Einführung in die Objektorientierte Programmierung
Vorlesung 2: Mikroskopisches und Makroskopisches Computational Thinking

Sebastian Küpper

Das makroskopische Modell: Objekte

- Objekt: Abbild konkreter individuell unterscheidbarer Gegenstände oder Träger von Rollen
- Im Fallbeispiel: Die Kundin Anna Müller oder die Rechnung mit der Nummer 20022
- Abstraktion auf funktionielle Einheit dient der Strukturierung von Programmen

Das makroskopische Modell: Objekte

- Objekt: Abbild konkreter individuell unterscheidbarer Gegenstände oder Träger von Rollen
- Im Fallbeispiel: Die Kundin Anna Müller oder die Rechnung mit der Nummer 20022
- Abstraktion auf funktionielle Einheit dient der Strukturierung von Programmen
- Klasse: Bauplan für gleichartige Objekte
- Im Fallbeispiel: Kunde oder Rechnung
- Wesentliche Codeelemente in der OOP

Bestandteile einer Klasse

- Attribute
- Im Fallbeispiel: Name des Kunden oder Rechnungsnummer der Rechnung

Bestandteile einer Klasse

- Attribute
- Im Fallbeispiel: Name des Kunden oder Rechnungsnummer der Rechnung
- Methode
- Beschreibt Verhalten von Objekten; ermöglicht die Veränderung von Attributwerten.
- Im Fallbeispiel: `markiereAlsBezahlt()` um Rechnung als bezahlt zu markieren.

Zustand eines Objekts

- Attributwert: Konkreter Wert den Attribut in einem Objekt annimmt
- Zustand eines Objekts: Gesamtheit der Attributwerte

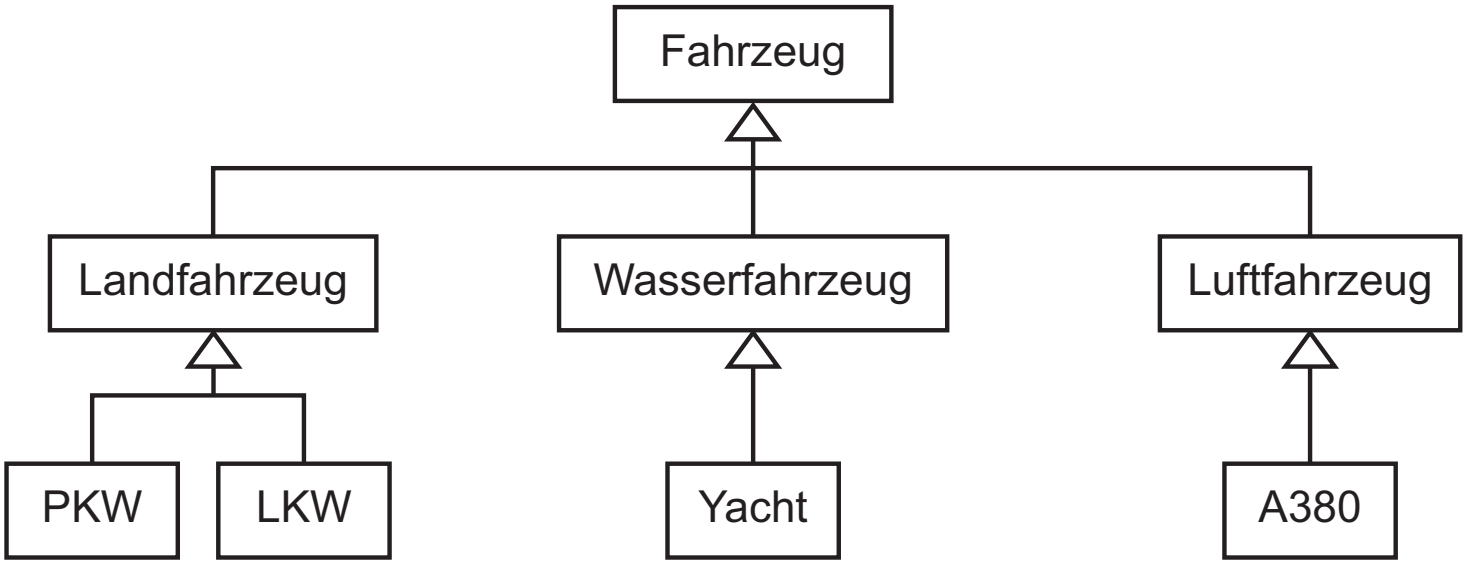
Verknüpfungen von Objekten

- Zwei Klassen können mit einer (benannten) Assoziation verbunden sein
- Die konkrete Verbindung zweier Objekte durch eine Assoziation heißt Verknüpfung

