protokoll mit anderen Rechnern verbindbar
- Datanaustausch, E-Mail über Intranet
- WYSIWYG-Oberfläche
- Hat die Größe einer Kühltruhe

2000 Rechner verkauft

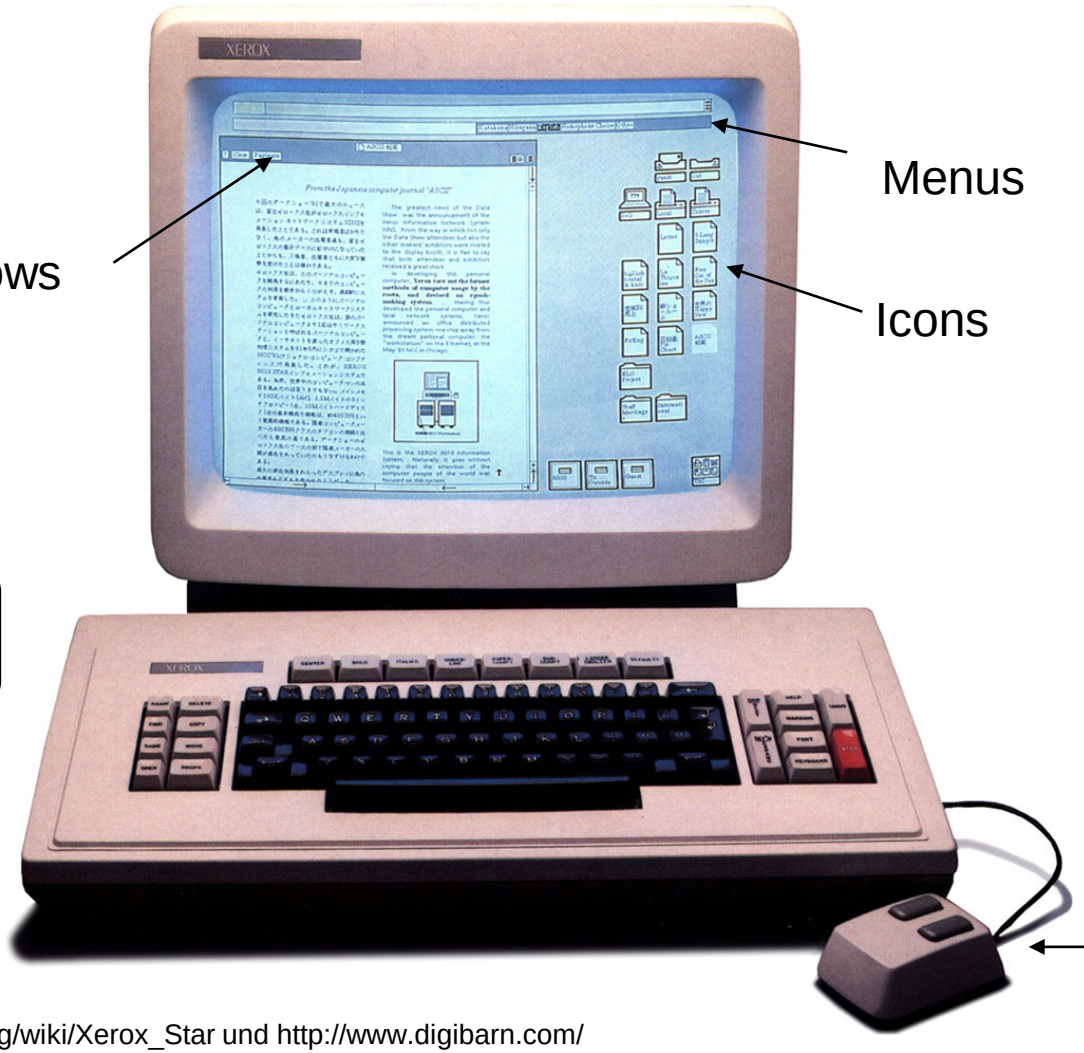
$32.000,-

Nachfolger ist der Xerox Star

Quelle: http://de.wikipedia.org/wiki/Xerox_Alto

April 1981: XEROX 8010 „Star“ erscheint

Diese größenreduzierte Weiterentwicklung des Alto kann als Urvater aller modernen WIMP-Systems (Windows, Icons, Menus, Pointing Devices, WYSIWYG) angesehen werden.



Basis-System

$75.000,-

$16.000,-

für jede weitere Workstation

Quellen: http://de.wikipedia.org/wiki/Xerox_Star und http://www.digibarn.com/

August 1981: Der IBM PC erblickt die Welt

Der Microcomputer wird zum standardisierten Industrie-Massenprodukt
Der VW-Käfer der IT ist geboren.

$3.000,-

Quelle: http://de.wikipedia.org/wiki/IBM_Personal_Computer und http://einestages.spiegel.de/static/topicalbumbackground/23321/charlie_chaplin_computerverkaeufer.html

1984: Apple Macintosh mit Mac OS 1.1

Der Apple Macintosh – die geniale Verschmelzung des PC-Konzepts mit den sichtbaren Vorteilen des XEROX 8010 „Star“

$2.495,-

Quelle: http://oldcomputers.net/macintosh.html

August 1984: Der IBM AT und erste Clones sind da!

Drittanbieter wie Hercules (Grafikkarten), Cherry (Tastaturen) und Compaq (Clones) und Microsoft (MS-DOS) nutzen die offene Architektur des IBM PC – und werden zu Global Playern!

$15.000,-

1984

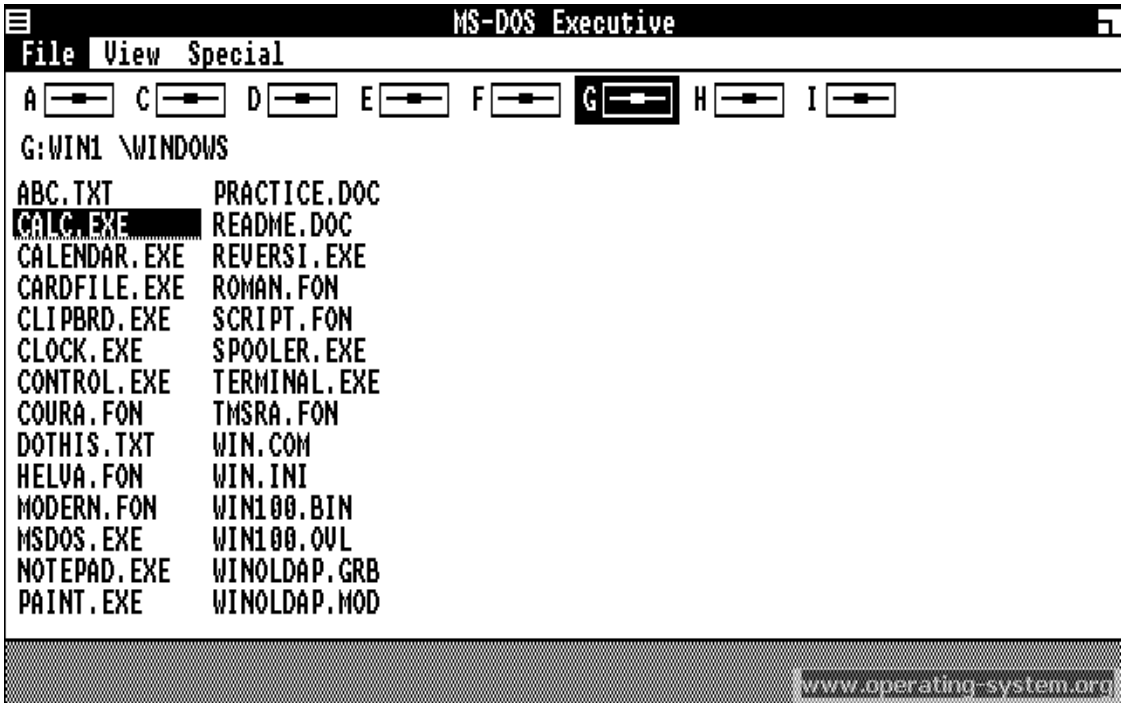
$2.995,-

1988

Quellen: http://www.historycorner.de/CoCo3/IBM/ibm5170AT.html und http://de.wikipedia.org/wiki/IBM_Personal_Computer/AT

1985: Desktop von Windows 1.0

Grafische Benutzeroberflächen die rein nach dem funktionsorientierten Ansatz erstellt waren, boten wenig Attraktivität



Quellen: http://toastytech.com/guis/win101executive.gif, http://www.operating-system.org/betriebssystem/_german/bs-win1.htm und http://osdp.bplaced.net/de/screen_gallery.php?bsgfx=microsoft/win1/windows1-scr-08.png

1992: Desktop von Windows 3.11

Grafische Benutzeroberflächen sind mit dem bislang dominierenden funktionsorientierten Ansatz kaum zu beherrschen

Fenster

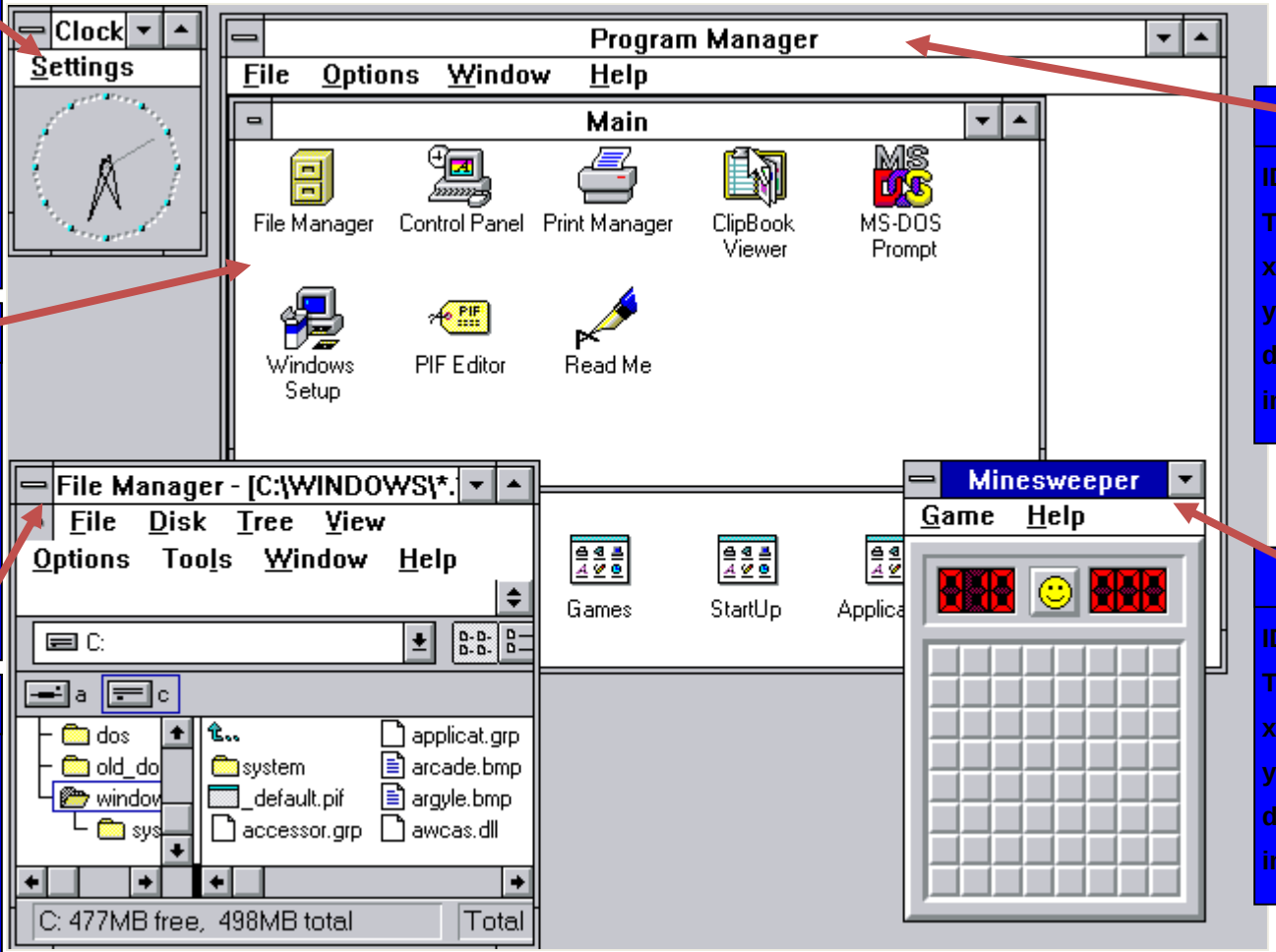
ID: 2
Titel: Clock
xPos: 0
yPos: 0
displayMode: resizeable
inputFocus: false

Fenster

ID: 4
Titel: Main
xPos: 110
yPos: 50
displayMode: resizeable
inputFocus: false

Fenster

ID: 5
Titel: File Manager
xPos: 10
yPos: 240
displayMode: resizeable
inputFocus: false



Fenster

ID: 1
Titel: Program Manager
xPos: 100
yPos: 5
displayMode: resizeable
inputFocus: false

