FHD

Fachhochschule Darmstadt
Fachbereich Informatik

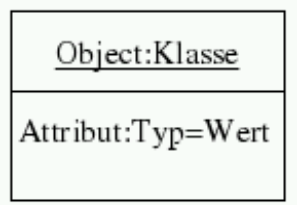
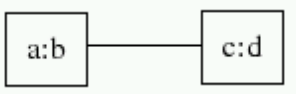
Softwaretechnik I

Kapitel 3 ff

- weitere Diagramme in der UML

Quellenhinweis:
Einige Folien zu dieser Vorlesung entstammen Präsentationen von Prof. G. Raffius und Prof. W. Weber

Objektdiagramm: Notation in UML 2.0

	**Objekt** Für jedes Objekt wird der Objektname und der Klassenname angegeben. Für die dargestellten Attribute wird der Attributname, Typ und die aktuelle Wertbelegung angegeben. Der Name des Objekts und der Klassenname werden unterstrichen.
	**Link** Ein Link ist eine aktuelle Beziehung zwischen Objekten. Er wird ähnlich der Assoziation im Klassendiagramm, als Linie zwischen den Objekten dargestellt. Es kann ein Name oder Rollenbezeichnungen angegeben werden

Objektdiagramm: Inhalt

Ein Objektdiagramm enthält folgende Elemente

- Objekt (Instanz von Klassen)
- Wert (Instanz von Attributen)
- Link (Instanz von Assoziationen)

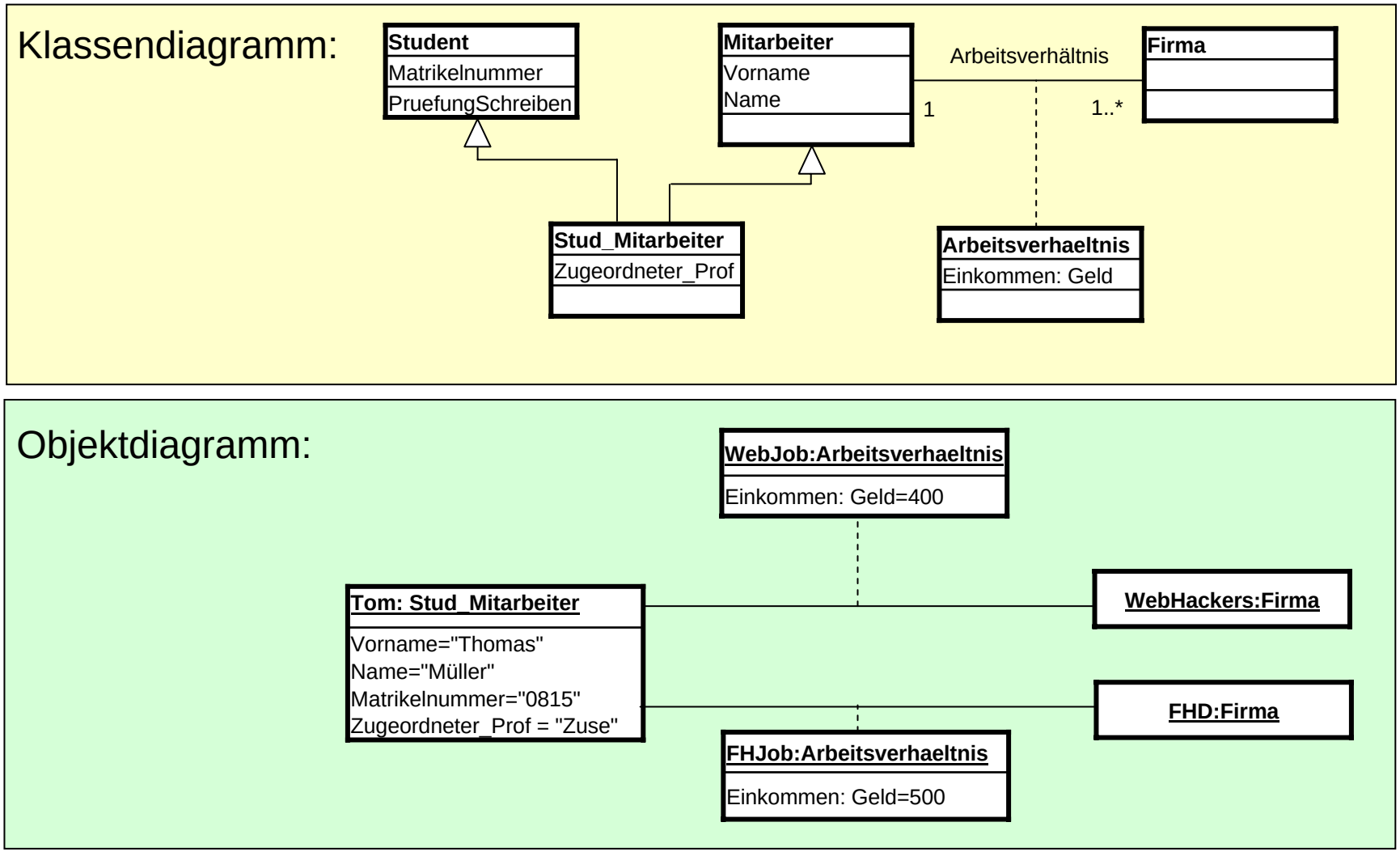
Instanzen von *Methoden* werden in Sequenz-diagrammen dargestellt!