Verknüpfungen von Objekten

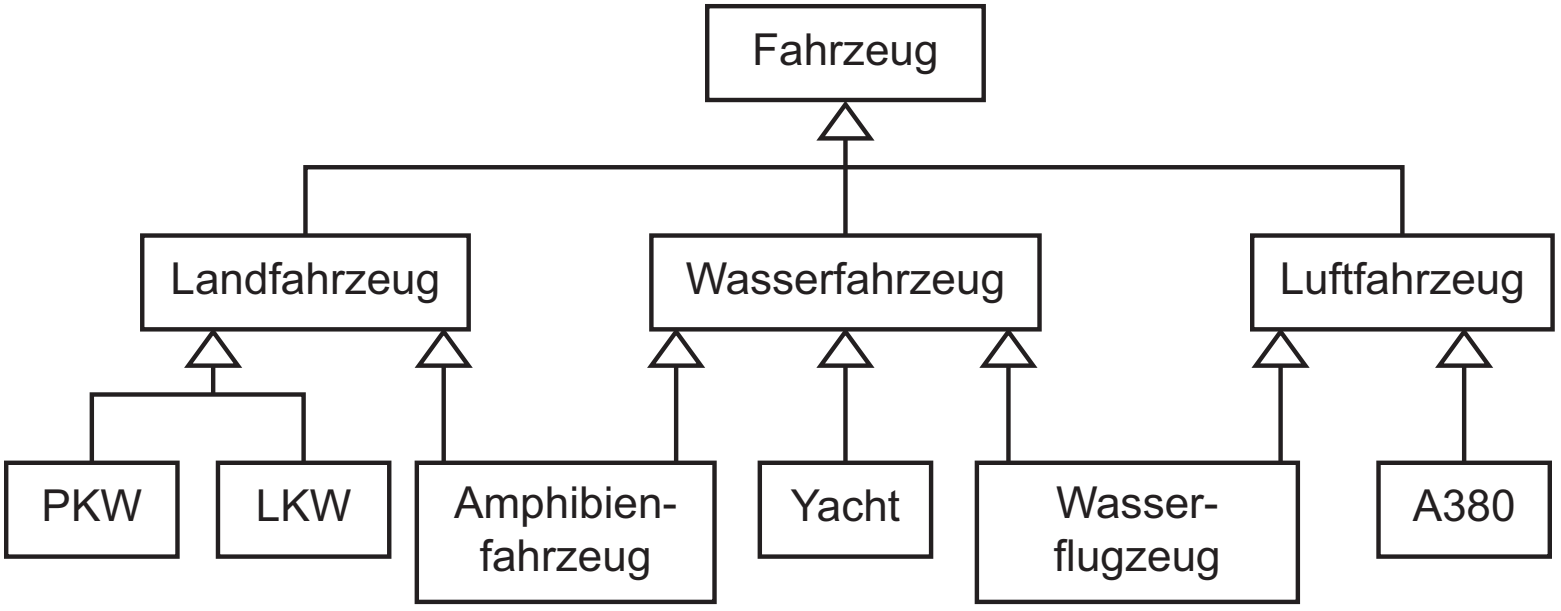
- Zwei Klassen können mit einer (benannten) Assoziation verbunden sein
- Die konkrete Verbindung zweier Objekte durch eine Assoziation heißt Verknüpfung
- Beispiel: Jede Rechnung ist für einen Kunden (Assoziation `rechnungsempfaenger`)
- Betrachten wir die Rechnung 20022 mit `rechnungsempfaenger` Jens Meier, so betrachten wir eine Verknüpfung dieser Rechnung mit dem Kunden Jens Meier

Spezialisierung / Generalisierung



Bei Spezialisierung bleiben Beziehungen, Eigenschaften und Verhalten erhalten, können aber erweitert oder modifiziert werden.

Mehrfachvererbung



In Java im Wesentlichen nicht erlaubt.

Kommunikation zwischen Objekten: Nachrichten

- Alternativer Terminus: Methodenaufruf
- Nachricht besteht aus:
  - Methode, die ausgeführt werden soll
  - ggf. Parametern, die für die Methode verwendet werden sollen

Kommunikation zwischen Objekten: Nachrichten

- Alternativer Terminus: Methodenaufruf
- Nachricht besteht aus:
  - Methode, die ausgeführt werden soll
  - ggf. Parametern, die für die Methode verwendet werden sollen
- Beispiel: E-Mail Client sendet Nachricht `SendeEMail` an Server, Parameter sind die Empfängeradresse, der Betreff und der Nachrichtentext

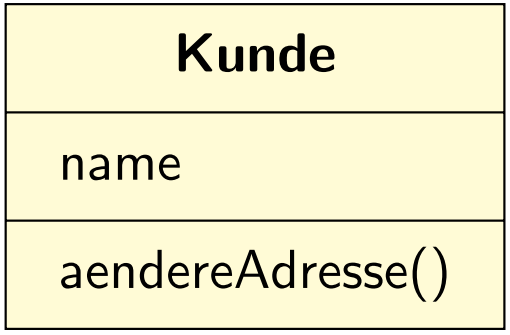
Grundlegende Fragen des objektorientierten Entwurfs

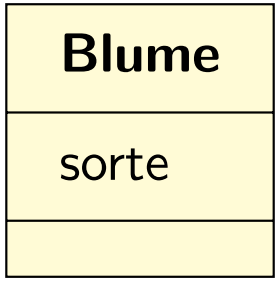
- Welche Gegenstände werden bearbeitet, verändert oder kommunizieren?
- Welche Eigenschaften haben diese Gegenstände?
- Welche Beziehungen herrschen zwischen den Gegenständen?
- Wie wird mit ihnen umgegangen?
- Welche Rollen gibt es und für welche Handlungen sind sie verantwortlich?

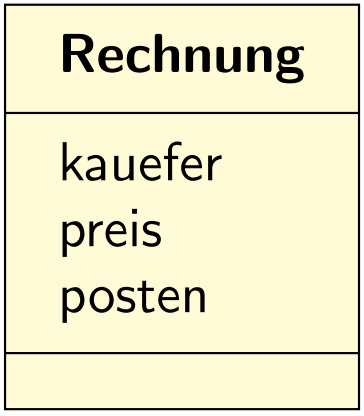
UML-Klassen- und Objektdiagramm

- Analyseergebnis führt zu Modell auf zwei Ebenen: Klassen und Objekte
- UML (Unified Modelling Language) bietet eine visuelle Beschreibungssprache hierfür
- Klassendiagramme stellen die statische Struktur des Programms dar, Objektdiagramme die handelnden Instanzen zur Laufzeit