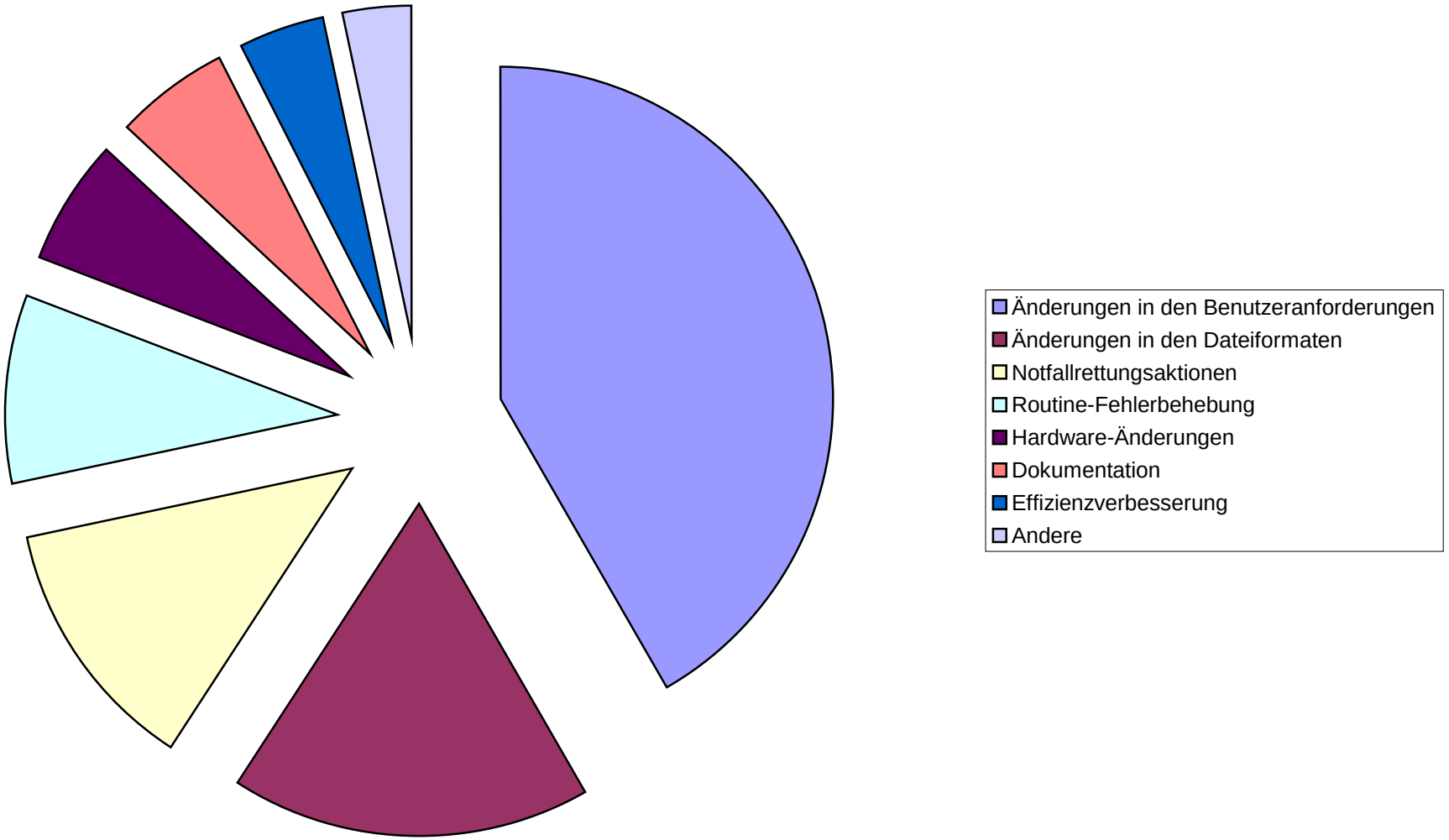
Fenster

ID: 3
Titel: Minesweeper
xPos: 500
yPos: 240
displayMode: resizeable
inputFocus: true

Quelle: http://www.techairlines.com/2009/10/22/a-brief-history-of-microsoft-windows/

Softwarewartung

Softwarewartung machten Ende der 1980ger ca. 70% der Gesamtkosten eines SW-Projekt aus



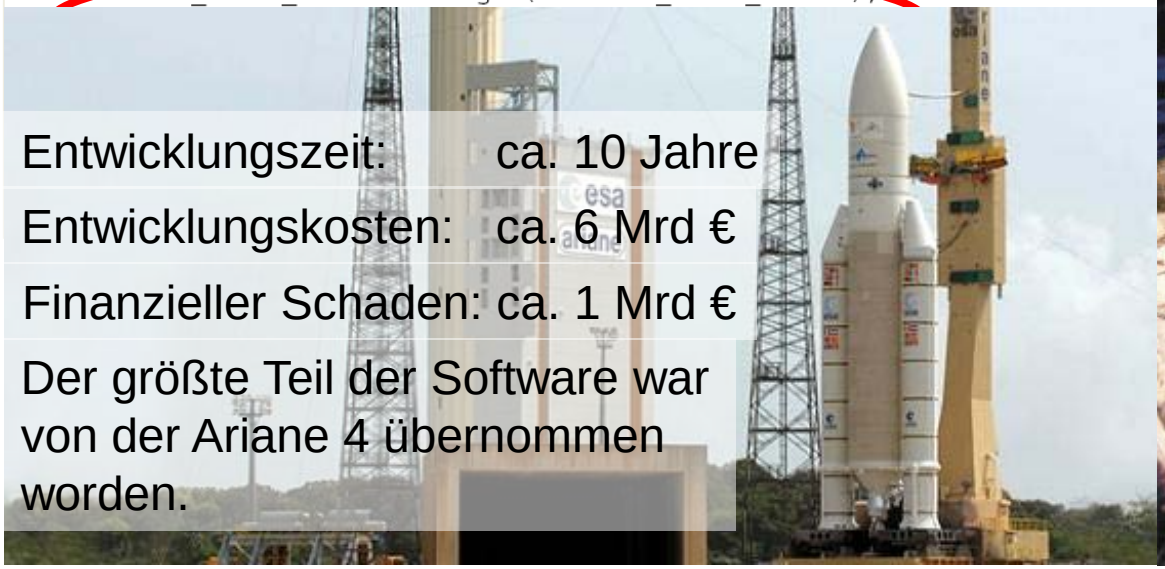
Softwarekrise

- Der Begriff der „Softwarekrise“ entsteht bereits in den 60gern
- Durch besonders spektakuläre Fälle erreichte dieser Begriff Ende der 70ger eine breite Öffentlichkeit:
  - 1979 : **Studie zu Softwareprojekten** (USA), inges. 7 Mio USD
    75% der Ergebnisse nie eingesetzt
    19% der Ergebnisse stark überarbeitet
    6% benutzbar
  - 1981 : **US Air Force** Command & Control Software überschreitet Kostenvoranschlag fast um den Faktor 10 (3,2 Mio USD)
  - 1984 : **Überschwemmungskatastrophe** im französischen Tarntal, weil Steuercomputer nach Überlaufwarnung Schleusen öffnet
  - 1990 : Ausfall des **Telefonnetzes** in den USA
  - 1996 : Absturz der **Ariane 5** wegen eines Software-Fehlers

Explosion der Ariane 5 am 4. Juni 1996 auf ihrem Jungfernflug

Ada-Programm des Trägheits-N (Ausschnitt):

```
...
declare
  vertical_veloc_sensor: float;
  horizontal_veloc_sensor: float;
  vertical_veloc_bias: integer;
  horizontal_veloc_bias: integer;
  ...
begin
  declare
    pragma suppress(numeric_error, horizontal_veloc_b
  begin
    sensor_get(vertical_veloc_sensor);
    sensor_get(horizontal_veloc_sensor);
    vertical_veloc_bias := integer(vertical_veloc_sensor);
```



Entwicklungszeit: ca. 10 Jahre

Entwicklungskosten: ca. 6 Mrd €

Finanzieller Schaden: ca. 1 Mrd €

Der größte Teil der Software war von der Ariane 4 übernommen worden.

Quelle: http://www4.in.tum.de/lehre/seminare/ps/WS0203/desaster/Riedl-Arianne5-Ausarbeitung-27-11-02.pdf

Ausfall des Telefonnetzes in den USA am 15. Januar 1990

```
switch expression {
            ...
            case (value):
                 if (logical) {
                      <statements>
                      break;
                 } else {
                      <statements>
                 }
                 <statements>
            ...
```

Dauer: 9 Stunden

Folge: 70 Mio (von 134 Mio) Ferngespräche nicht vermittelt

Finanzieller Schaden: ca. 500 Mio €

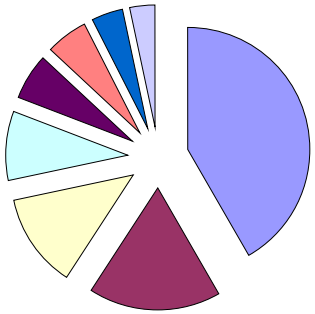
Auslöser: Kettenreaktion ausgelöst durch eine Schaltzentrale in Manhatten

Ursache: Software Update mit falsch eingesetztem break-Befehl; dieses Update war zuvor auf alle Schaltzentralen verteilt worden

Quelle: http://formal.iti.kit.edu/~beckert/teaching/Seminar-Softwarefehler-SS03/Ausarbeitungen/sesterhenn.pdf und http://formal.iti.kit.edu/~beckert/teaching/Seminar-Softwarefehler-SS03/Folien/sesterhenn.pdf

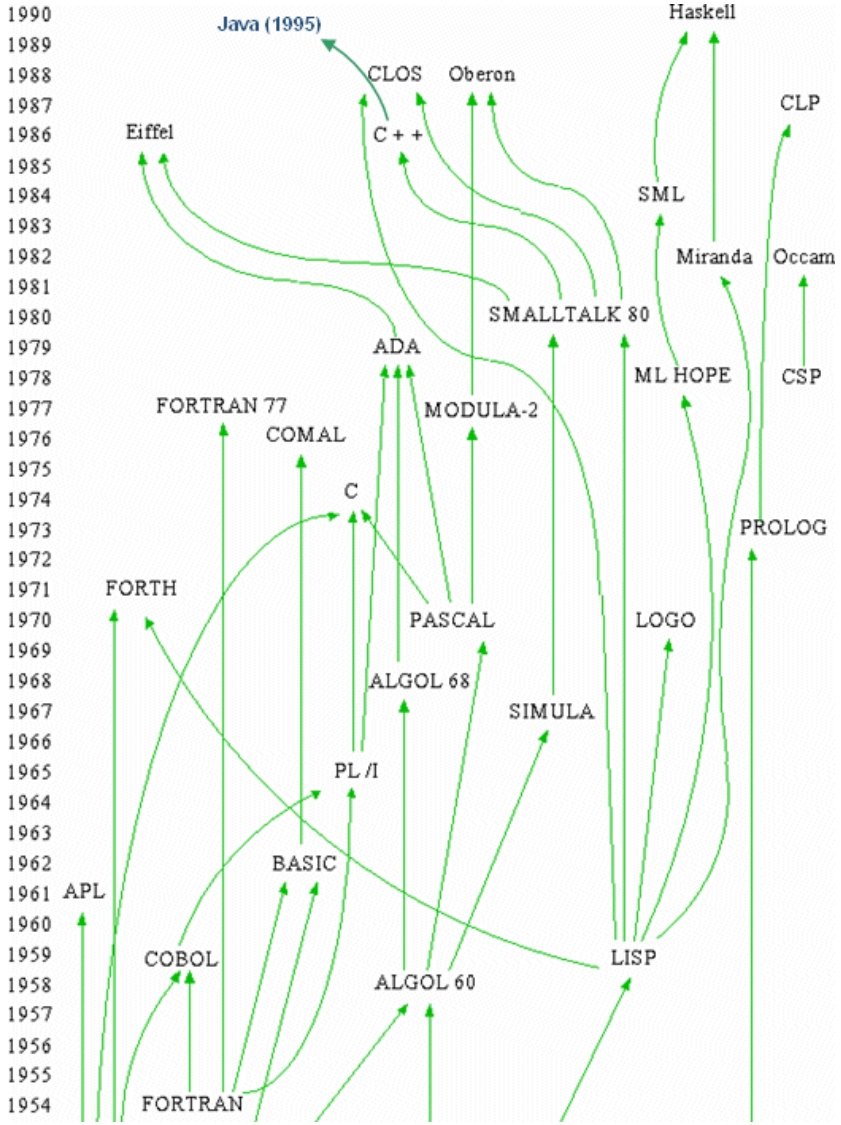
Die Treiber für die Objektorientierung (Ende der 1980er Jahre)

Grafische Benutzer-Oberflächen für den neuen PC-Massen-Markt (Mac OS, IBM OS/2, MS Windows)!



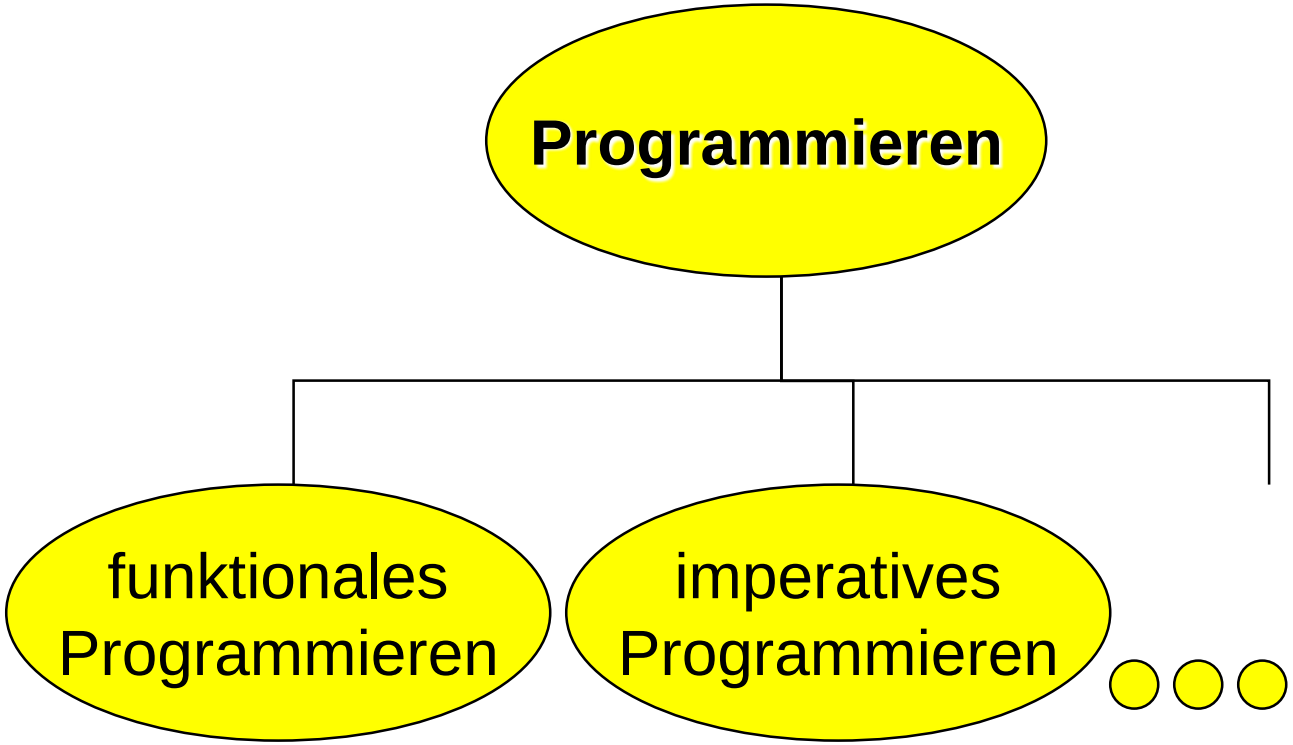
Die **Kosten für die Wartung** der Software sind zu hoch, Anpassungen dauern zu lange und kosten zu viel!