Ein Objektdiagramm zeigt

- **Instanzen zu einem bestimmten Zeitpunkt**

⇨ **einen "Snapshot" des Systems**

Objektdiagramm: Beispiel



Quelle: UML 2 glasklar, C. Rupp et.al.

Objektdiagramm: Diskussion

Vergleich mit dem Klassendiagramm

Klassendiagramm	Objektdiagramm
stellt Vererbungsbeziehungen dar	Stellt das Ergebnis der Vererbung dar
Zeigt Assoziationen in der allgemeinen Form	Zeigt Assoziationen als konkrete Links zu Objekten
Keine Darstellung von Werten (außer Defaultwerten)	Konkrete Werte
Abstraktion teilweise schwer zu verstehen	Unvollständige Darstellung in "Beispielform"

Objektdiagramm: Anwendung

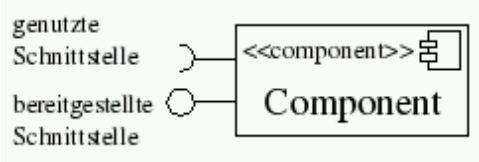
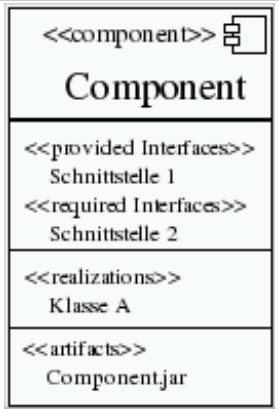
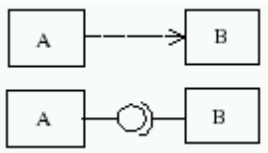
Objektdiagramme sind z.B. in folgenden Situationen nützlich:

- zur Dokumentation von Architekturen mit abstrakten Klassen
- zur Erläuterung von zirkulären Assoziationen

  (z.B. *n* Personen kennen *n* Personen)
- zur Überprüfung der Modellierung mit konkreten Beispielen
- zur Darstellung von konkreten Daten zum Debugging / Testen
- Zum Darstellen der Daten- bzw. Objekt-Verteilung in verteilten Systemen

Komponentendiagramm: Notation in UML 2.0

verändert gegenüber UML1.x!