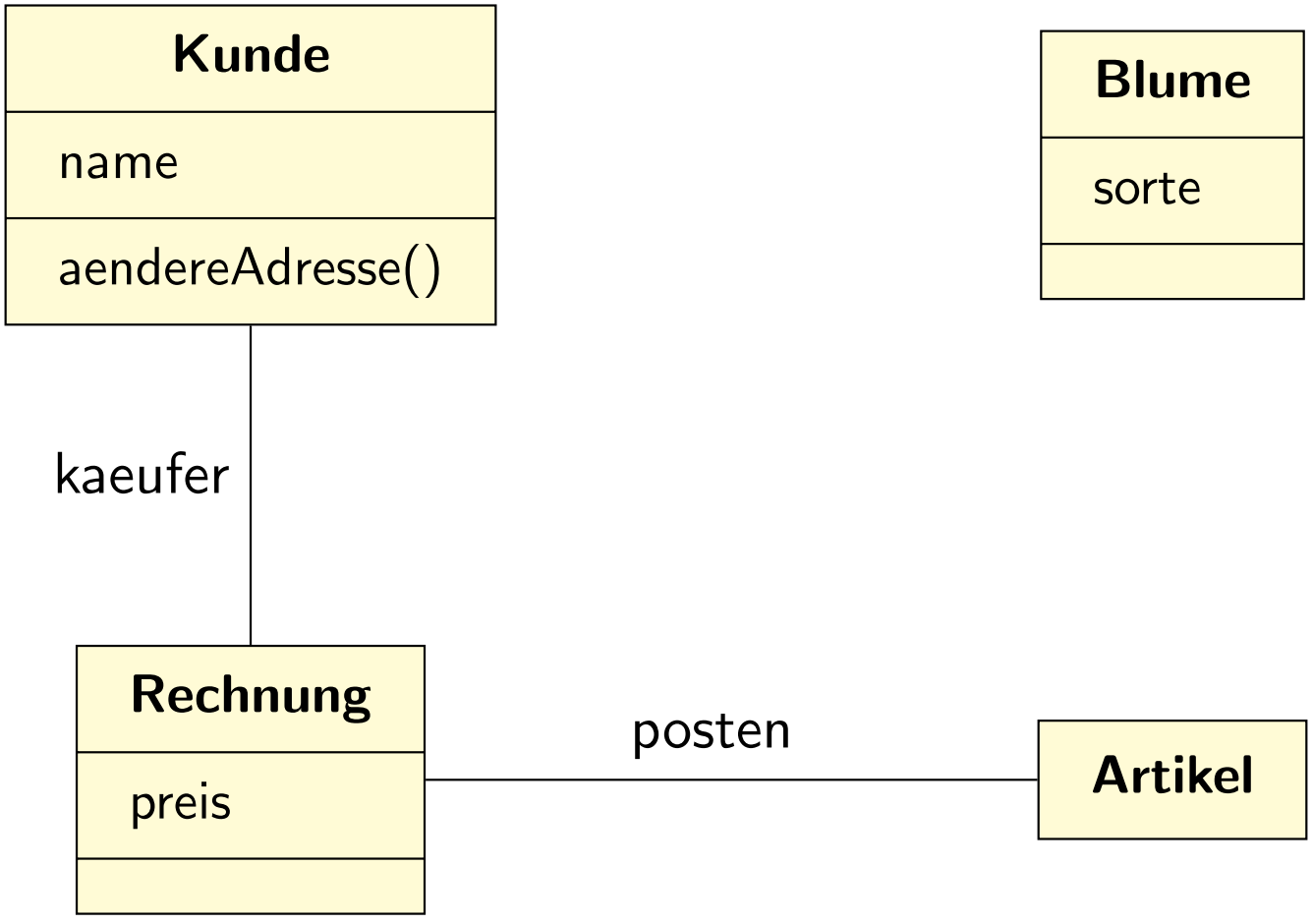
Klassen in UML



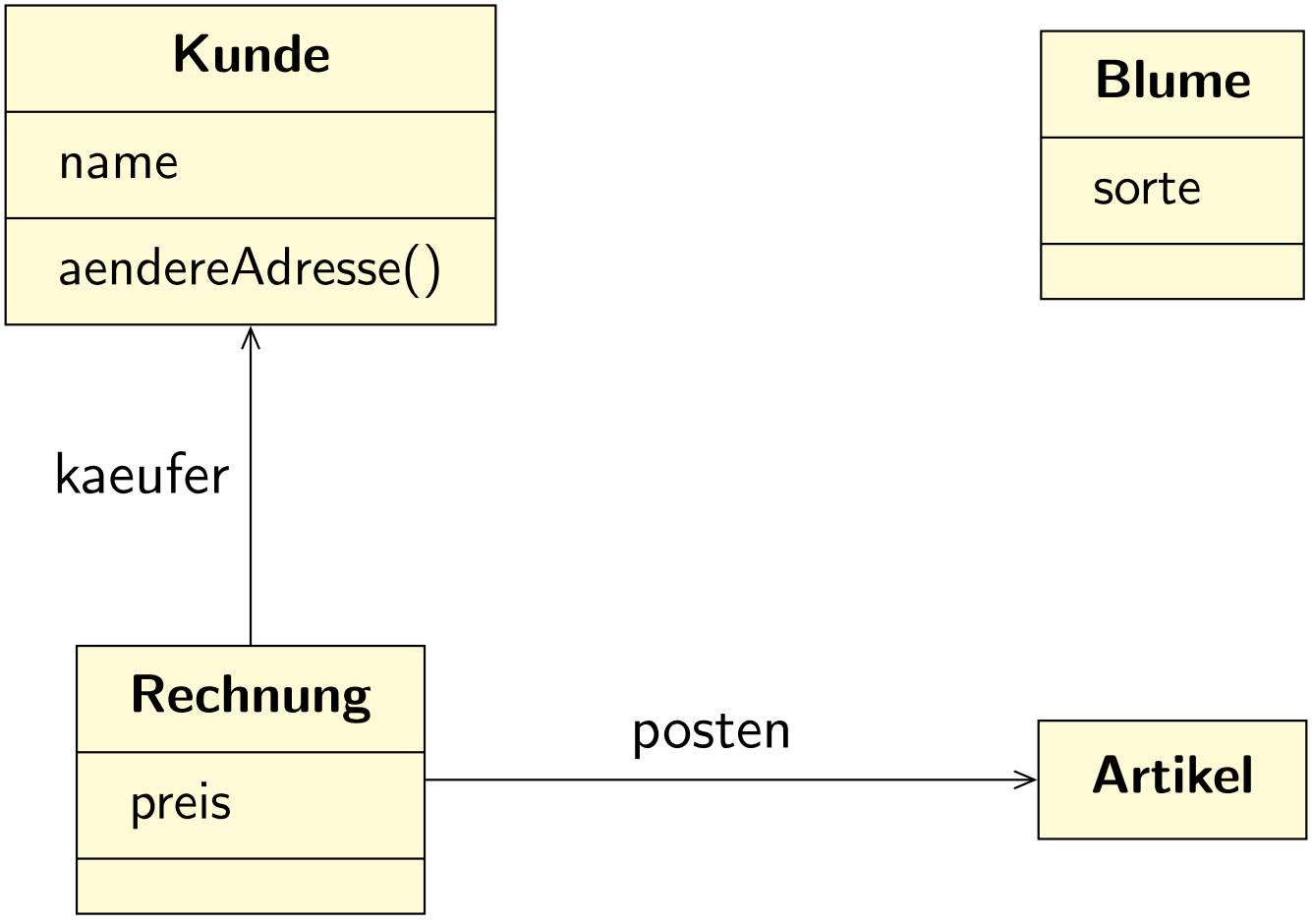




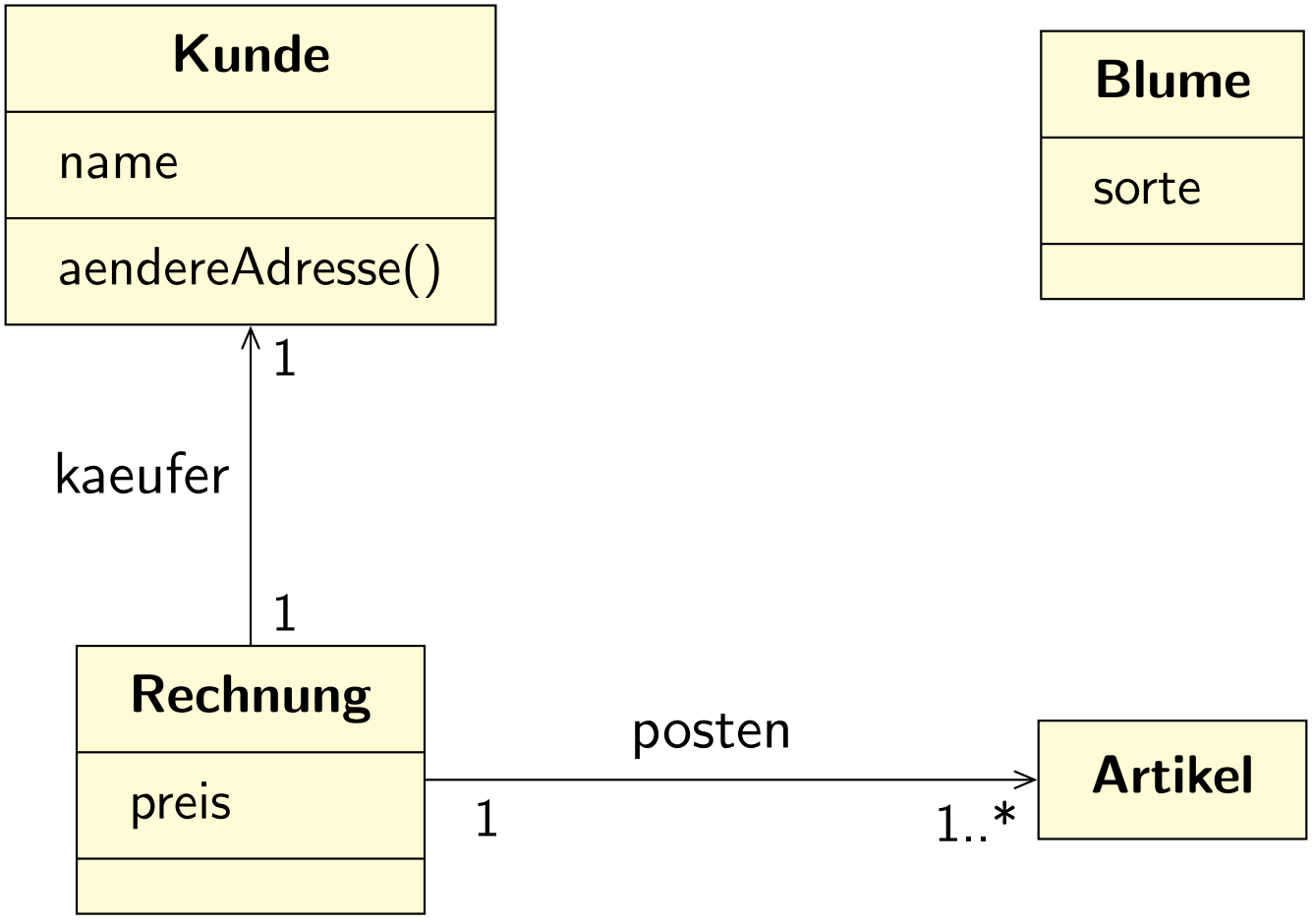
Assoziationen in UML



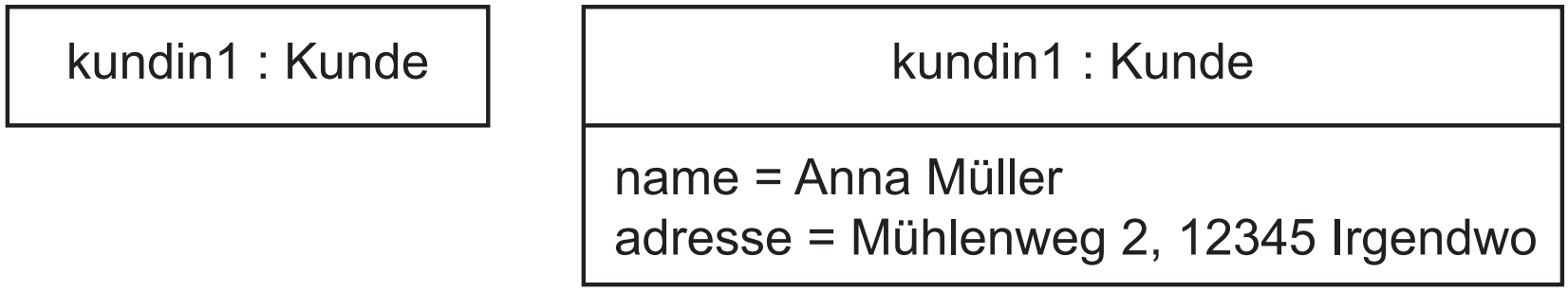
Gerichtete Assoziationen in UML



Assoziationen mit Multiplizitäten in UML



Objektdiagramme in UML



Assoziationen wie in Klassendiagrammen

Wechsel zur Mikroskopischen Sicht: Wie arbeiten Methoden?

- Bislang: Makroskopische Sicht auf Programm
- Fokus auf Struktur und Akteure
- Noch nicht beachtet: Wie arbeiten die Methoden?
- Die mikroskopische Sicht bieten Algorithmen (Rechenvorschriften)

Probleme

- Problemklasse vs. Problem: Abstrakte Aufgabe vs. konkrete Aufgabe
- Beispiel: Sortiere eine Zahlenfolge (Problemklasse) vs. Sortiere die Zahlenfolge $6, 32, 1, 64, 12$
- Achtung: Weicht ab von Definition in der Theoretischen Informatik
- in der TI: Problem: Klasse von Aufgaben, Probleminstanz: Konkrete Aufgabe. Zusätzliche Einschränkung: Ja / Nein Aufgabe

Algorithmus

Definition