Softwarekrise führt zu immer größeren und imageschädigerenden Unglücken!



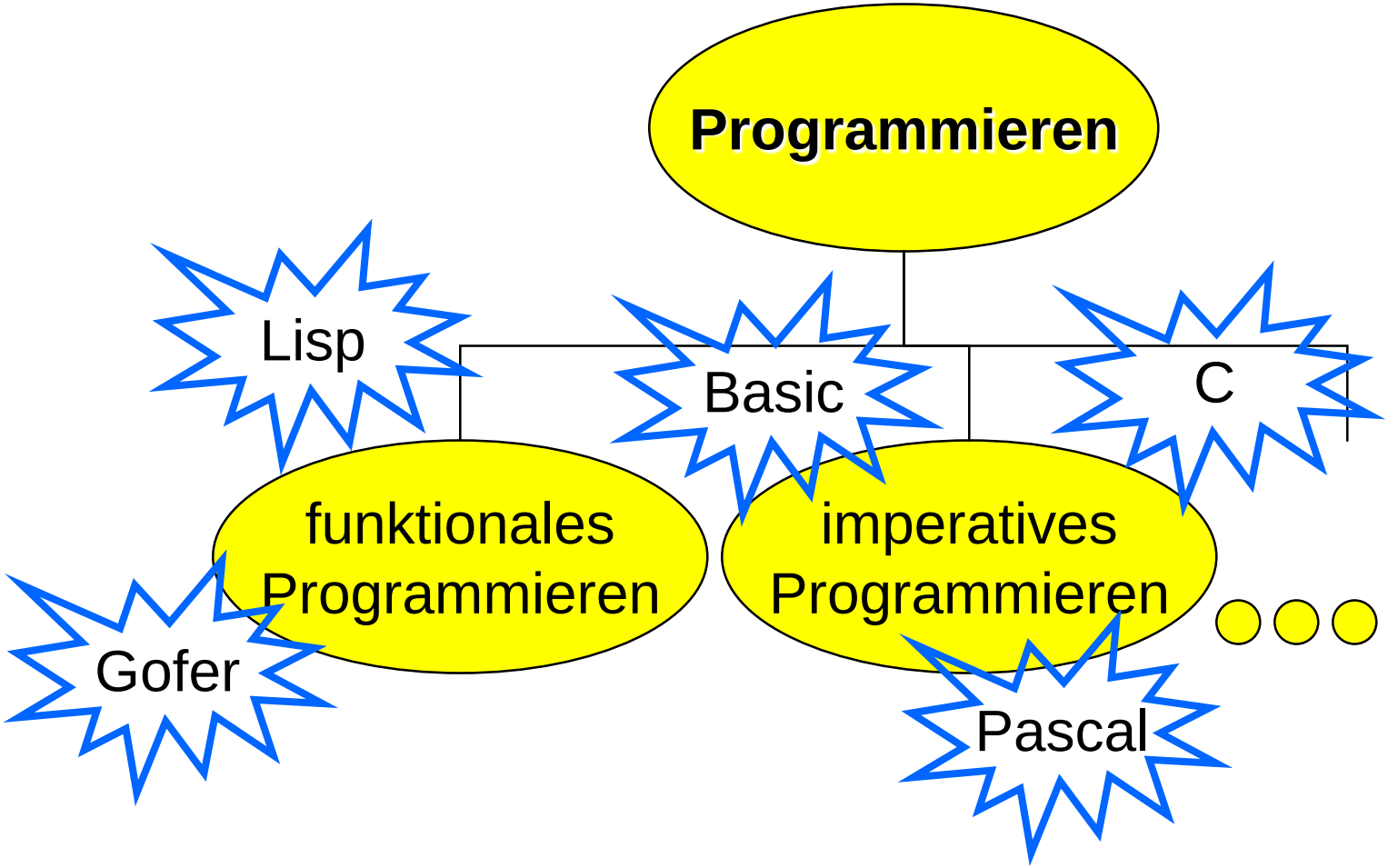
Was bedeutet „Objektorientierung“?

Es gibt verschiedene Philosophien, wie Programme auf dem Rechner verfasst werden sollten, sogenannte **Paradigmen**



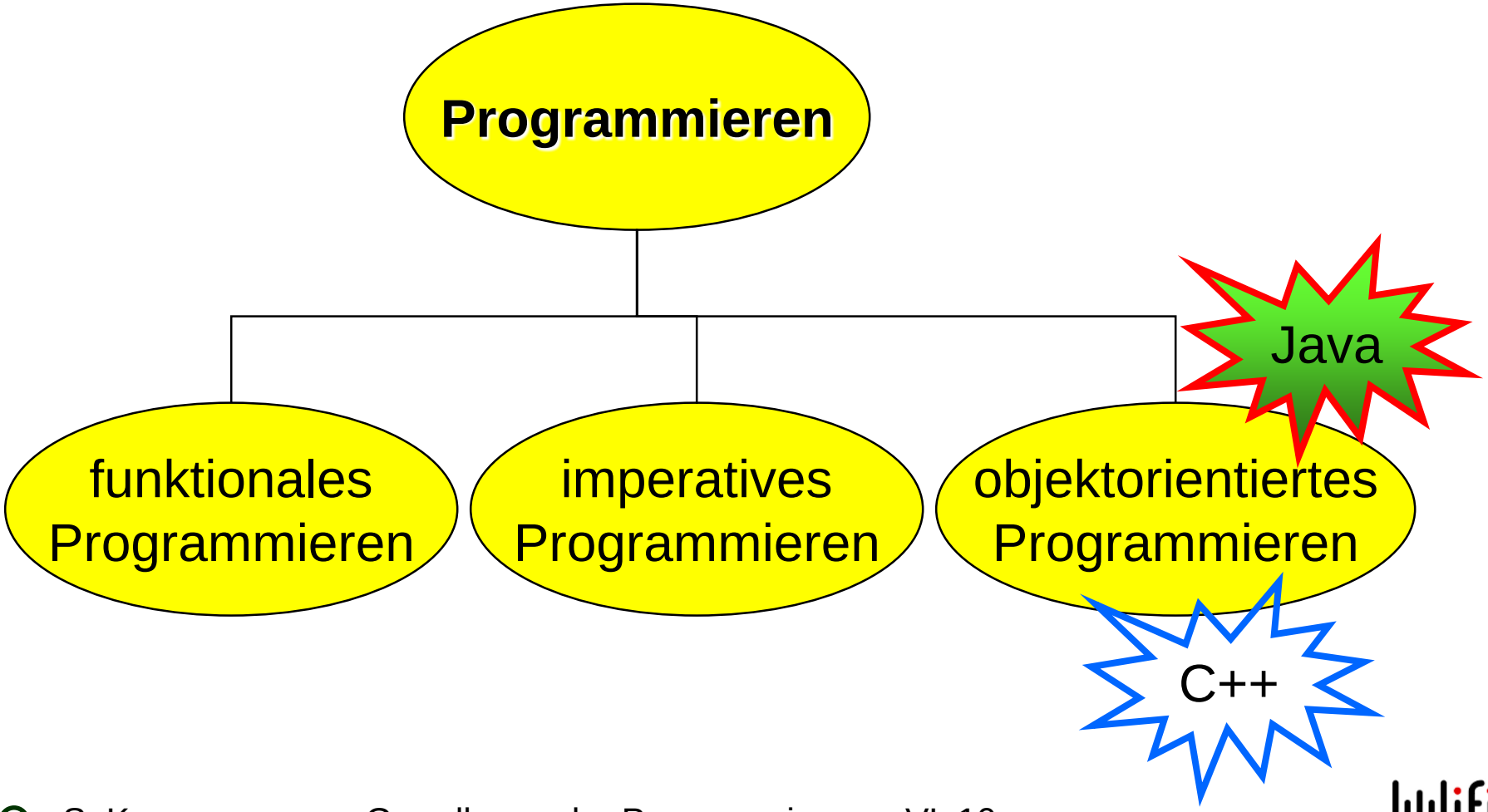
Was bedeutet „Objektorientierung“?

In der Entwicklungsgeschichte der Programmiersprachen haben sich zu den verschiedenen Paradigmen unterschiedliche Sprachen entwickelt...



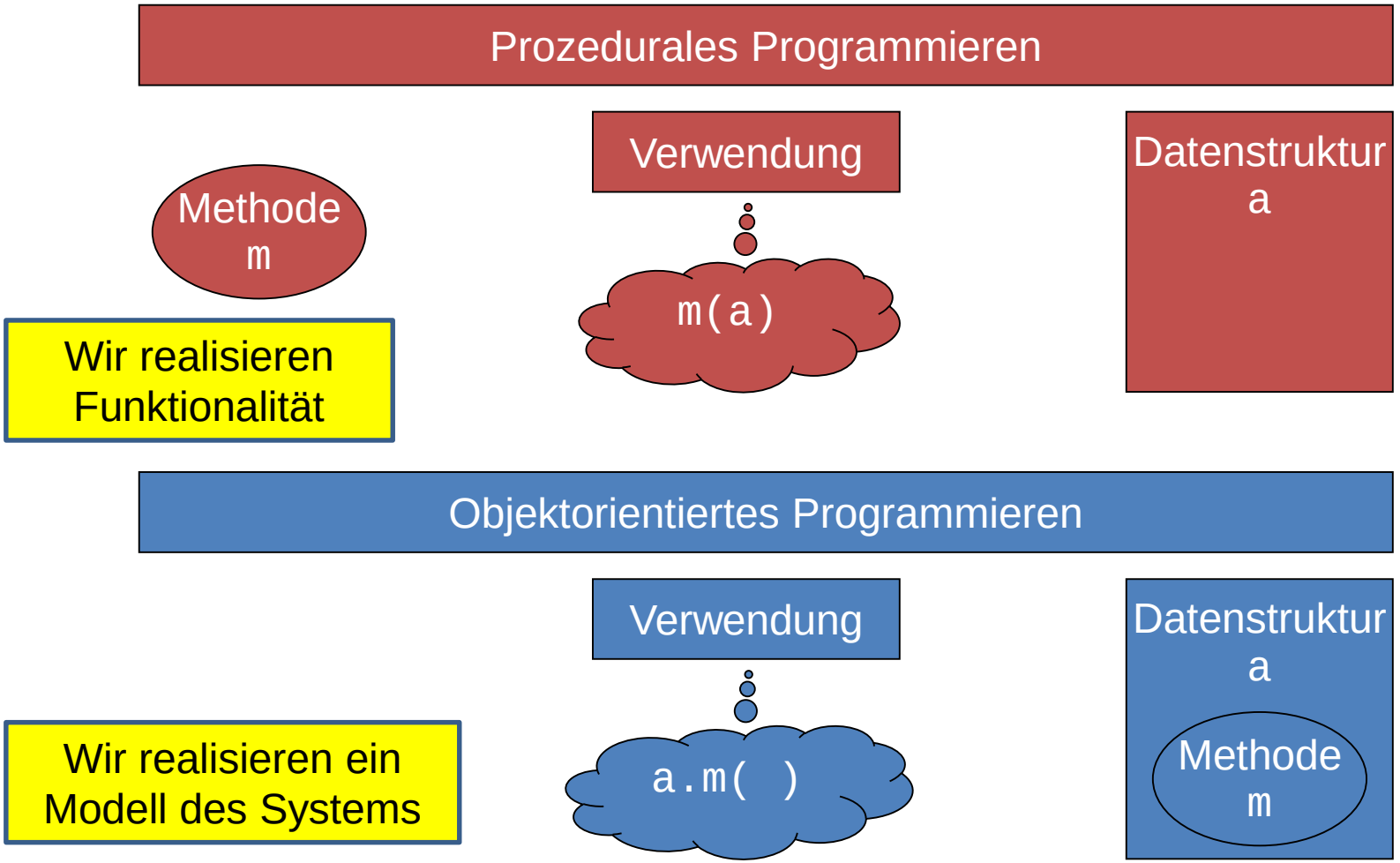
Was bedeutet „Objektorientierung“?

Der Programmiersprache Java liegt ein noch relativ junger Ansatz zugrunde: das **objektorientierte Programmieren**



Paradigmenwechsel

Paradigmenwechsel beim Übergang von der Dominanz der Methoden hin zu der Dominanz der Datenstrukturen



Objektbasierung vs. Objektorientierung

- Verschiedene Formen des objektorientierten Ansatzes
  - **Objektbasierte Programmierung** ist die Entwicklung von Softwaresystemen als Sammlung von Implementierungen abstrakter Datentypen

Struktur = Vererbungsbaum

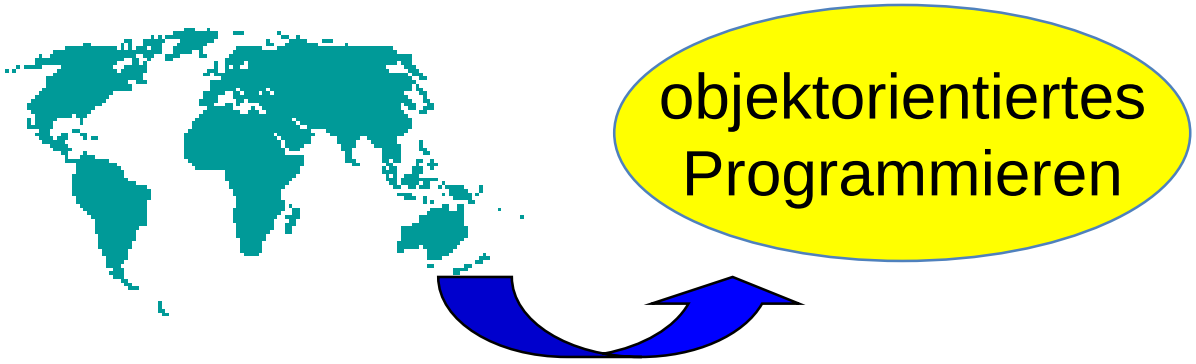
  - **Objektorientierte Programmierung** ist die Entwicklung von Softwaresystemen als strukturierte Sammlung von Implementierungen Abstrakter Datentypen

"Object-oriented software construction is the building of software systems as structured collections of possibly partial abstract data type implementations."

Bertrand Meyer, Object-Oriented Software Construction, 2nd Ed., Prentice Hall PTR, 2009, page 147

Was bedeutet „Objektorientierung“?

