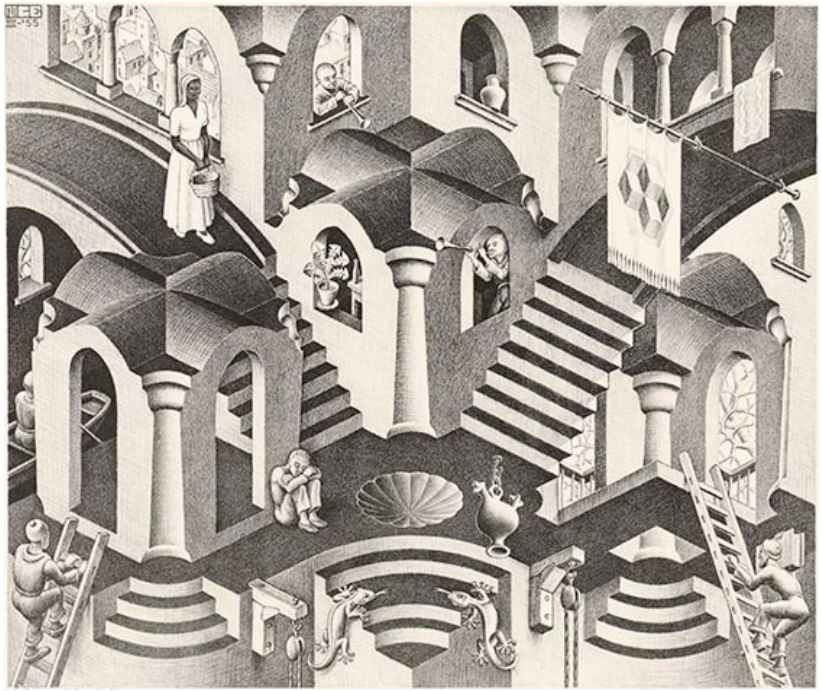


Software Architectures

Part 1: Modeling and Design Principles

Knowledge

- What is architecture?
- Architectural objectives in software design
- UML structural modeling elements

Skills

- Can detect violations of object oriented design principles
- Can explain all object oriented design principles using UML diagrams

The Parthenon in Athens, Greece

Some Genius Improvising

The Houses of Parliament, London

The Bauhaus in Dessau, Germany

The Taj Mahal, India

The National Congress of Brazil

The Carpenter did the Design

The Gare de Oriente in Lisbon, Portugal

The Golden Pavilion in Kyoto, Japan

Notre Dame de Paris, France

MI6 Headquarter in London

Shantytown, New Orleans

The Opera House in Sydney, Australia

Architecture

- A general term to describe buildings and other physical and some non-physical structures.
- The art and science of designing buildings and (some) nonbuilding structures (”Baukunst”).
- The style of design and method of construction of buildings and other physical and non-physical structures.

Architecture has to consider

- functional
- technical
- aesthetic

aspects in the planning, designing, and construction of its artefacts.

We use *models* to describe *things* in the real world.

Models are abstractions of the real world, simplifying reality.