Ein Algorithmus ist eine wohldefinierte Verfahrensbeschreibung, die aus einem oder mehreren Eingabewerten einen oder mehrere Ausgabewerte mit bestimmten Eigenschaften produziert. Ein Algorithmus löst dabei immer eine Klasse von Problemen.

Eigenschaften von Algorithmen

- Terminierung
- Finitheit (der Beschreibung)
- Effektivität

In unserem Kontext stets gefordert;

Eigenschaften von Algorithmen

- Terminierung
- Finitheit (der Beschreibung)
- Effektivität

In unserem Kontext stets gefordert; weiterhin von uns oft gefordert:

- Determiniertheit (des Ergebnisses)
- Determinismus (der Ausführung)

Determiniertheit ist für Effektivität oft notwendig.

Eigenschaften von Algorithmen

- Terminierung
- Finitheit (der Beschreibung)
- Effektivität

In unserem Kontext stets gefordert; weiterhin von uns oft gefordert:

- Determiniertheit (des Ergebnisses)
- Determinismus (der Ausführung)

Determiniertheit ist für Effektivität oft notwendig.
Üblicherweise wünschenswert:

- Effizienz

Beispiel: Euklids Algorithmus für ggT

1. Wenn b $>$ a ist, vertausche a und b.
2. Sei c der Rest der Division von a durch b.
3. Wenn c $= 0$ ist, dann ist der Algorithmus zu Ende und b das Ergebnis.