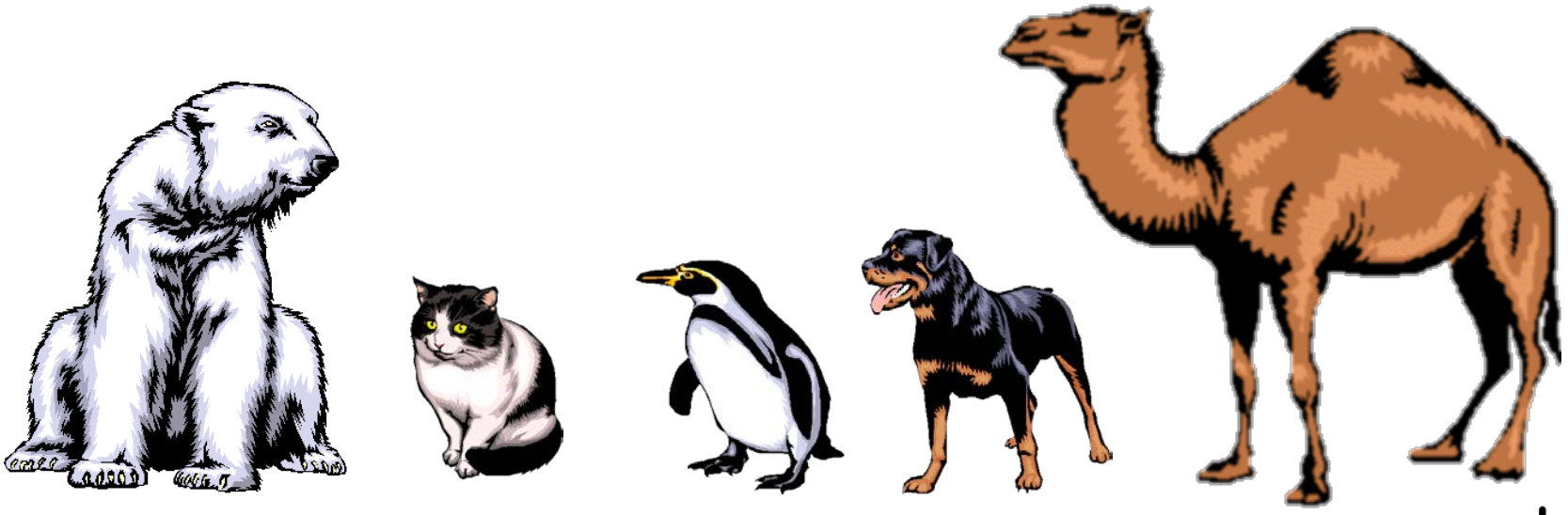
- Objektorientierte Programmierung geht von dem Grundsatz aus, dass ein wie auch immer geartetes Programm meist einen Teil der Welt widerspiegelt, in der wir leben
- Wenn wir also ein Modell dieser Welt auf dem Computer erzeugen wollen, sollten wir die einzelnen **Objekte**, mit denen wir im Leben konfrontiert werden, auf dem Computer abbilden können



Was bedeutet „Objektorientierung“?

Beispiel

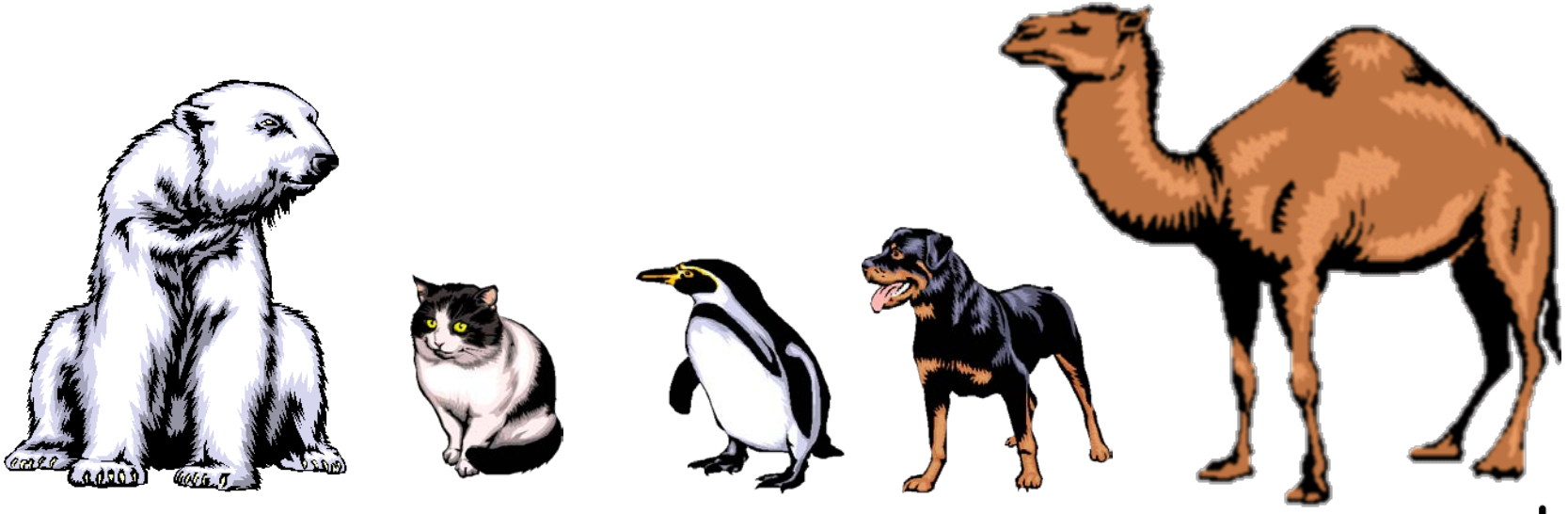
Wir wollen auf unserem Computer verschiedene Tiere „simulieren“



Was bedeutet „Objektorientierung“?

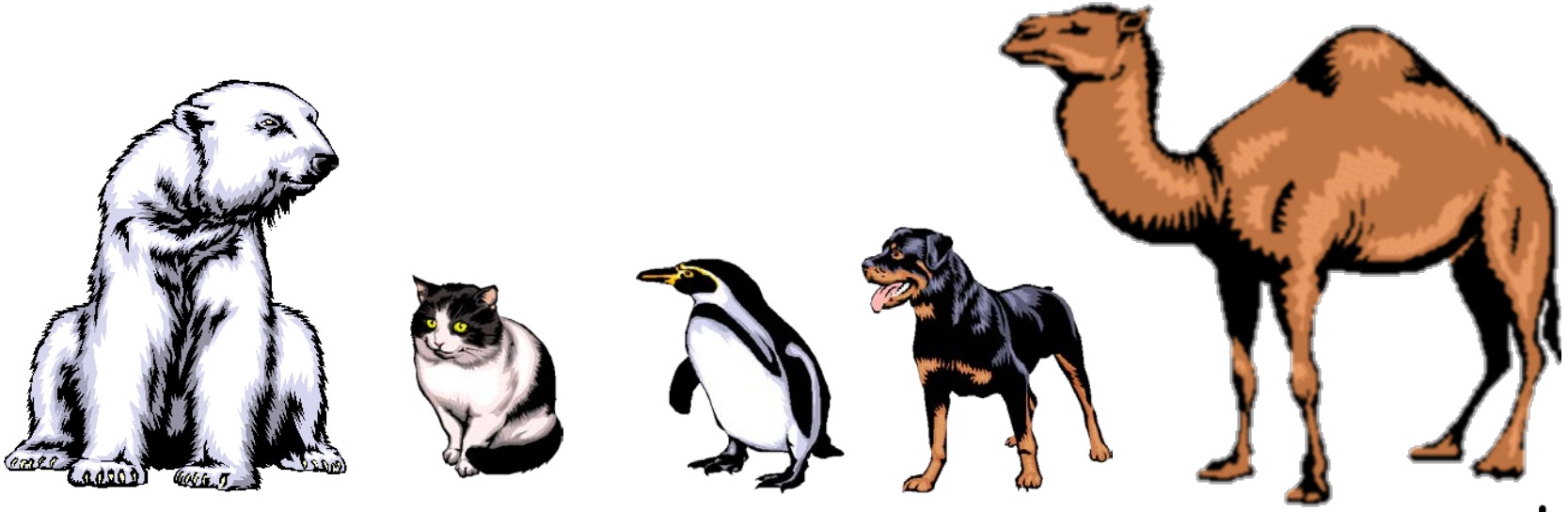
- Würden wir (wie bisher) imperativ programmieren, so müssten wir in Java eine Unmenge von Daten und Methoden erstellen, um das Verhalten der einzelnen Tiere abzubilden

Unser Programm würde unübersichtlich und kompliziert!



Was bedeutet „Objektorientierung“?

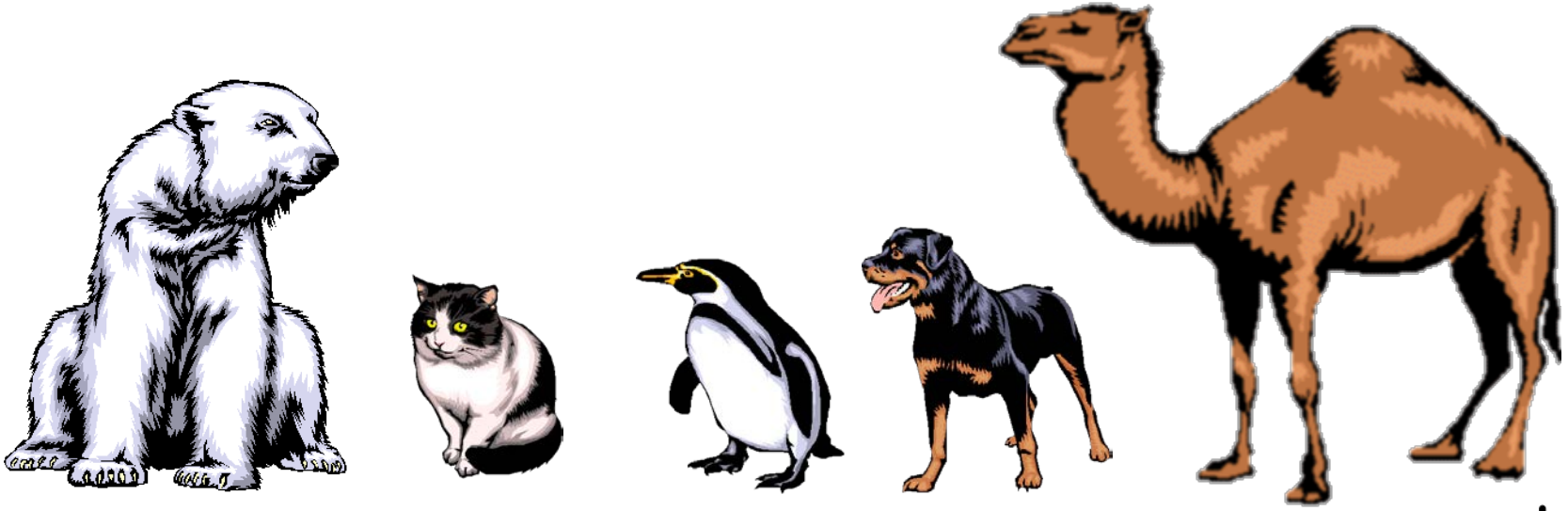
- Mit Hilfe des objektorientierten Ansatzes umgehen wir dieses Problem, indem wir die auftretenden Objekttypen durch verschiedene Klassen (abstrakte Datentypen, Schlüsselwort **`class`**) modellieren



Was bedeutet „Objektorientierung“?

- Mit Hilfe des objektorientierten Ansatzes umgehen wir dieses Problem, indem wir die auftretenden Objekttypen durch verschiedene Klassen (abstrakte Datentypen, Schlüsselwort **`class`**) modellieren

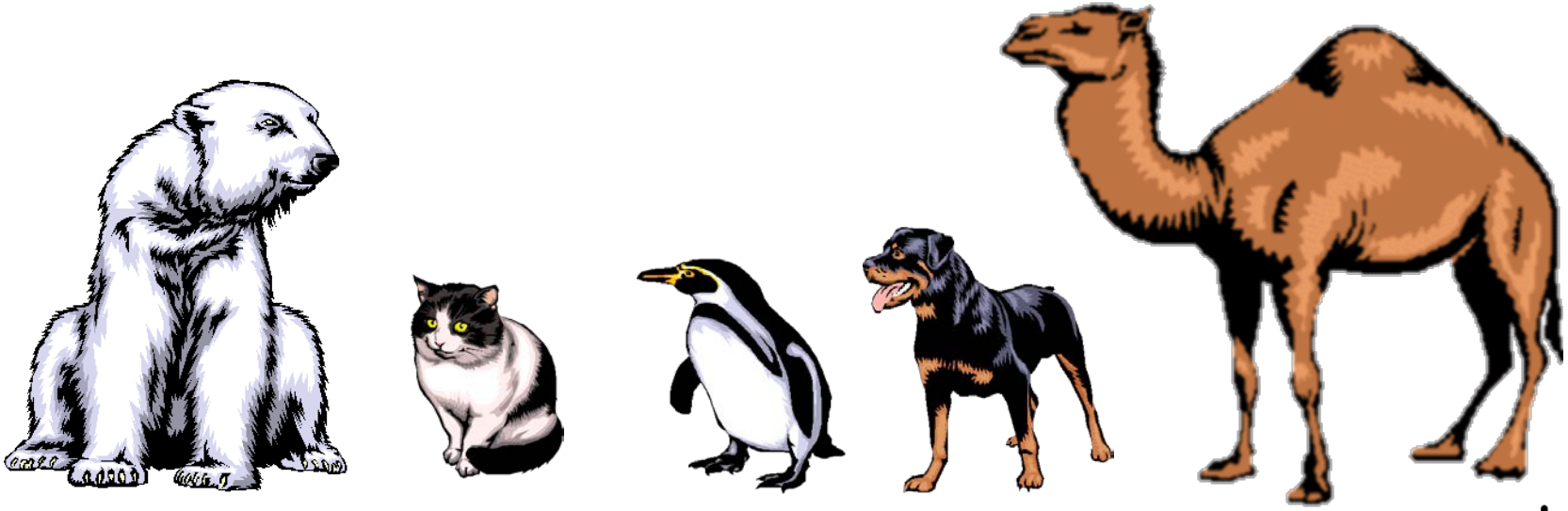
Eisbaer	Katze	Pinguin	Hund	Dromedar



Was bedeutet „Objektorientierung“?

