

Überblick

- Von C nach Java
- Background
- Konzepte des objektorientierten Programmierens
- Ein ausgearbeitetes Java-Programm als Beispiel
- Ein weiteres Java-Programm als Beispiel
- Typen und Subtypen
- ➔ ■ Vom Design zum Programmieren (OOD zu OOP)
- Patterns in OOP
- Testen (von objektorientierten Programmen)



Institute of Computer Technology



Vom Design zum Programmieren (OOD zu OOP)

- Übersicht zu OOD und UML (Unified Modeling Language)
- Übergang zum Programmieren
- UML vs. Java
- Was ist Generalisierung (in UML 2)?
- Wie implementiert man Generalisierung (in Java)?
- Wie implementiert man Assoziationen (in Java)?
- Wie implementiert man Interaktionen (in Java)?



Institute of Computer Technology

Was ist (Software-)Design?

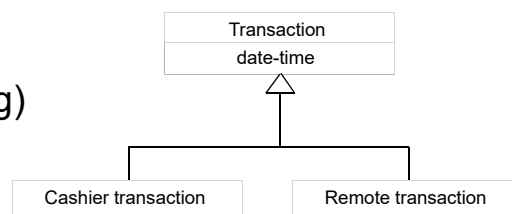
- *UML Glossary* 
"The phase of the system development process whose primary purpose is to decide how the system will be implemented. During design strategic and tactical decisions are made to meet the required functional and quality requirements of a system."
- **Objektorientiertes Design** (OOD) ist Design mit (Design-)Objektklassen. 

Klasse, Generalisierung und Vererbung

Klasse in **UML** (Unified Modeling Language)
<http://www.omg.org>

**Generalisierung /
Spezialisierung**
(in UML-Darstellung)

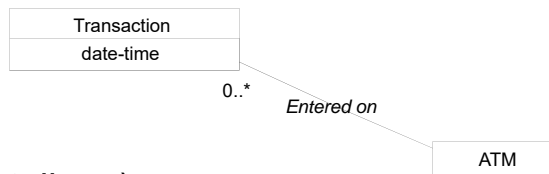
Vererbung





Attribute und Assoziationen

Attribut für die Darstellung von Eigenschaft



Assoziation
(in UML-Darstellung)

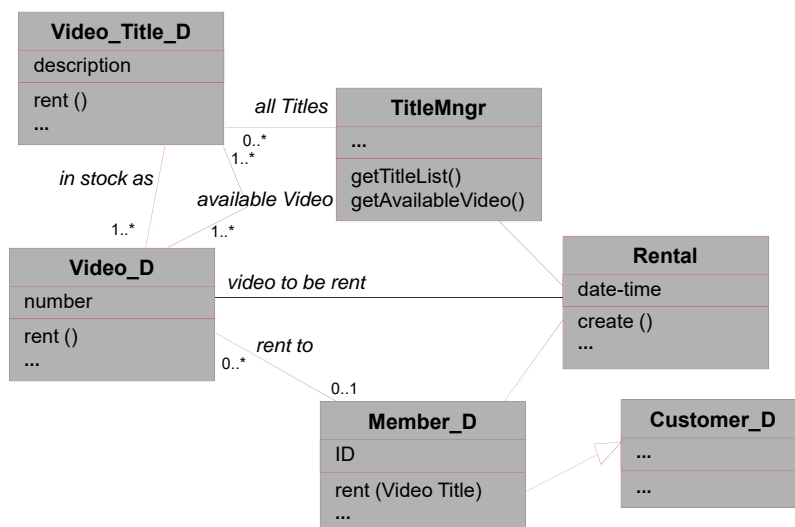
Relation für das Verbinden von Instanzen
Multiplizität: Bereich zulässiger Kardinalitäten