4. a wird der Wert von b zugewiesen und b der Wert von c.
5. Fahre mit Schritt 2 fort.

Beispiel: Euklids Algorithmus für ggT

1. Wenn b $>$ a ist, vertausche a und b.
2. Sei c der Rest der Division von a durch b.
3. Wenn c $= 0$ ist, dann ist der Algorithmus zu Ende und b das Ergebnis.
4. a wird der Wert von b zugewiesen und b der Wert von c.
5. Fahre mit Schritt 2 fort.

Berechne ggT von 22 und 12:

a	b	c
22	12	10

Beispiel: Euklids Algorithmus für ggT

1. Wenn b $>$ a ist, vertausche a und b.
2. Sei c der Rest der Division von a durch b.
3. Wenn c $= 0$ ist, dann ist der Algorithmus zu Ende und b das Ergebnis.
4. a wird der Wert von b zugewiesen und b der Wert von c.
5. Fahre mit Schritt 2 fort.

Berechne ggT von 22 und 12:

a	b	c
12	10	2

Beispiel: Euklids Algorithmus für ggT

1. Wenn b $>$ a ist, vertausche a und b.
2. Sei c der Rest der Division von a durch b.
3. Wenn c $= 0$ ist, dann ist der Algorithmus zu Ende und b das Ergebnis.
4. a wird der Wert von b zugewiesen und b der Wert von c.
5. Fahre mit Schritt 2 fort.

Berechne ggT von 22 und 12:

a	b	c
10	2	0

$c = 0$, also ist das Ergebnis $b = 2$.