- Jede dieser Klassen repräsentiert die gesammelten Eigenschaften eines Objekttyps und stellt somit sozusagen dessen „Bauplan“ (abstrakter Datentyp) dar

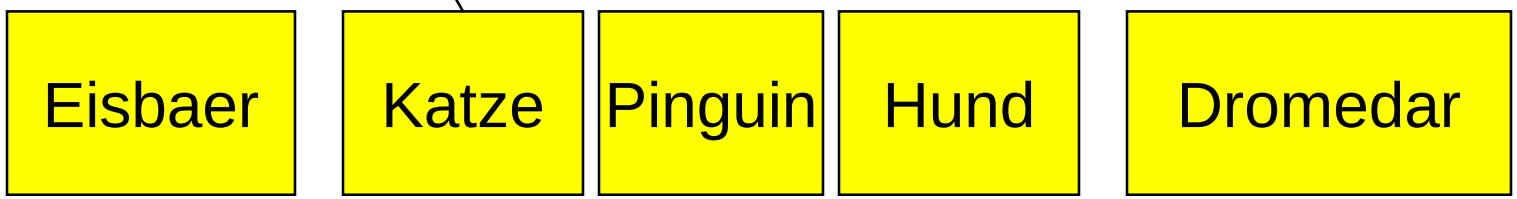
Eisbaer	Katze	Pinguin	Hund	Dromedar



Was bedeutet „Objektorientierung“?

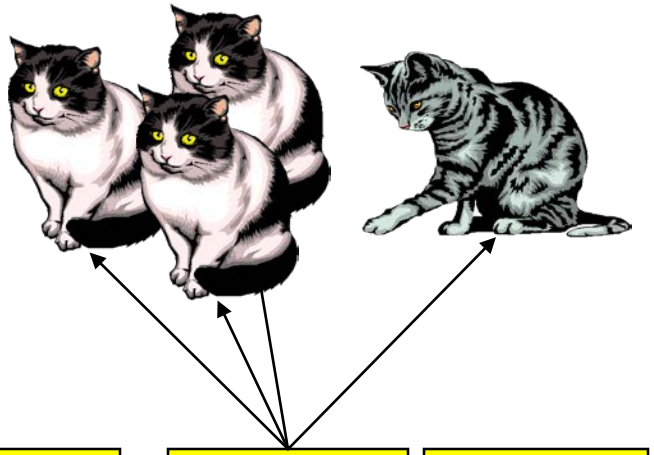
- Wollen wir nun in unserem Programm etwa ein Exemplar der Gattung „Katze“ verwenden, so können wir unsere Klassenbeschreibung benutzen, um aus ihr das gewünschte Objekt zu erzeugen

- Man bezeichnet diesen Vorgang als **Instantiierung**



Was bedeutet „Objektorientierung“?

- Natürlich ist es auch möglich, anhand einer Klassenbeschreibung mehrere Objekte zu erzeugen
- Wie bei verschiedenen Häusern, die nach dem selben Bauplan gebaut wurden, ist jedes Objekt individuell verschieden und kann seine eigenen Besonderheiten haben



Eisbaer | Katze | Pinguin | Hund | Dromedar

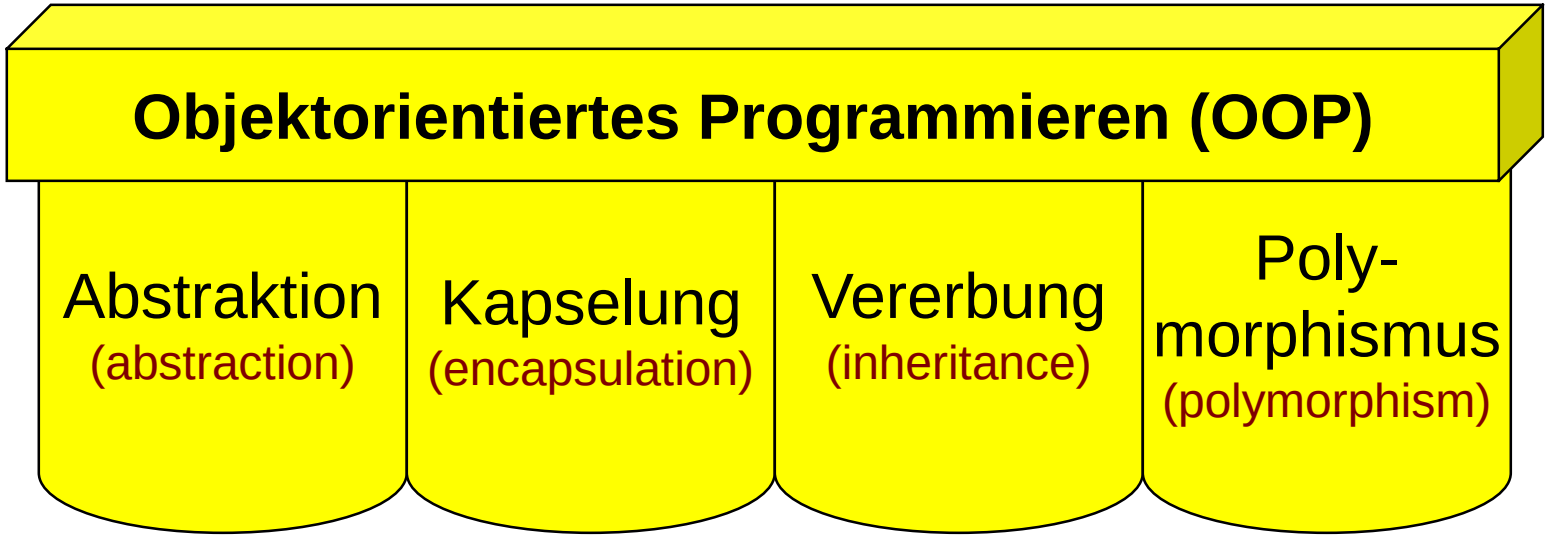
Was bedeutet „Objektorientierung“?

- Egal jedoch, wie viele **Instanzen** unserer Klasse wir bilden - es handelt sich stets um ein und denselben Bauplan, welchen wir beliebig oft als Vorlage verwenden können

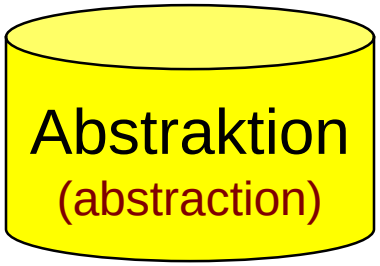
Eisbaer | Katze | Pinguin | Hund | Dromedar

Das objektorientierte Paradigma

- Die objektorientierte Philosophie stützt sich allgemein auf vier grundlegende Prinzipien:



Das objektorientierte Paradigma



ist das Zusammenfassen mehrerer Klassen mit gemeinsamen Merkmalen zu einer allgemeineren Oberklasse

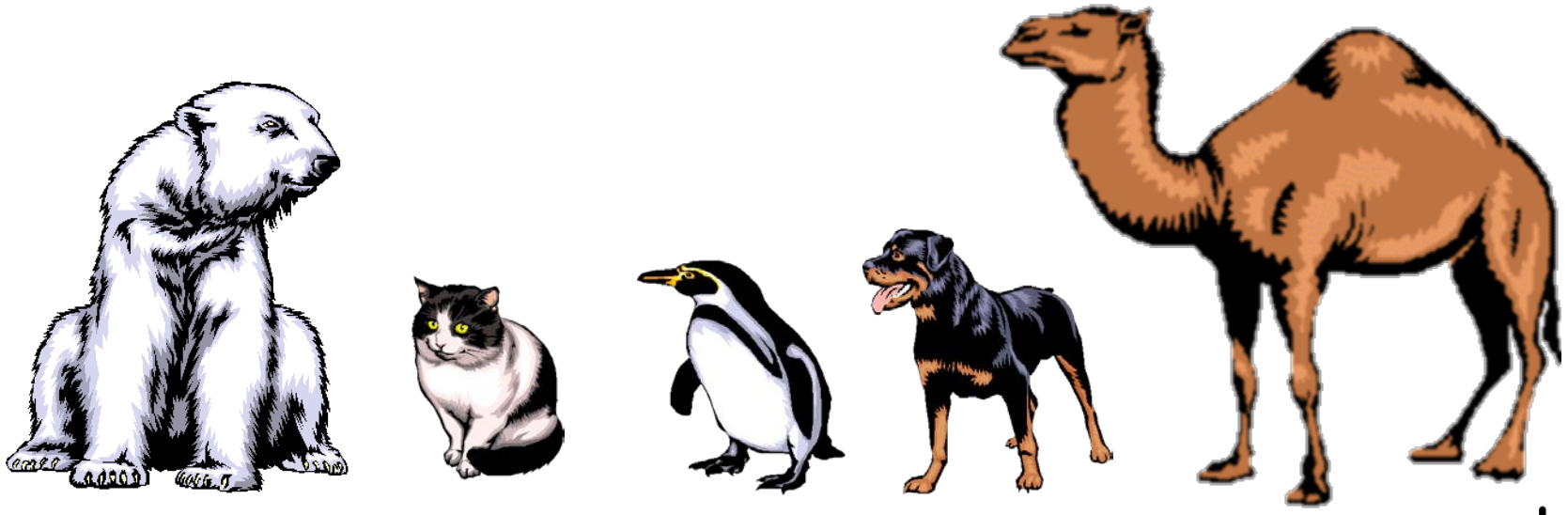
Ziel der **Abstraktion** ist es,

- eine Hierarchie zu erzeugen, in der Objekte wie in der wahren Welt miteinander in Beziehung stehen,
- gemeinsame Eigenschaften verschiedener Klassen einheitlich zu modellieren und somit
- Entwicklungszeit zu sparen und
- eine Vielzahl von Fehlerquellen auszuschalten

Das objektorientierte Paradigma

Beispiel

Wir wollen unsere fünf Tierarten in verschiedene Kategorien unterteilen

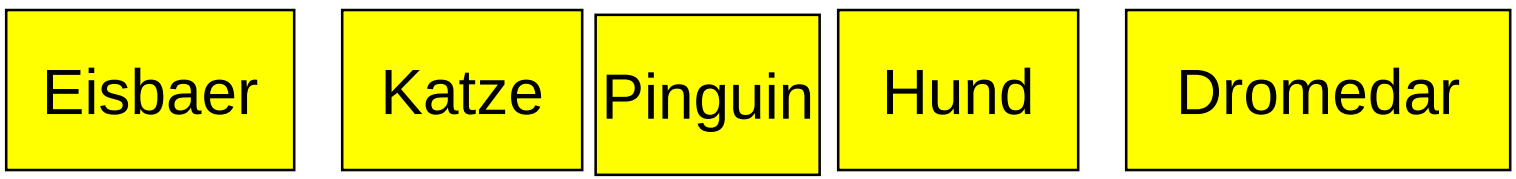


Das objektorientierte Paradigma

Beispiel

Wir wollen unsere fünf Tierarten in verschiedene Kategorien unterteilen

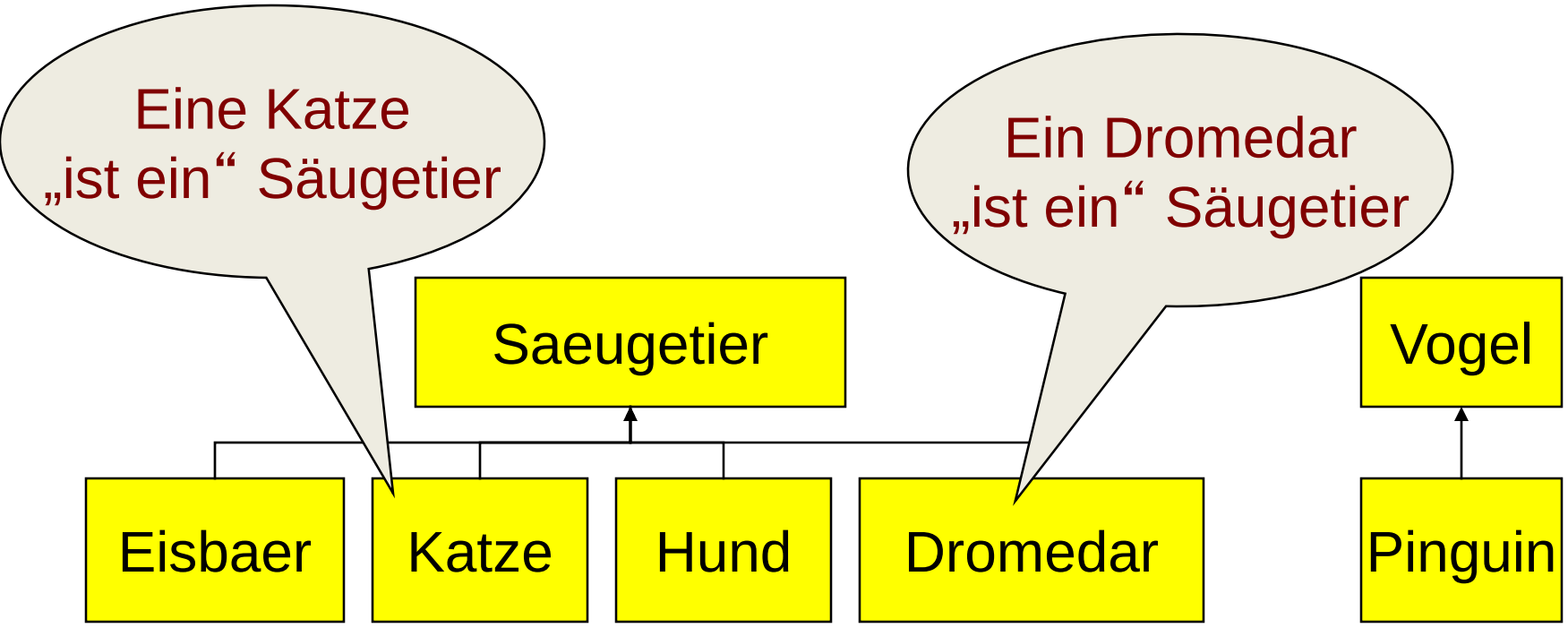
Wir wollen als erstes Kriterium unterscheiden, ob es sich bei den Tieren um **Säugetiere** oder **Vögel** handelt