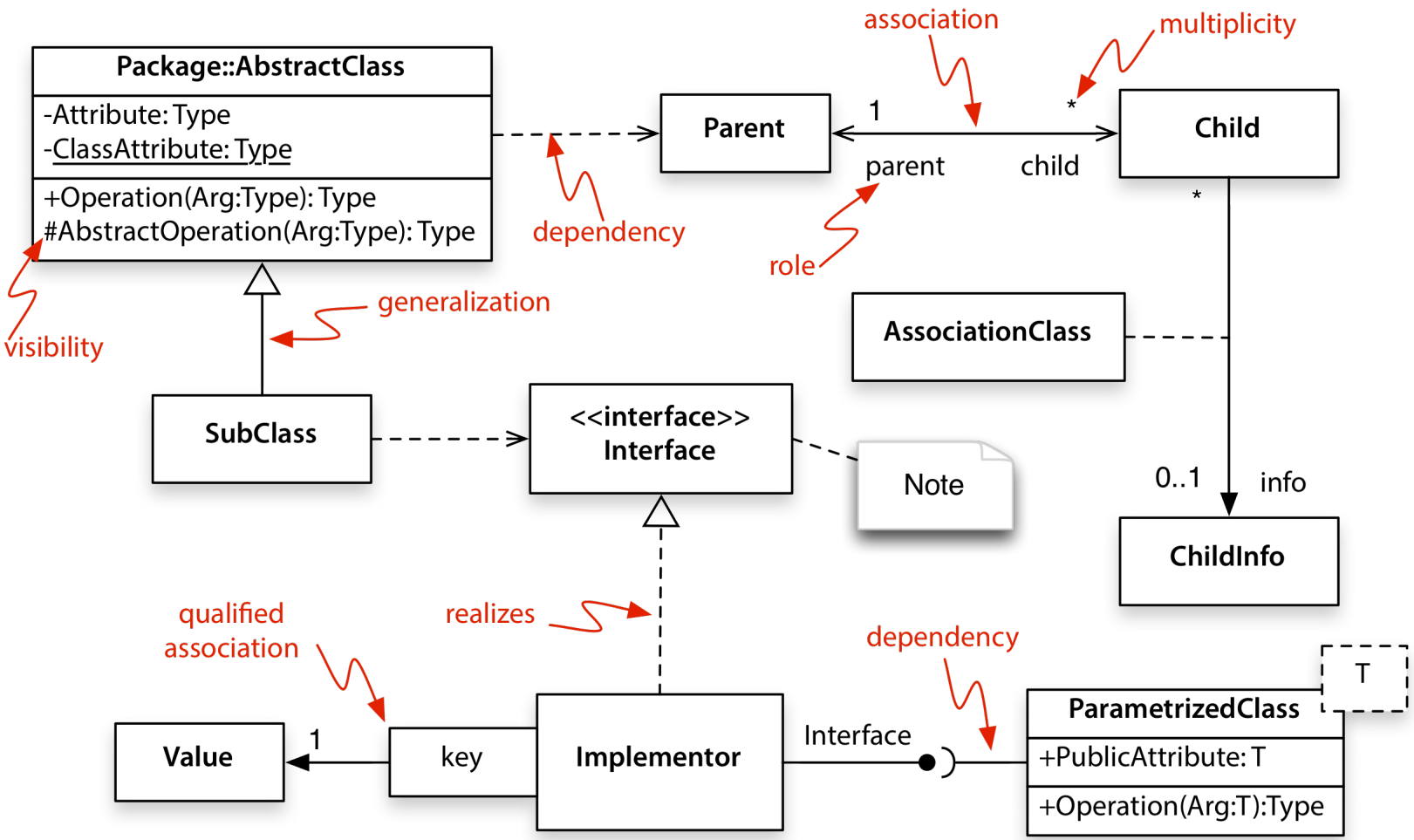
Models describe

- *Structure*
- *Behaviour*

Examples

- Buildings (structure)
- Elevators (structure and behaviour)
- Buying something from an online-shop (behaviour)
- Cars (structure and behaviour)

Describing architecture means describing structure.
For software systems we use the *Unified Modelling Language* (*UML*).