Beispiel: Eigenschaften von Algorithmen

- Der Algorithmus terminiert, denn in jedem Schritt wird b kleiner, aber niemals kleiner als 0
- Der Algorithmus ist finit, er besteht aus genau fünf Schritten.
- Der Algorithmus ist deterministisch, jeder Schritt ist eindeutig bestimmt.
- ... dadurch ist der Algorithmus auch determiniert, da die Werte in jedem Schritt eindeutig vom vorherigen Schritt abhängen.

Beispiel: Eigenschaften von Algorithmen

Wir zeigen noch: Der Algorithmus ist effektiv.

- Bei Terminierung gilt: a ist Vielfaches von b (weil $c = 0$)

Beispiel: Eigenschaften von Algorithmen

Wir zeigen noch: Der Algorithmus ist effektiv.

- Bei Terminierung gilt: a ist Vielfaches von b (weil $c = 0$)
- Es gilt: $a = x \cdot \mathrm{ggT}(a, b)$ und $b = y \cdot \mathrm{ggT}(a, b)$ vor dem Schritt haben den gleichen ggT wie deren neue Werte b und a mod b nach dem Schritt:

Beispiel: Eigenschaften von Algorithmen

Wir zeigen noch: Der Algorithmus ist effektiv.

- Bei Terminierung gilt: a ist Vielfaches von b (weil $c = 0$)
- Es gilt: $a = x \cdot \mathrm{ggT}(a, b)$ und $b = y \cdot \mathrm{ggT}(a, b)$ vor dem Schritt haben den gleichen ggT wie deren neue Werte b und $a \bmod b$ nach dem Schritt:
  - $a \bmod b = x \cdot \mathrm{ggT}(a, b) \bmod y \cdot \mathrm{ggT}(a, b) =$ $(x \bmod y) \cdot \mathrm{ggT}(a, b)$, der ggT von a und b teilt also $a \bmod b$.

Beispiel: Eigenschaften von Algorithmen

Wir zeigen noch: Der Algorithmus ist effektiv.

- Bei Terminierung gilt: a ist Vielfaches von b (weil $c = 0$)
- Es gilt: $a = x \cdot \mathrm{ggT}(a, b)$ und $b = y \cdot \mathrm{ggT}(a, b)$ vor dem Schritt haben den gleichen ggT wie deren neue Werte b und a mod b nach dem Schritt:
  - a mod $b = x \cdot \mathrm{ggT}(a, b)$ mod $y \cdot \mathrm{ggT}(a, b) =$ $(x \bmod y) \cdot \mathrm{ggT}(a, b)$, der ggT von a und b teilt also a mod b.
  - Wenn es noch einen größeren gemeinsamen Teiler $t \cdot \mathrm{ggT}(a, b)$ von a mod b und b gäbe, dann wären x mod y und y nicht teilerfremd. Sie hätten den gemeinsamen Teiler t.

Beispiel: Eigenschaften von Algorithmen

Wir zeigen noch: Der Algorithmus ist effektiv.

- Bei Terminierung gilt: a ist Vielfaches von b (weil $c = 0$)
- Es gilt: $a = x \cdot \mathrm{ggT}(a, b)$ und $b = y \cdot \mathrm{ggT}(a, b)$ vor dem Schritt haben den gleichen ggT wie deren neue Werte b und a mod b nach dem Schritt:
  - a mod $b = x \cdot \mathrm{ggT}(a, b)$ mod $y \cdot \mathrm{ggT}(a, b) =$ $(x \bmod y) \cdot \mathrm{ggT}(a, b)$, der ggT von a und b teilt also a mod b.
  - Wenn es noch einen größeren gemeinsamen Teiler $t \cdot \mathrm{ggT}(a, b)$ von a mod b und b gäbe, dann wären x mod y und y nicht teilerfremd. Sie hätten den gemeinsamen Teiler t.
    Demnach sind x und y nicht teilerfremd, da $x = (x \operatorname{div} y) \cdot y + (x \bmod y)$ und beide Summanden durch t geteilt werden, ebenso wie y.

Beispiel: Eigenschaften von Algorithmen

Wir zeigen noch: Der Algorithmus ist effektiv.

- Bei Terminierung gilt: a ist Vielfaches von b (weil $c = 0$)
- Es gilt: $a = x \cdot \text{ggT}(a, b)$ und $b = y \cdot \text{ggT}(a, b)$ vor dem Schritt haben den gleichen ggT wie deren neue Werte b und a mod b nach dem Schritt:
  - a mod $b = x \cdot \text{ggT}(a, b)$ mod $y \cdot \text{ggT}(a, b) =$ $(x \text{ mod } y) \cdot \text{ggT}(a, b)$, der ggT von a und b teilt also a mod b.
  - Wenn es noch einen größeren gemeinsamen Teiler $t \cdot \text{ggT}(a, b)$ von a mod b und b gäbe, dann wären x mod y und y nicht teilerfremd. Sie hätten den gemeinsamen Teiler t.
    Demnach sind x und y nicht teilerfremd, da $x = (x \text{ div } y) \cdot y + (x \text{ mod } y)$ und beide Summanden durch t geteilt werden, ebenso wie y.
    Schließlich teilt $t \cdot \text{ggT}(a, b)$ sowohl a als auch b, was ein Widerspruch ist.