Das objektorientierte Paradigma

Beispiel

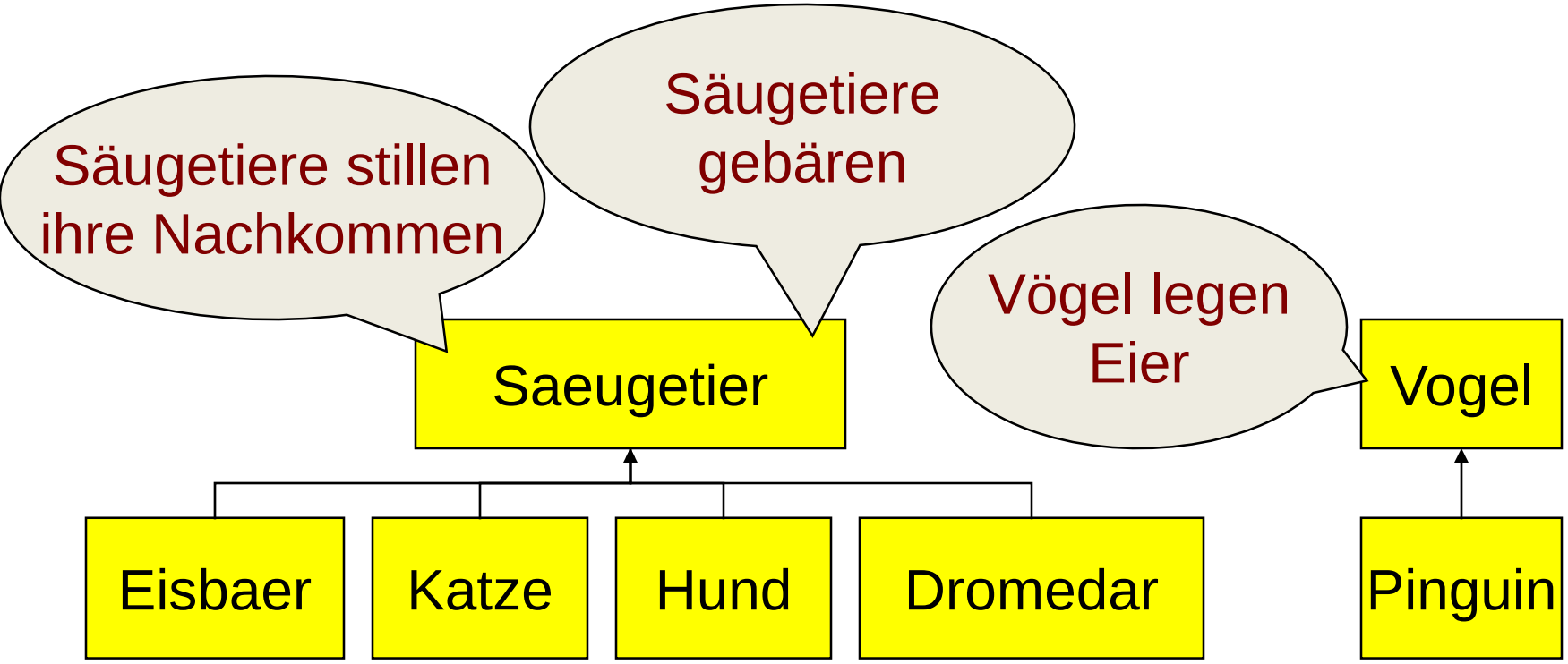
Bei der hier angegebenen Abstraktion (sie wird auch als **Generalisierung** bezeichnet), handelt es sich um eine „ist-ein“-Beziehung



Das objektorientierte Paradigma

Beispiel

Sämtliche Tiere, für die diese „ist-ein“-Beziehung gilt, haben gewisse Eigenschaften gemeinsam

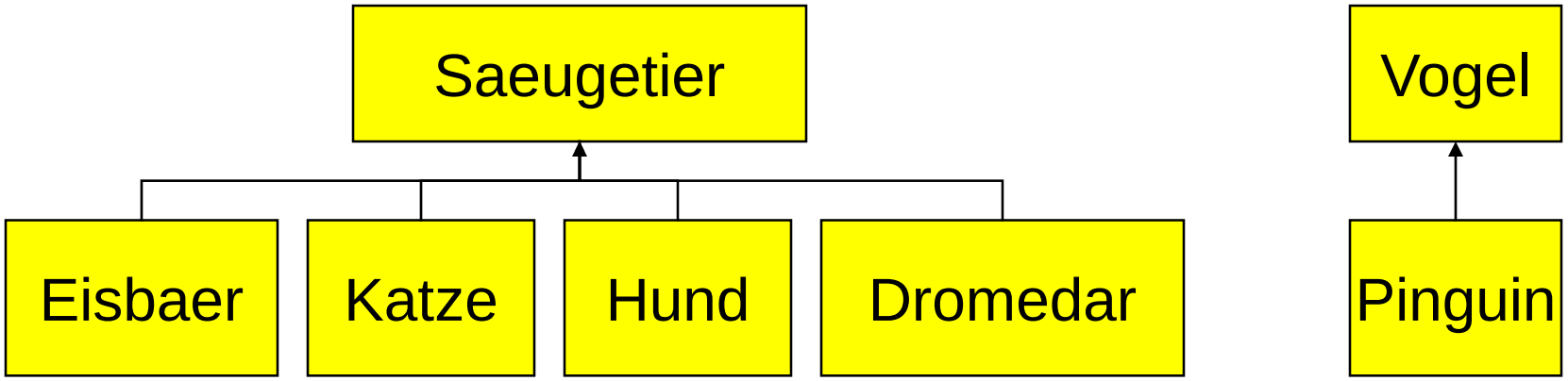


Das objektorientierte Paradigma

Beispiel

Sämtliche Tiere, für die diese „ist-ein“-Beziehung gilt, haben gewisse Eigenschaften gemeinsam

Anstatt also etwa den Vorgang des „Milch gebens“ für Eisbär, Katze, Hund und Dromedar einzeln zu modellieren, wird der geschickte Programmierer dies lediglich für die **Superklasse** „Saeugetier“ tun

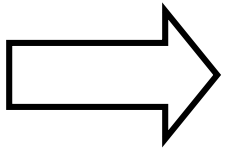


Das objektorientierte Paradigma

Sämtliche Tiere, für die diese „ist-ein“-Beziehung gilt, haben gewisse Eigenschaften gemeinsam

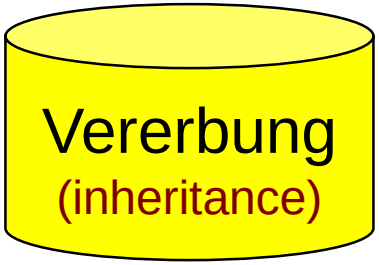
Anstatt also etwa den Vorgang des „Milch gebens“ für Eisbär, Katze, Hund und Dromedar einzeln zu modellieren, wird der geschickte Programmierer dies lediglich für die **Superklasse** „Saeugetier“ tun

Hierdurch erhalten auch alle **Subklassen** (das sind die Klassen, die zur Superklasse in einer „ist-ein“-Beziehung stehen) automatisch die Eigenschaft, Milch geben zu können



Wir bezeichnen diesen Vorgang als ***Vererbung***

Das objektorientierte Paradigma



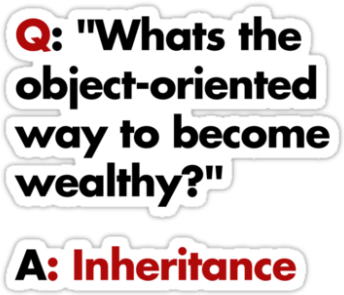
ist das Weiterreichen von Eigenschaften zwischen Klassen, die in einer „ist-ein“-Beziehung stehen

Ziel der **Vererbung** ist es

- gemeinsame Eigenschaften verwandter Klassen einheitlich zu modellieren,
- möglichst viel Programmcode wiederverwenden zu können und somit
- Entwicklungszeit zu sparen und
- eine Vielzahl von Fehlerquellen auszuschalten

Das objektorientierte Paradigma

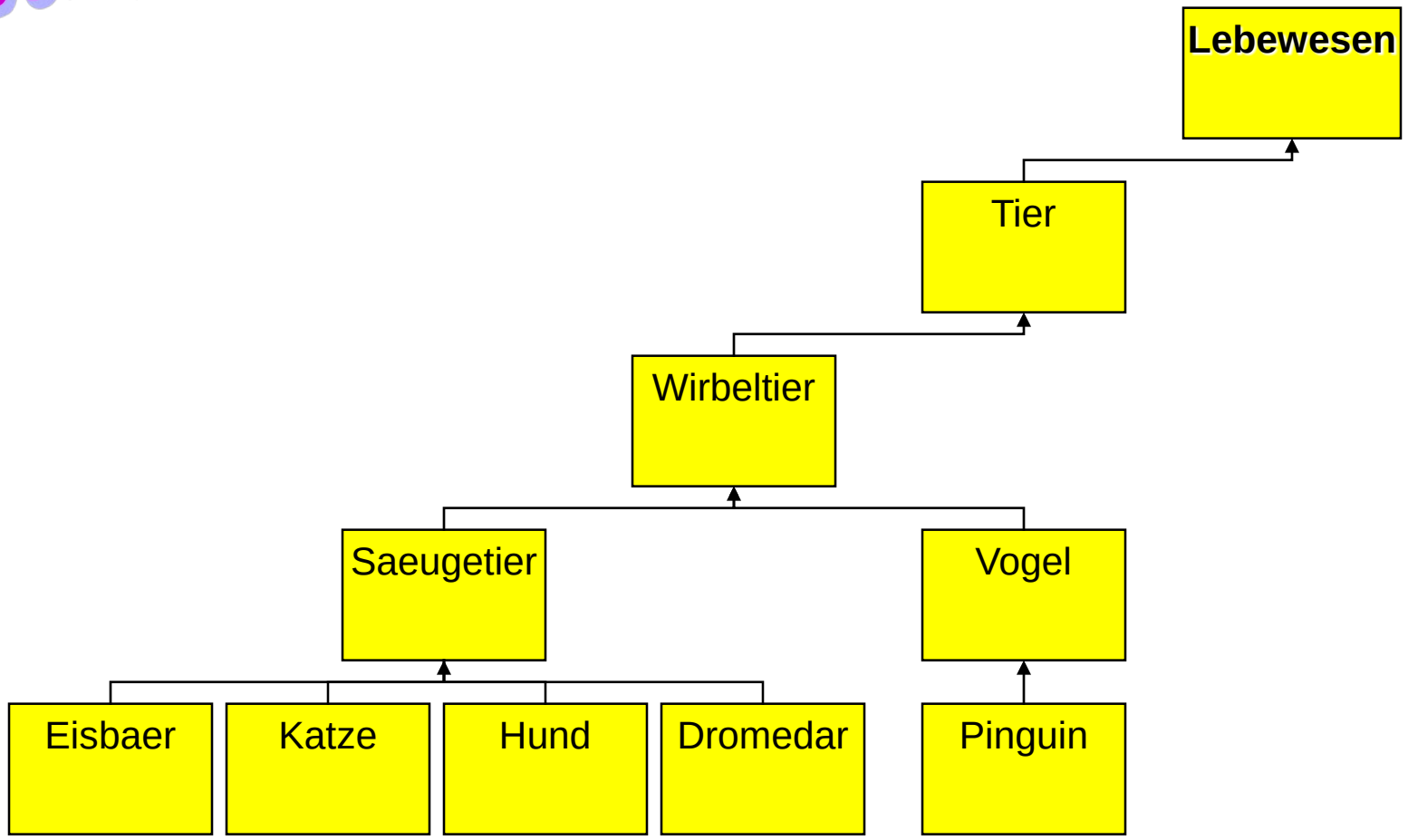
Betrachten wir eine „erweiterte Version“ unserer Tierhierarchie:



Das objektorientierte Paradigma

Beispiel

Jede Klasse in diesem Schaubild „ist ein“ Lebewesen und soll deshalb mit diesem gewisse Eigenschaften teilen

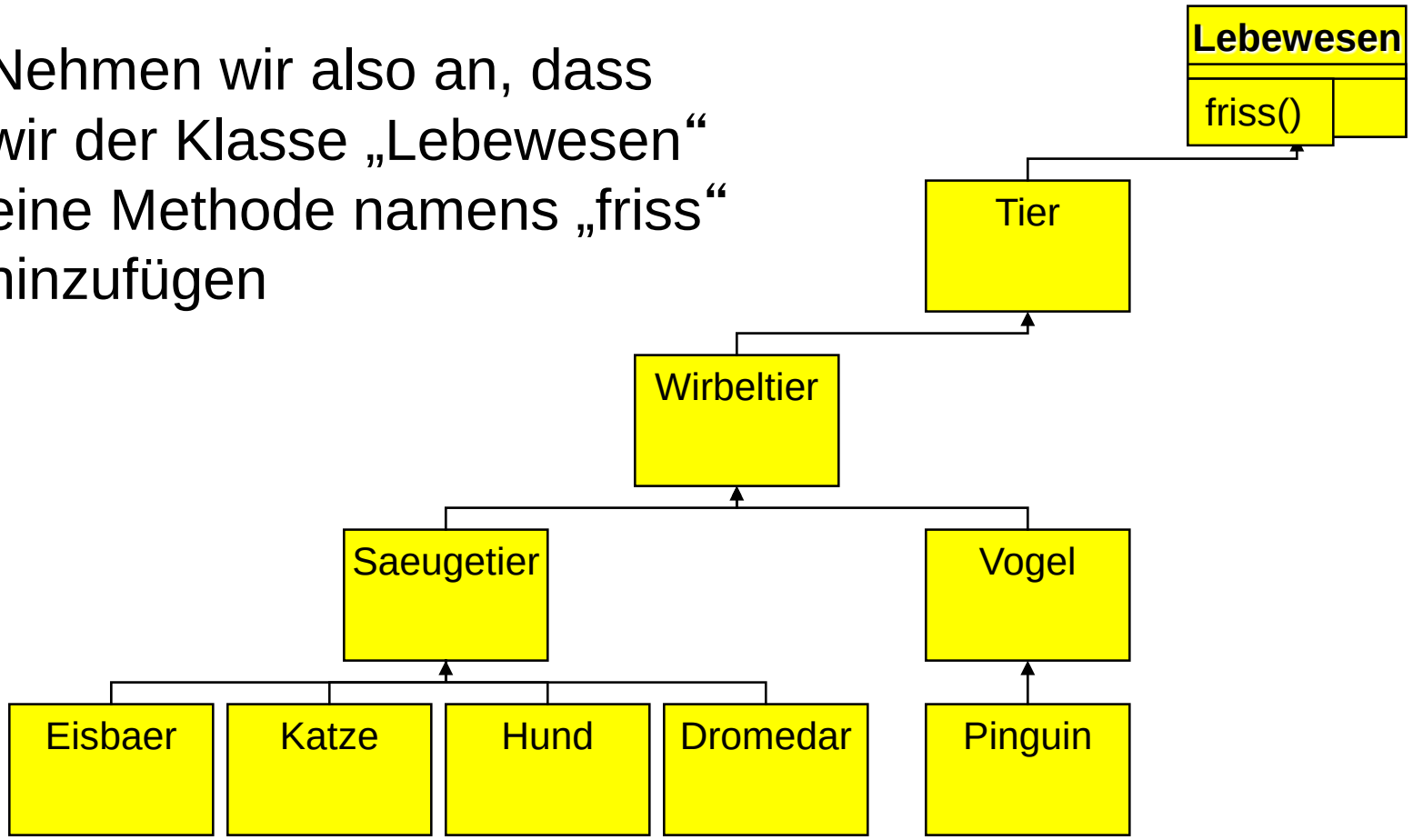


Das objektorientierte Paradigma

Beispiel

Jedes Lebewesen muss beispielsweise essen

Nehmen wir also an, dass wir der Klasse „Lebewesen“ eine Methode namens „friss“ hinzufügen