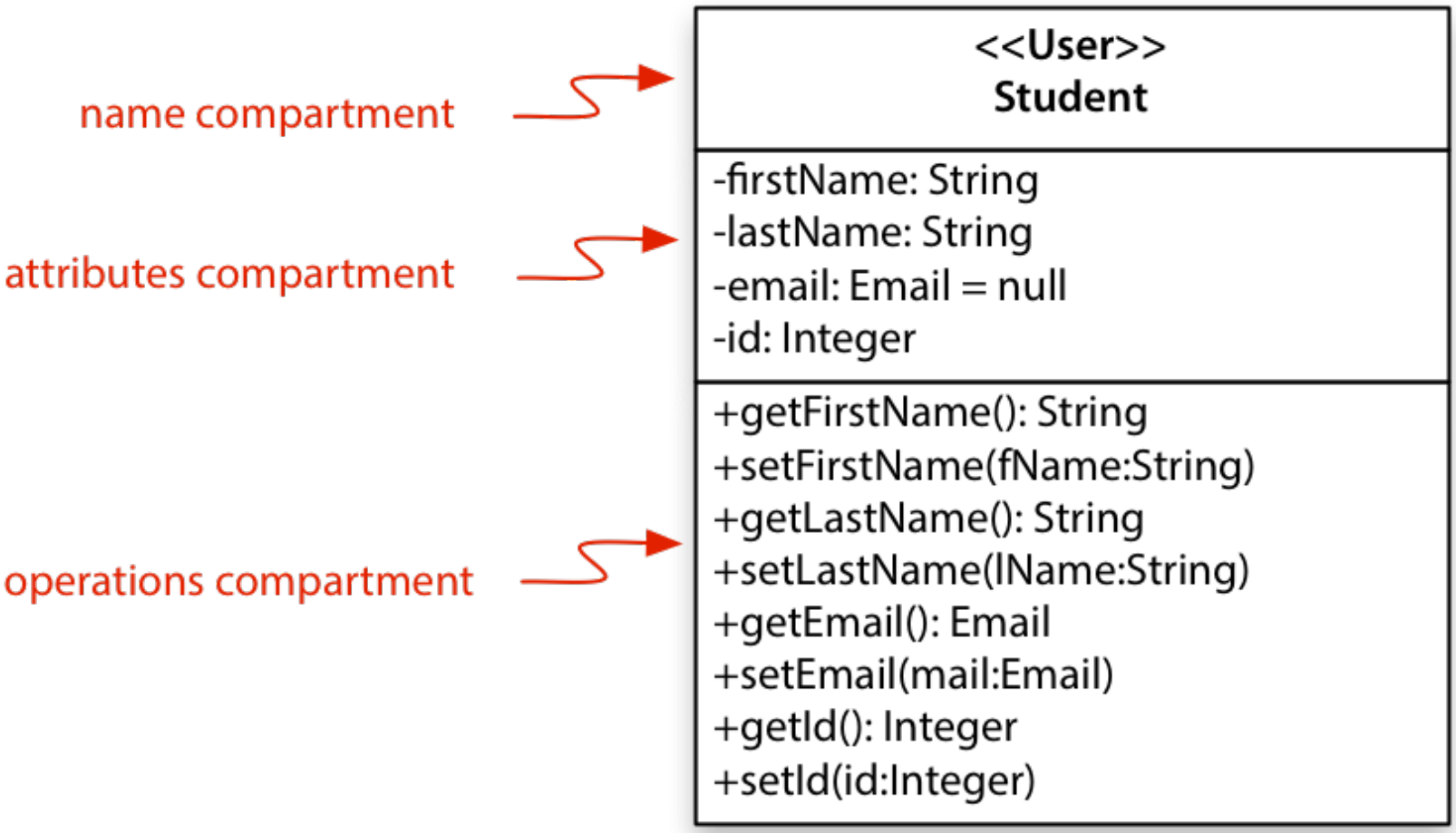


The UML offers a rich set of diagram types to describe structure and behaviour.

- **Class diagrams** ← this is what we use here
- Component diagrams
- Activity diagrams ← covered in "Real-Time Systems"
- State charts ← covered in "Real-Time Systems"
- Sequence diagrams
- Use case diagrams
- and eight more ...

- *Attributes* describe the appearance and knowledge of a class of objects.
- *Operations* define the behavior that a class of objects can manifest.
- *Stereotypes* help you understand this type of object in the context of other classes of objects with similar roles within the system's design.
- *Properties* provide a way to track the maintenance and status of the class definition.
- *Association* is just a formal term for a type of relationship that this type of object may participate in. Associations may come in many variations, including simple, aggregate and composite, qualified, and reflexive.
- *Inheritance* allows you to organize the class definitions to simplify and facilitate their implementation.