Korrektheit abhängig von Problemklasse

- Korrektheit von Algorithmen ist wichtig für korrekte Funktion der Software
- Algorithmen können nur korrekt in Hinblick auf die Spezifikation der Problemklasse sein

Korrektheit abhängig von Problemklasse

- Korrektheit von Algorithmen ist wichtig für korrekte Funktion der Software
- Algorithmen können nur korrekt in Hinblick auf die Spezifikation der Problemklasse sein
- Wichtig also: Eine klare Formulierung der Problemklasse
- Korrektheit entspricht mit gegebener Problembeschreibung Effektivität

Komplexität eines Algorithmus

- Gemessen in Speicherbedarf und Rechenschritten
- Eine niedrige Komplexität eines Algorithmus entspricht einer hohen Effizienz

Weitere weiche Kriterien an Algorithmen

- Robustheit (gegen fehlerhafte Eingaben)
- Anpassbarkeit (gegenüber veränderter Spezifikation)
- Wiederverwendbarkeit (für andere, verwandte Problemklassen)

Weitere weiche Kriterien an Algorithmen

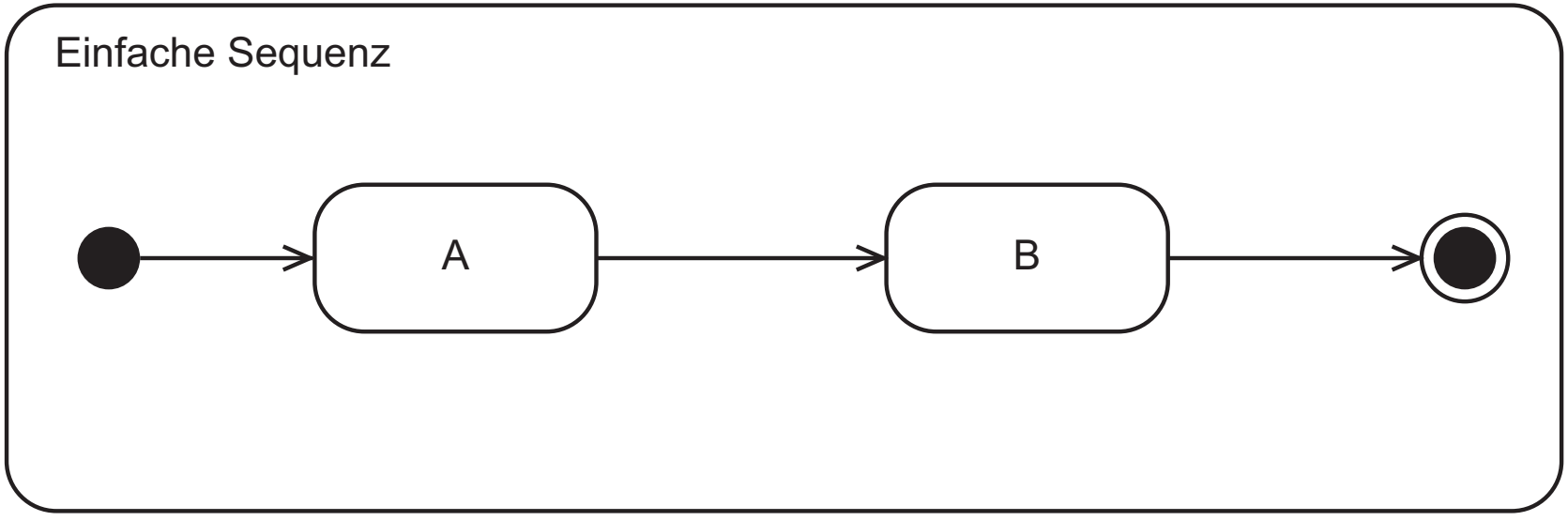
- Robustheit (gegen fehlerhafte Eingaben)
- Anpassbarkeit (gegenüber veränderter Spezifikation)
- Wiederverwendbarkeit (für andere, verwandte Problemklassen)

Diese Eigenschaften sind oft Gegenspieler zur Effizienz.

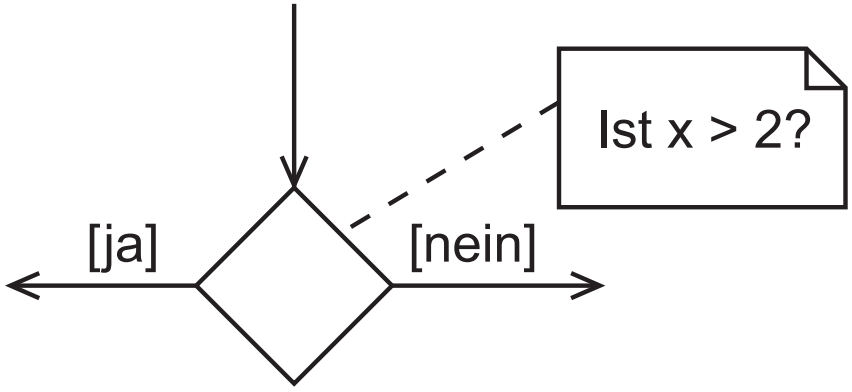
Formale Spezifikation von Algorithmen

- Bereits kennen gelernt: Natürlichsprachliche Beschreibung
- UML bietet Aktivitätsdiagramm
- Pseudocode unter Ausnutzung von Kontrollstrukturen die es in den meisten Programmiersprachen (auch Java) gibt