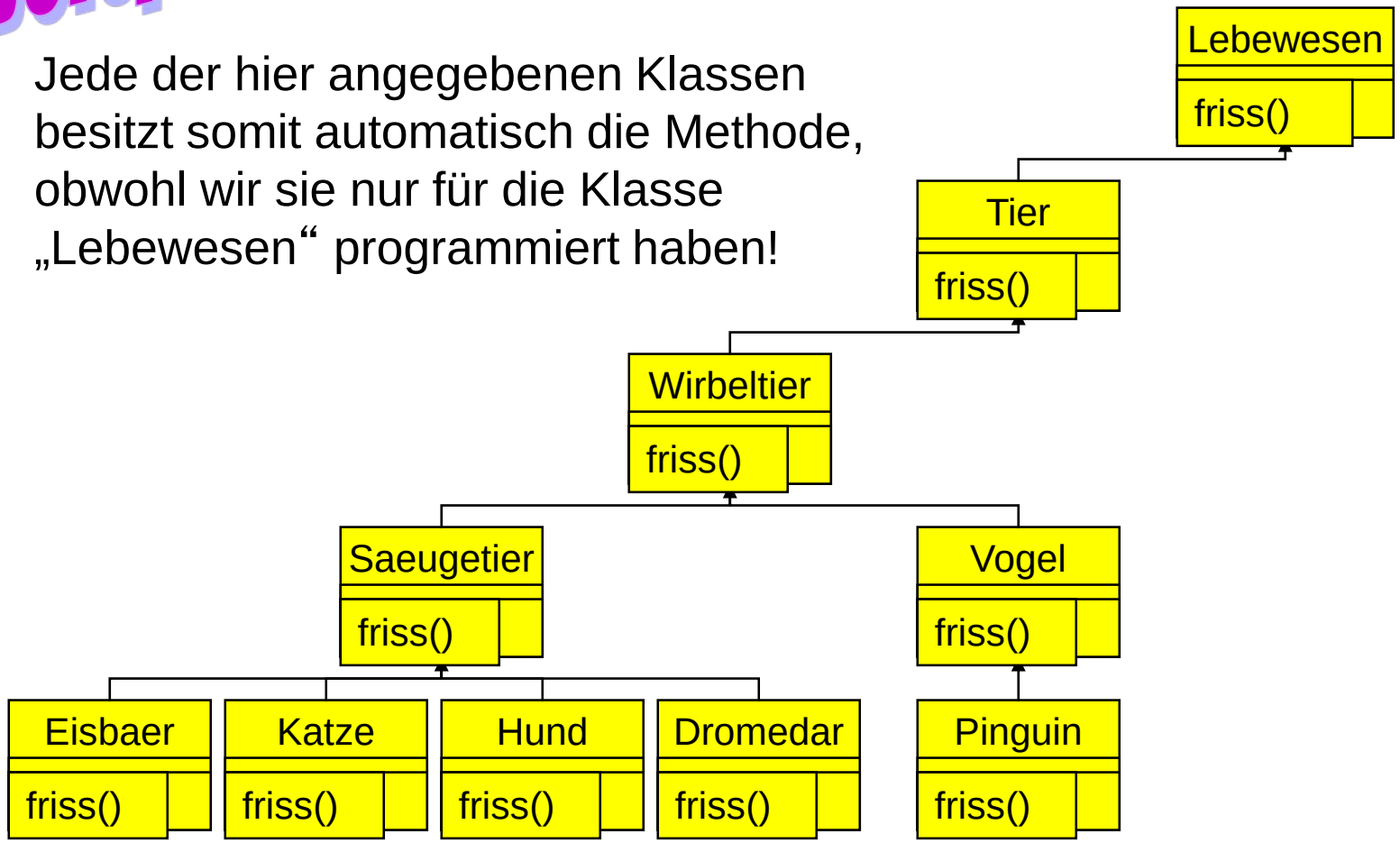


Das objektorientierte Paradigma

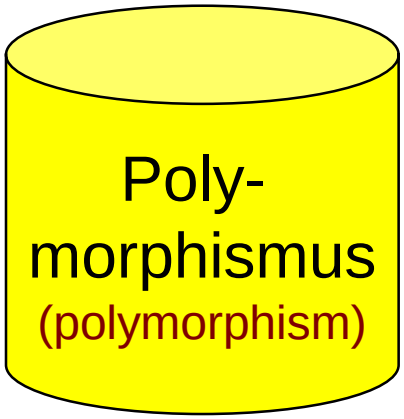
Beispiel

Dann **vererbt** die Klasse diese Methode **automatisch** an alle ihre Subklassen

Jede der hier angegebenen Klassen besitzt somit automatisch die Methode, obwohl wir sie nur für die Klasse „Lebewesen“ programmiert haben!



Das objektorientierte Paradigma



ist die Möglichkeit, Methoden innerhalb der Objekthierarchie zu **überschreiben** (override) und somit individuelles Verhalten verschiedener Klassen auf dieselbe Anweisung zu modellieren

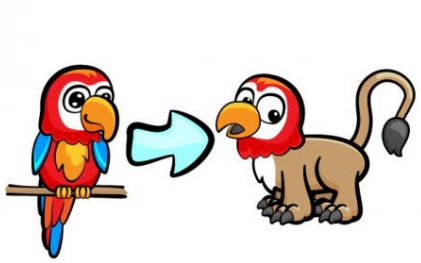
Mit Hilfe des **Polymorphismus** kann der Entwickler

- ohne Probleme Besonderheiten bei den verschiedenen Subklassen berücksichtigen,
- einheitliche Schnittstellen erzeugen und somit
- Entwicklungszeit sparen und
- eine Vielzahl von Fehlerquellen ausschalten

Das objektorientierte Paradigma

Beispiel

Werfen wir erneut einen Blick auf unser Tiermodell:

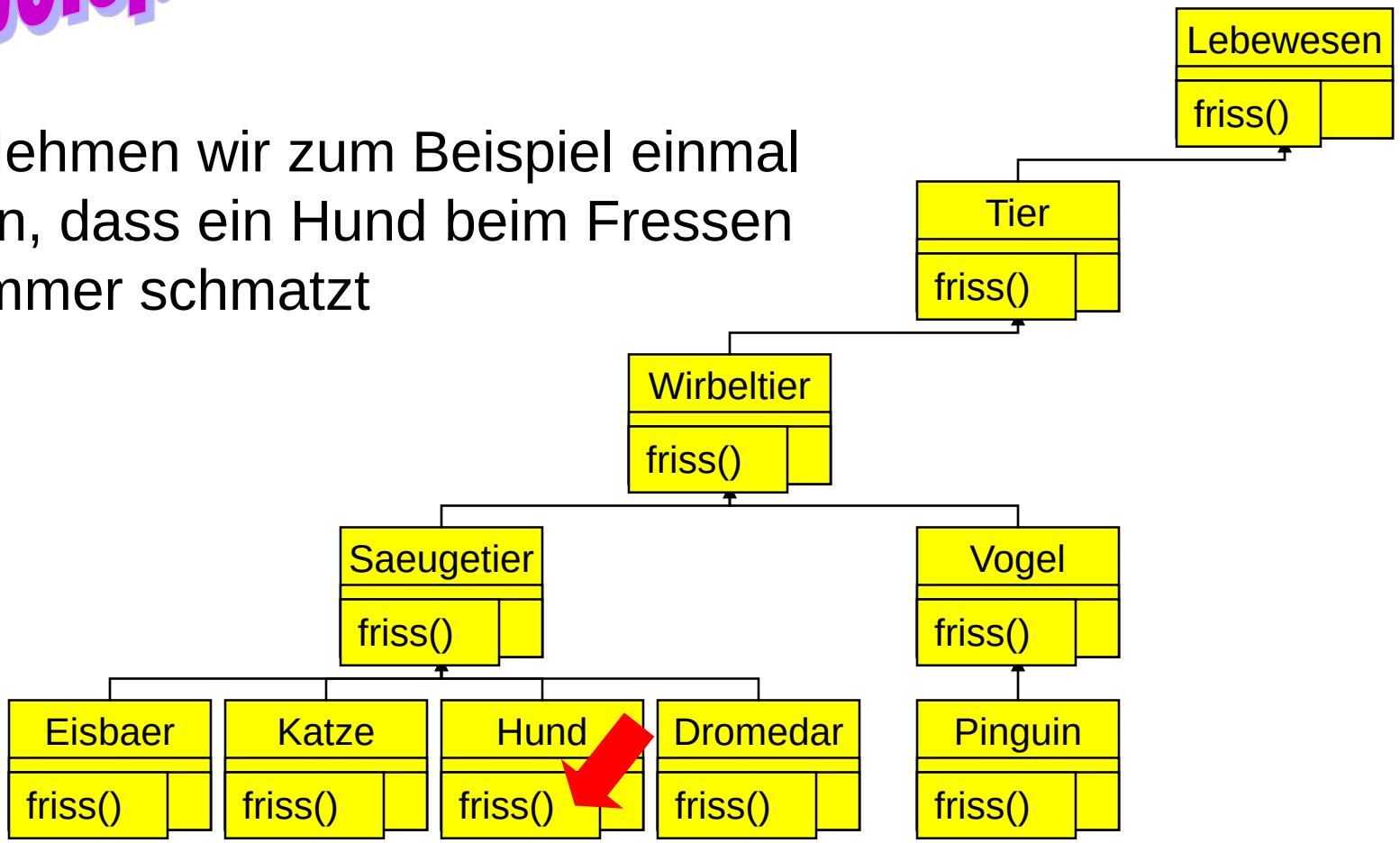


Das objektorientierte Paradigma

Beispiel

Nicht jedes Lebewesen frisst in unserer Simulation auf dieselbe Weise

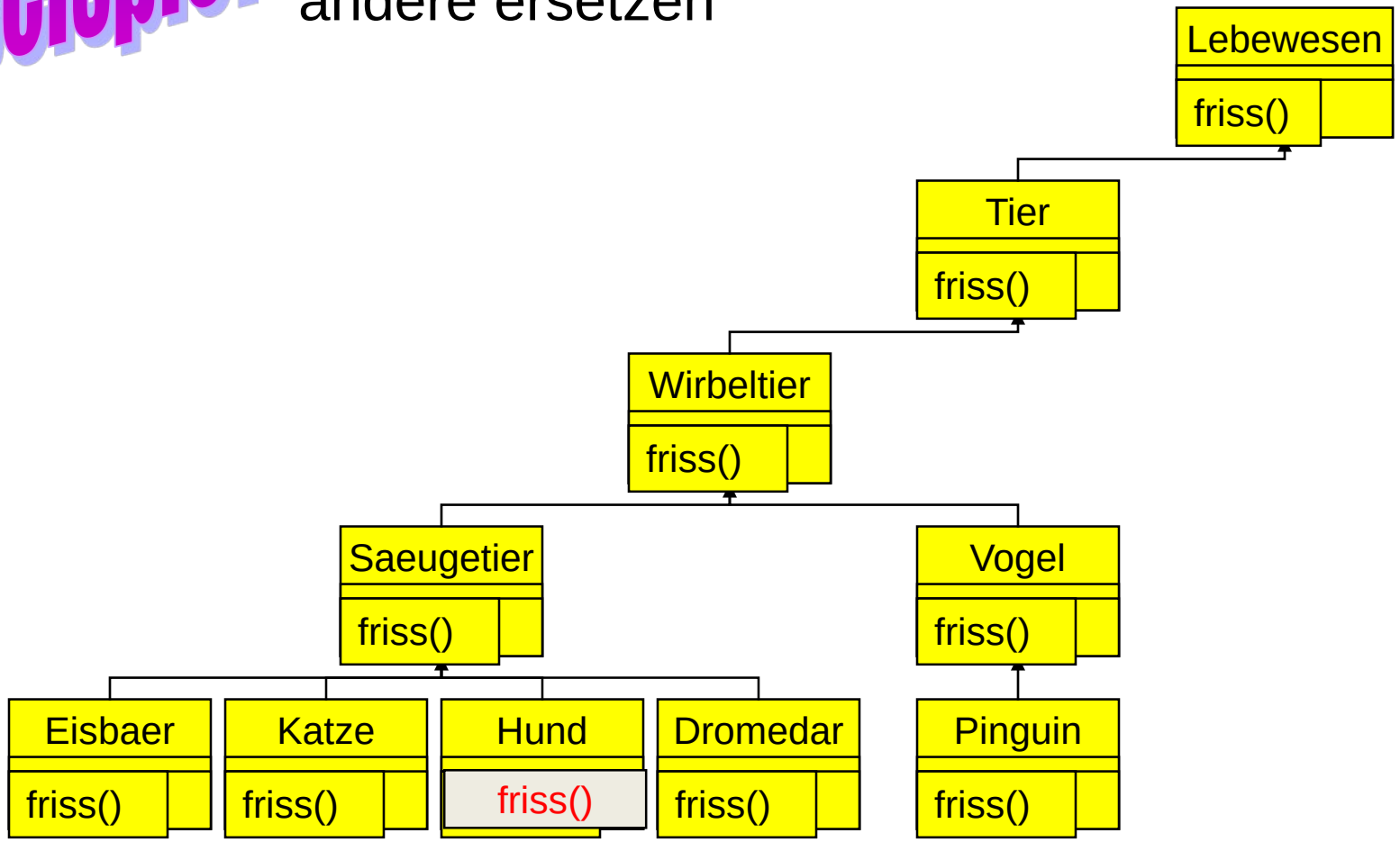
Nehmen wir zum Beispiel einmal an, dass ein Hund beim Fressen immer schmatzt



Das objektorientierte Paradigma

Beispiel

In dieser Situation können wir die alte Methode „friss“ für die Klasse Hund einfach durch eine andere ersetzen