	**Komponente (externe Sicht)** Eine Komponente wird als Rechteck dargestellt. Sie trägt die Stereotypbezeichnung <<component>> und hat einen Namen. In der rechten, oberen Ecke kann sie das Komponentensymbol tragen. Das Symbol mit dem vollen Kreis stellt eine Schnittstelle dar, die die Komponente bereitstellt; das Symbol mit dem Halbkreis bezeichnet eine Schnittstelle, die von der Komponente genutzt wird.
	**Komponente (interne Sicht)** Komponenten können, ähnlich wie Klassen, vollständig dargestellt werden. In dem Bereich unter dem Namen werden die geforderten und bereitgestellten Schnittstellen aufgeführt; im nächsten Abschnitt werden die Klassen, die die Komponente realisieren angegeben. Im unteren Bereich wird die Datei angegeben, die die Implementierung der Komponente enthält.
	**Beziehungen** können im Komponentendiagramm als gestrichelte Linien angegeben werden. Der Pfeil gibt zu Zugriffsrichtung an. Sind die Schnittstellen der Komponenten angegeben, ergibt sich aus den Symbolen die Zugriffsrichtung.

Komponentendiagramm : Inhalt

mehr als die Zuordnung zu Quellcodedateien in UML1.4!

Komponenten

- sind modulare Systemteile, die ihren Inhalt kapseln (und vor dem Benutzer verbergen)
- sind eigenständige Anwendungen mit einer klar definierten Schnittstelle (bereitgestellte und benutzte Funktionalität)
- bestehen aus enthaltenen Klassen oder Komponenten

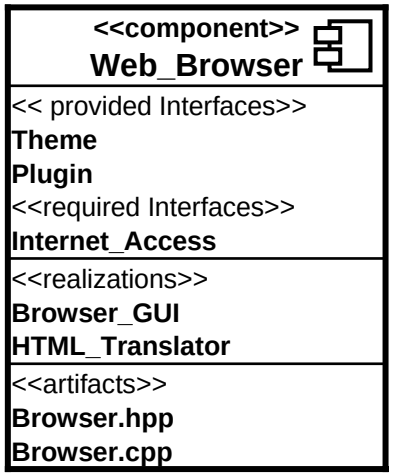
Anpassung an den Komponenten-begriff im Sinne von CORBA, COM, Java,…

Ein Komponentendiagramm beschreibt

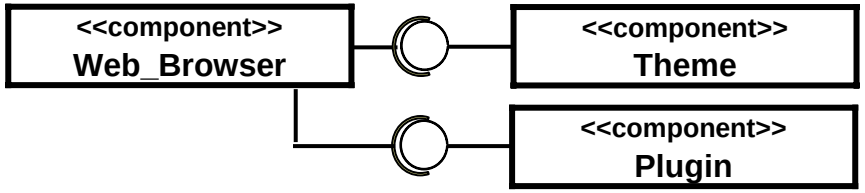
- die Komponenten, ihre Beziehungen und die öffentlichen Schnittstellen
- die Struktur eines Systems
- wie diese Strukturen erzeugt werden
- die physikalischen Bestandteile eines Systems

Dateien mit Source-Code, Dokumentation, ByteCode etc. heißen jetzt Artefakte (stereotyp <<artifact>>)

Komponentendiagramm : Beispiel



White-Box
(mit Interna)



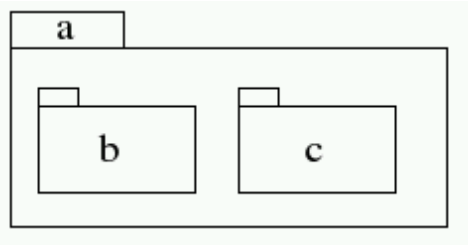
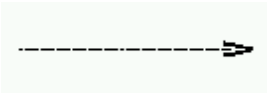
Black-Box
(Überblick)

Komponentendiagramm : Anwendung

Komponentendiagramme sind z.B. in folgenden Situationen nützlich:

- Entwicklung von SW mit Komponententechnologie
- Entwicklung von SW-Komponenten zur Wiederverwendung (Austauschbarkeit von Komponenten als Black-Box)
- Verteilte SW-Entwicklung (Darstellung von Schnittstellen)

Paketdiagramm: Notation in UML 2.0

	**Paket** Ein Paket fasst eine Gruppe von beliebigen Modellelementen zusammen. Pakete können verschachtelt sein. Sie definieren einen Namensraum. In diesem Beispiel umfasst das Paket a die Pakete b und c.
	**Abhängigkeiten** Zwischen Paketen werden als gestrichelte Pfeile dargestellt. Sie drücken aus, dass Pakete in einem Client-Server-Verhältnis zueinander stehen. Häufige Stereotypen für Abhängigkeiten in Paketdiagrammen sind <<import>> und <<access>>, womit ausgedrückt wird, dass ein Paket ein anderes importiert, bzw. darauf zugreift.