Aktivitätsdiagramme: Sequenz, Aktion, Initialknoten, Aktivitätsendknoten

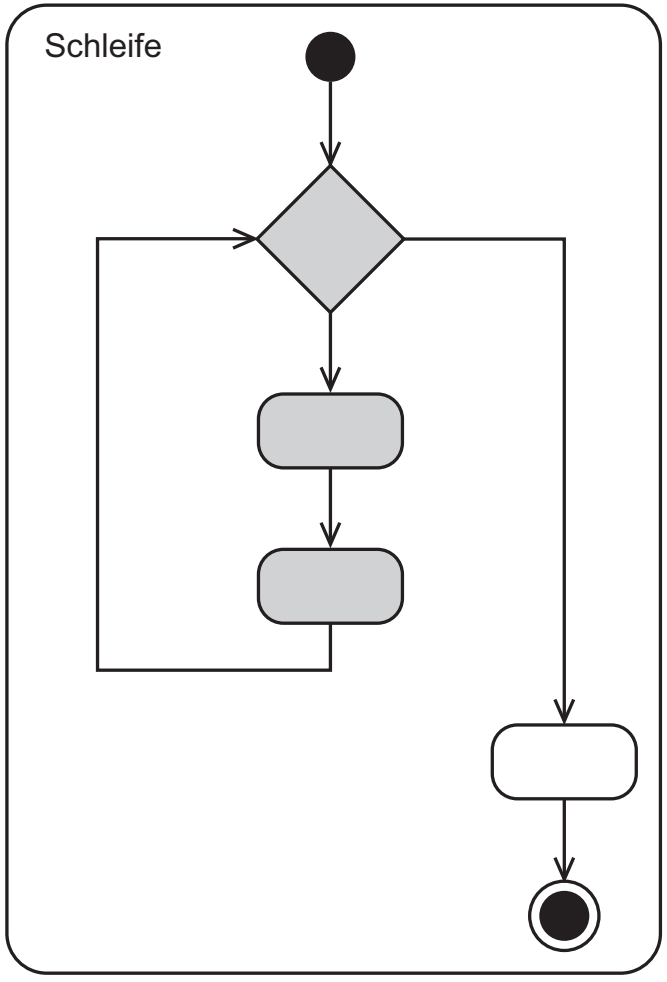


Aktivitätsdiagramme: Entscheidungsknoten, Vereinigungsknoten



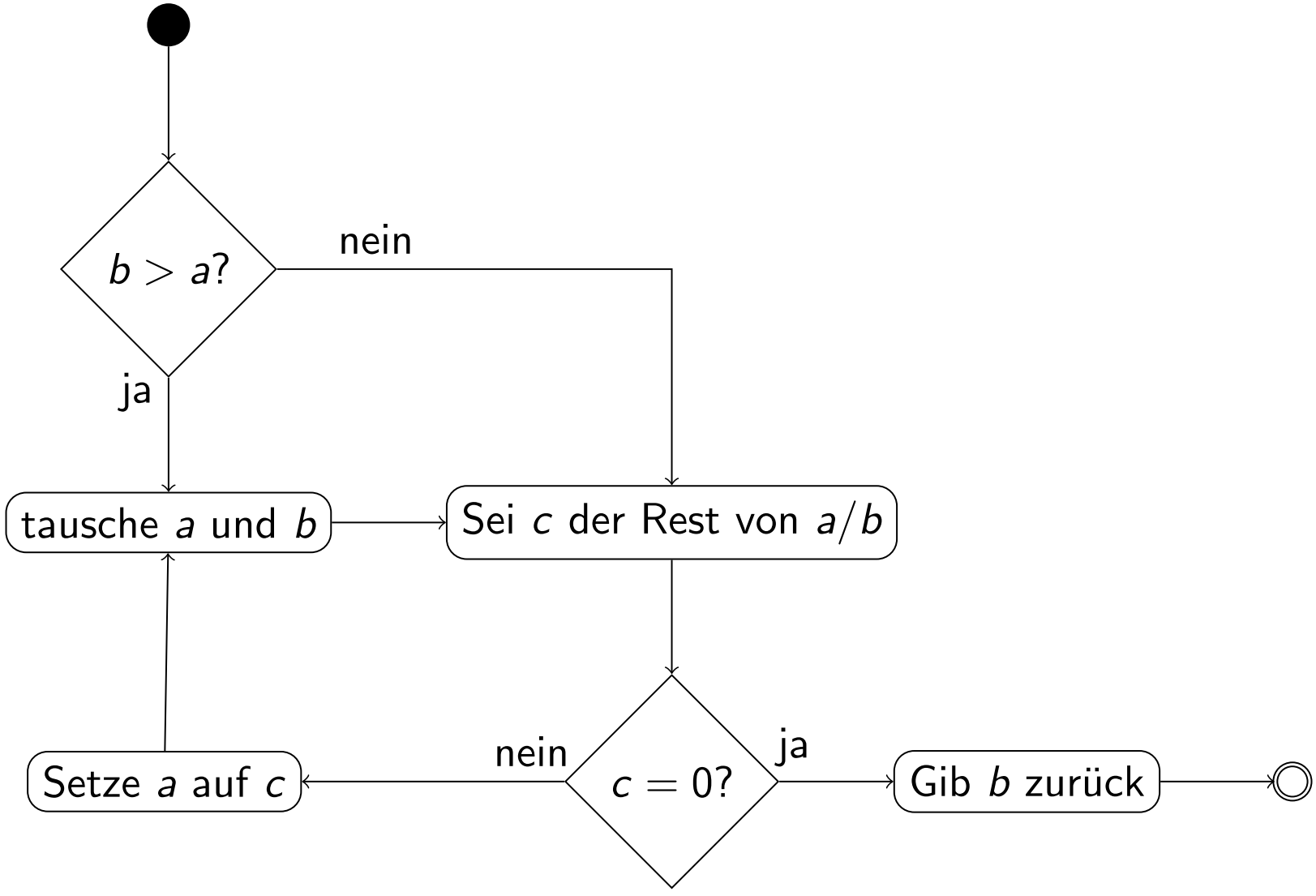
Pseudocode: `if`

Aktivitätsdiagramme: Schleife via Rückwärtskante



Pseudocode: `while`

Beispiel: ggT-Algorithmus als Aktivitätsdiagramm



Fragen zur Vorlesungseinheit

1. Aus welchen Bestandteilen besteht eine Klasse?
2. In welchem Zusammenhang stehen die Begriffe „Klasse“ und „Objekt“?
3. Was ist der Unterschied zwischen den Begriffen „effizient“ und „effektiv“?