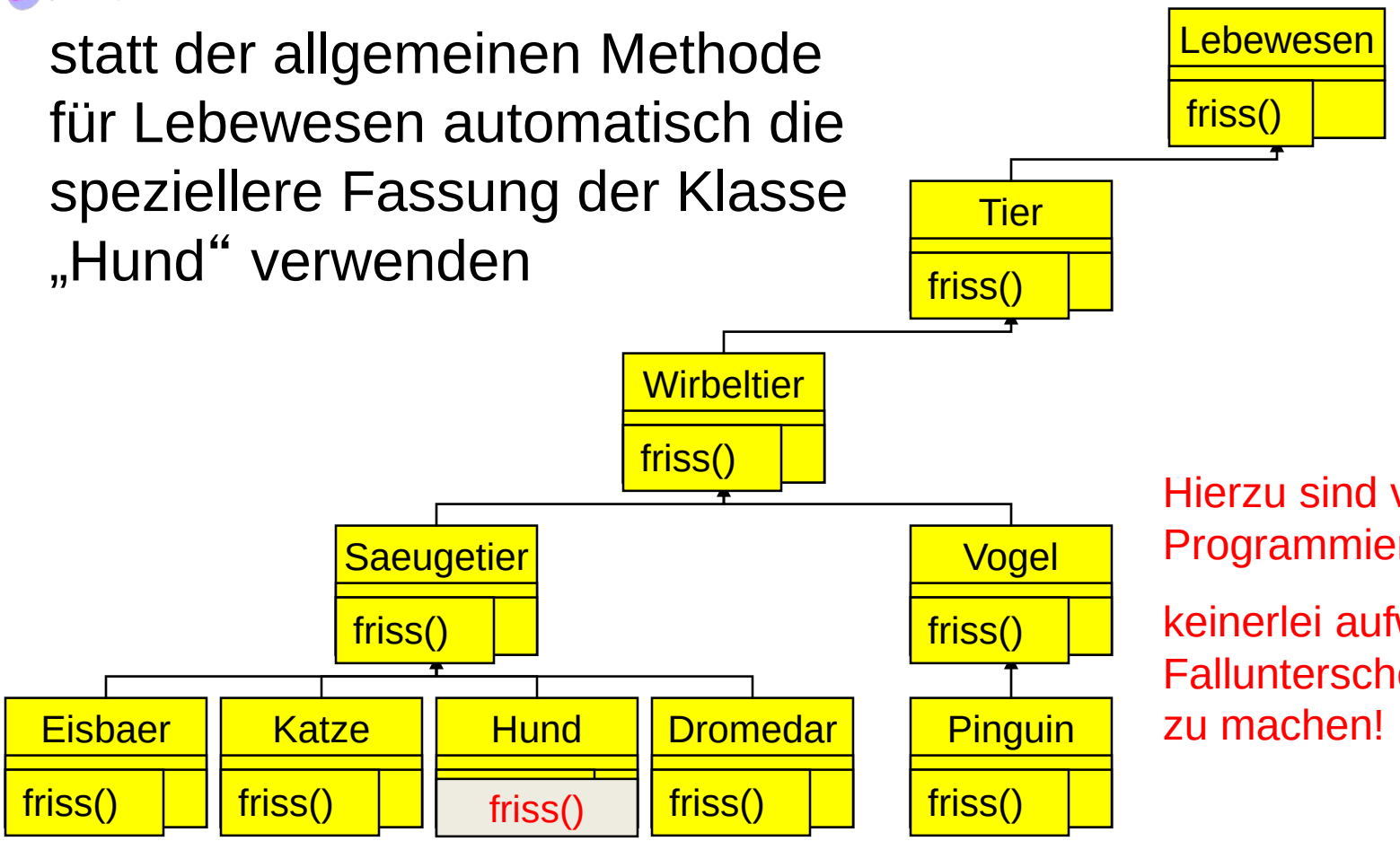


Das objektorientierte Paradigma

Beispiel

Wenn wir nun für ein Lebewesen der Gattung „Hund“ die Methode „friss“ aufrufen, wird das System

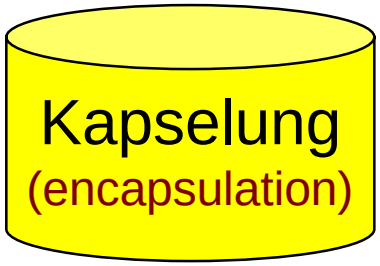
statt der allgemeinen Methode für Lebewesen automatisch die speziellere Fassung der Klasse „Hund“ verwenden



Hierzu sind vom Programmierer

keinerlei aufwändige Fallunterscheidungen zu machen!

Das objektorientierte Paradigma



ist die Philosophie, **Daten und Methoden** fest an ihr zugehöriges Objekt zu **binden**

Hierbei werden Daten dem Benutzer normalerweise nicht direkt zugänglich gemacht, sondern vor direkten Zugriffen von außen geschützt (**information hiding**). Das gesamte Verhalten eines Objekts wird nur durch seine Methoden bestimmt.

Durch diese Trennung von **Schnittstelle** (der nach außen sichtbaren Methoden) und der **internen Realisierung** werden Programmierer vor Problemen bewahrt, die sich bei einer internen Umstrukturierung der Klasse ergeben.

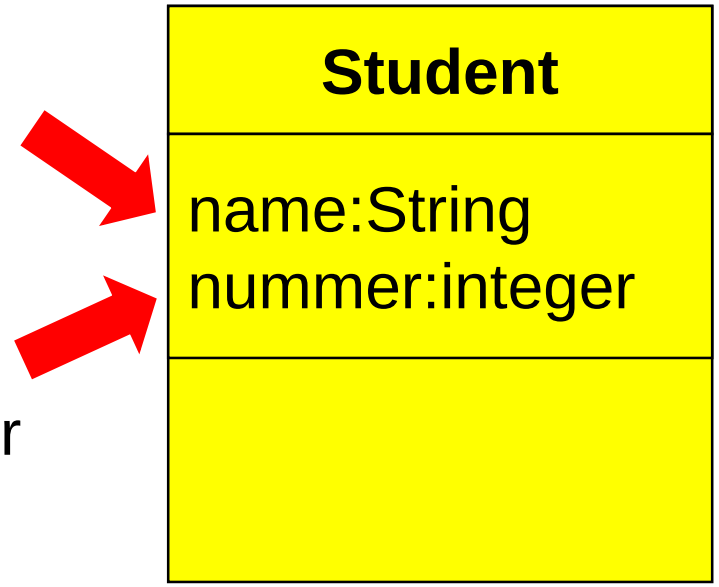
Das objektorientierte Paradigma

Beispiel

Wir wollen eine Klasse „Student“ entwerfen, die Namen und Matrikelnummer eines Studenten abspeichern kann

Hierzu spendieren wir der Klasse zwei Variablen:

- in einem String namens „name“ wollen wir den Namen des Studenten ablegen und
- in einer Integer-Variable namens „nummer“ soll die Matrikelnummer gespeichert werden



Das objektorientierte Paradigma

Beispiel

Wenn wir nun aus unserem „Bauplan“ verschiedene **Instanzen** erzeugen, können wir für jede Instanz die Variablen individuell belegen

Peter:Student
name=„Peter Petersen“
nummer=0978532

Lena:Student
name=„Lena Lustig“
nummer=0875329

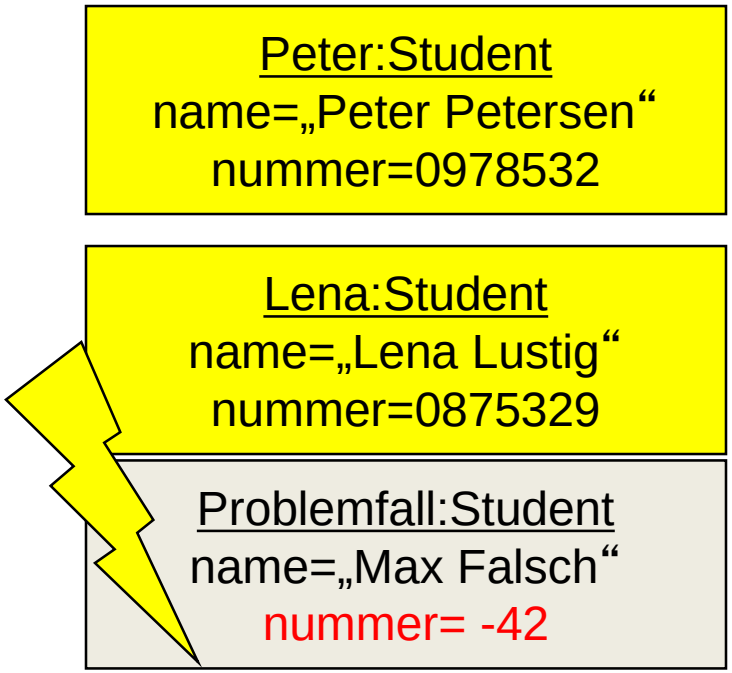
Student

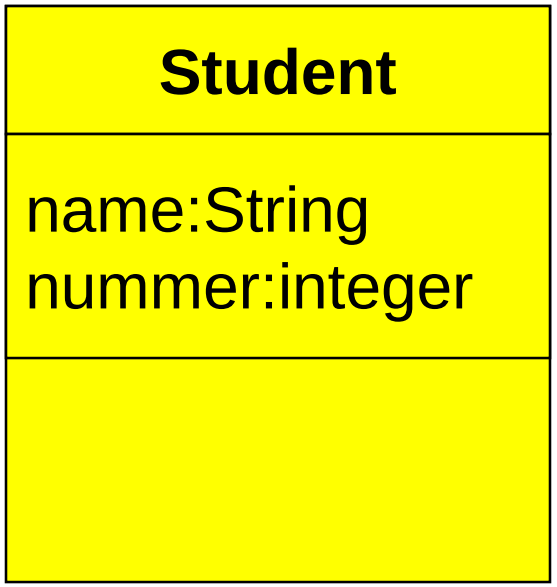
name:String
nummer:integer

Das objektorientierte Paradigma

Beispiel

Hierbei können wir jedoch nicht verhindern, dass etwa bei der Matrikelnummer eine fehlerhafte Zuweisung erfolgt



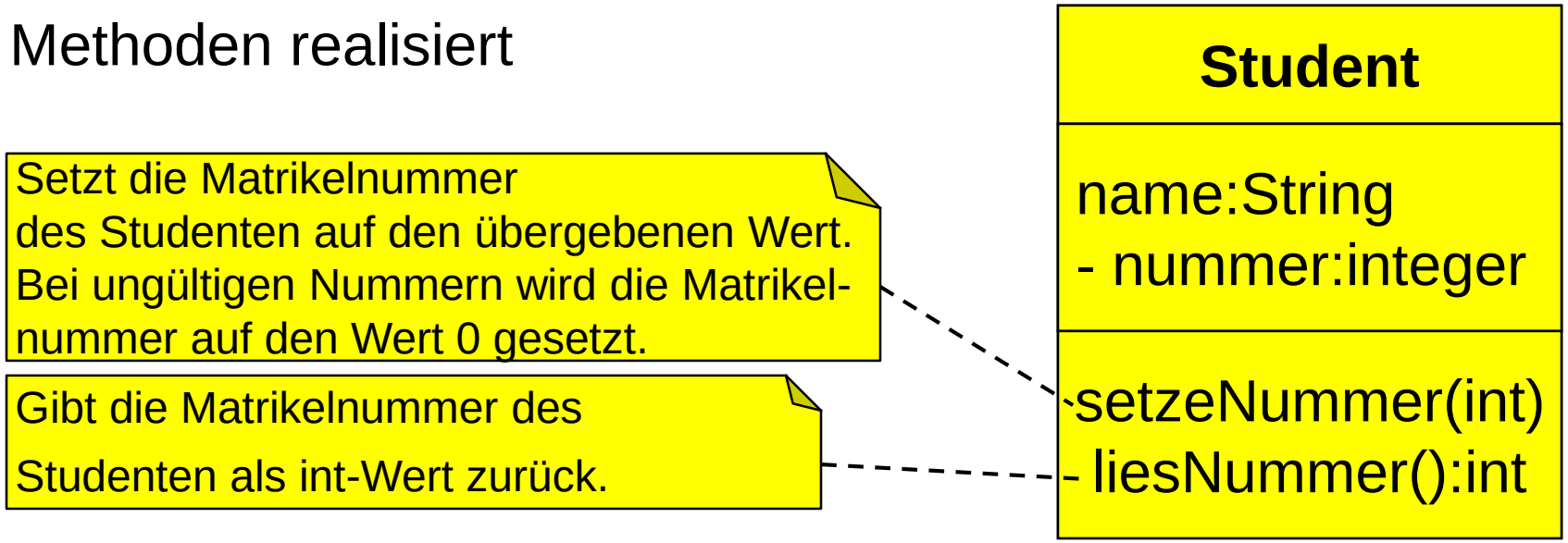


Das objektorientierte Paradigma

Beispiel

Um diese Probleme zu vermeiden, wird man die internen Datenstrukturen einer Klasse vor den Nutzern üblicherweise verbergen - in diesem Fall also etwa die Variable „nummer“

Zugriffe auf die entsprechenden Daten werden stattdessen über Methoden realisiert



Das objektorientierte Paradigma

Diesen Prozess bezeichnen wir üblicherweise als **information / data hiding**, das heißt als „Verstecken“ der Daten vor dem Benutzer

Student
name:String - nummer:integer
setzeNummer(int) liesNummer():int

Das objektorientierte Paradigma

Diesen Prozess bezeichnen wir üblicherweise als **information / data hiding**, das heißt als „Verstecken“ der Daten vor dem Benutzer

Das Binden von Daten und Methoden an eine spezielle Klasse erhebt die Objekte somit über ein reines Konstrukt zum Speichern von Daten hinaus.

Der Programmierer ist schon beim Entwurf seiner „Datenspeicher“ in der Lage, das spezifische Verhalten seiner Objekte (etwa auf Datenfehler) zu modellieren.

Student
name:String - nummer:integer
setzeNummer(int) liesNummer():int

Fragen

“Error, no keyboard - press F1 to continue”

-- early PC error message (BIOS)

Danksagung

- Vorlesungsmaterialien von Prof. Dr. Detlef Seese wurden als Basis verwendet
- Unterstützung bei der technischen und inhaltlichen Gestaltung des Vorlesungsmaterials leisteten:

  Jóakim v. Kistowski

  Dietmar Ratz, Joachim Melcher, Roland Küstermann, Jana Weiner, Hagen Buchwald, Matthes Elstermann, Oliver Schöll, Niklas Kühl, Tobias Diederich