Institute of Computer Technology

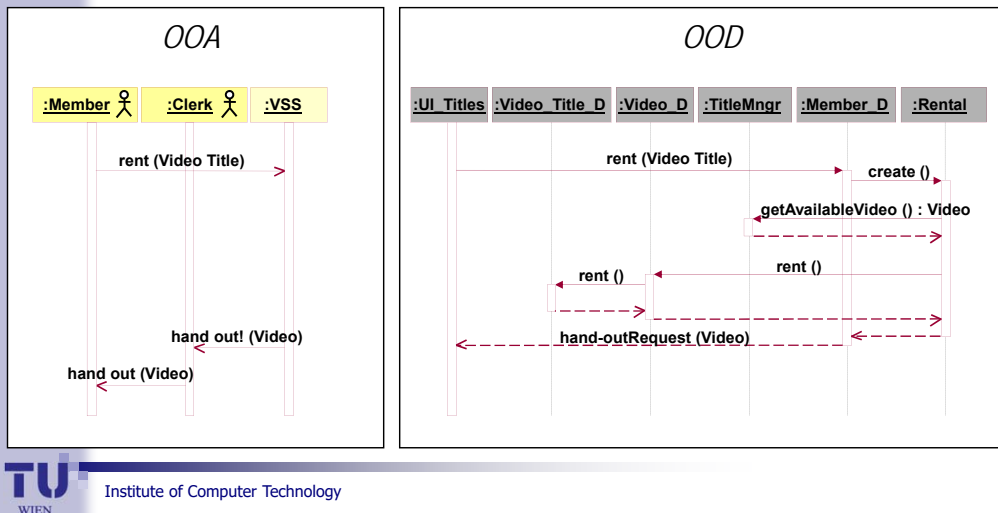


(Teil von) OOD-Modell



Institute of Computer Technology

Sequenzdiagramme – Beispiel Videogeschäft



Übergang zum Programmieren

- Wie implementiert man solche Modelle in einer Programmiersprache (wie zum Beispiel Java)?
- Abbildung von Attributen auf (Instanz-)Variable
- Programmieren von Algorithmen in Methoden oder durch Senden von Messages
- Alle fehlenden Details


UML vs. Java

- Primär grafische vs. textuelle Sprache
- Modellierungs- vs. Programmiersprache
- Nicht-operationale vs. operationale Sprache (Ausnahme XUML – Executable UML)
- Trotzdem sehr ähnlicher objektorientierter Ansatz
- Aber was ist mit
 - Generalisierung vs. Subklassen?
 - Assoziationen?
 - Interaktionen?

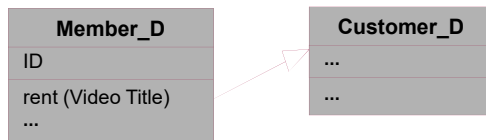
Rental
date-time
create ()
...

```
class rental {  
    ... date-time;  
    void create() {  
        ...  
    }  
    ...  
}
```

Was ist Generalisierung (in UML 2)?

- *UML Glossary* 
"A taxonomic relationship between a more general classifier and a more specific classifier. Each instance of the specific classifier is also an indirect instance of the general classifier. Thus, the specific classifier indirectly has features of the more general classifier."

Wie implementiert man Generalisierung (in Java)?



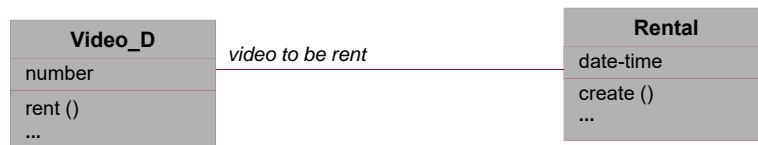
Entsprechender Java Code:

```
public class Member_D extends Customer_D { ... };
```

Was ist mit **Subtyping**?

Muss zusätzlich behandelt werden!

Wie implementiert man Assoziationen (in Java)?



Entsprechender Java Code:

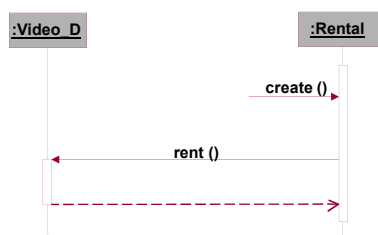
```
public class Video_D { ...
    public Rental theRental;
... };

public class Rental { ...
    public Video_D videoToBeRent;
... };
```

Wie implementiert man Assoziationen (in Java)? (Fortsetzung)

- Wie wäre es mit **Datenkapselung** der Assoziation?
private oder protected
- Wie geht man mit **Navigation** der Assoziation um?
Methoden für den Zugriff
- Wie geht man mit anderen **Kardinalitäten** einer Assoziation um?
array oder vector
- Wie wäre es mit einem **nicht-invasiven** Ansatz?
Assoziation implementiert durch eine zusätzliche Klasse

Wie implementiert man Interaktionen (in Java)?



Senden von Messages

im Code der Methode `create` der Klasse `Rental`:
`videoToBeRent.rent();`

Wie implementiert man Interaktionen (in Java)? (Fortsetzung)

- Prozedurale Sequenzdiagramme mit **synchronen** Messages
Methoden-Aufrufe ähnlich wie Unterprogramme
- Wie geht man mit **asynchronen** Messages zwischen Objekten um?
threads oder events