Paketdiagramm : Inhalt

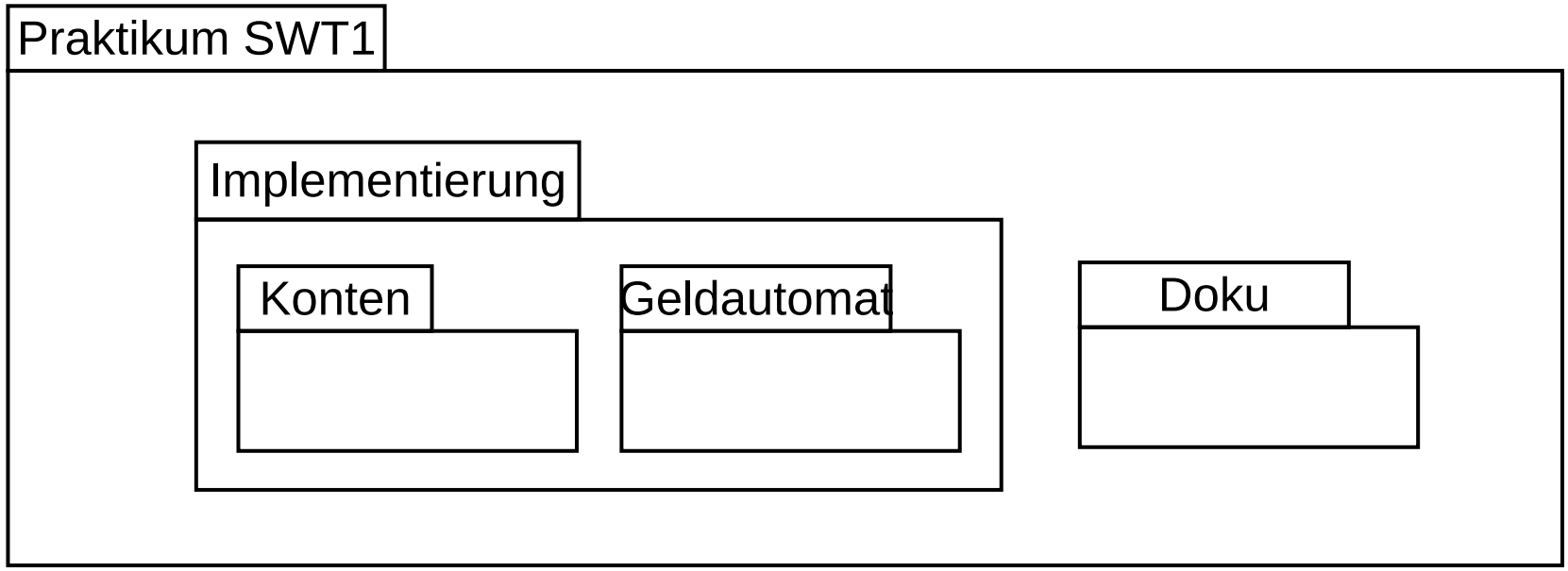
Ein Paketdiagramm beschreibt

- eine Aufteilung des Systems in Gruppen ("Pakete")
  – ähnlich zu Ordnern in Dateisystemen
- die Pakete und deren Beziehungen untereinander

Anmerkungen

- Ein Modellelement darf nur zu einem Paket gehören
- Ein Paket definiert einen Namensraum und eine Sichtbarkeit
- Pakete können verschachtelt sein

Paketdiagramm : Beispiel



Paketdiagramm : Anwendung

Paketdiagramme sind z.B. in folgenden Situationen nützlich:

- ein großes System soll in handhabbare Arbeitspakete aufgeteilt werden
- Modellelemente werden nach Themen gruppiert (Use Cases, Klassen, Diagramme,…)
  - funktional
  - logisch
  - in Schichten
  - ...