Class Diagram: Class Element Compartments

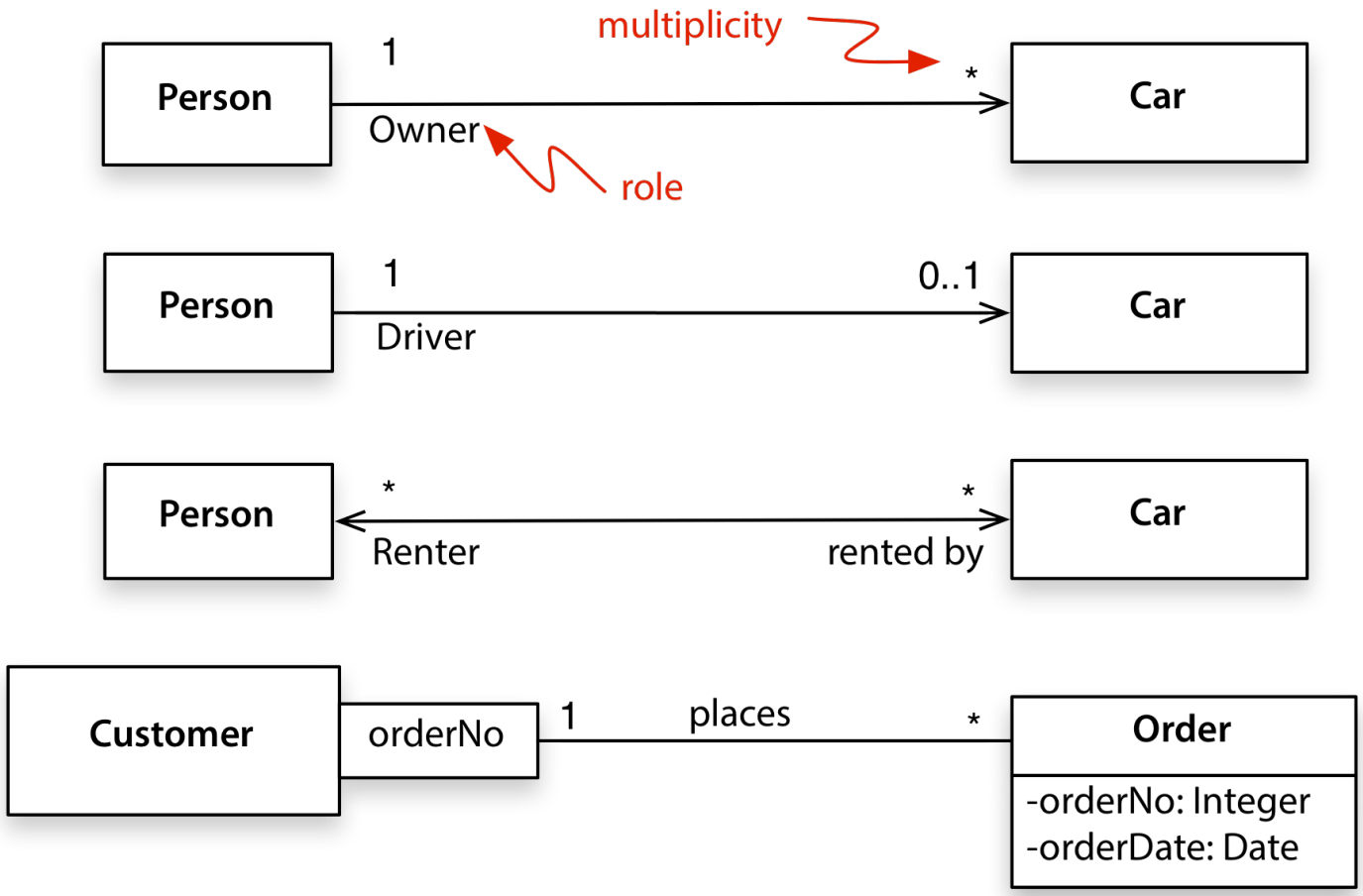


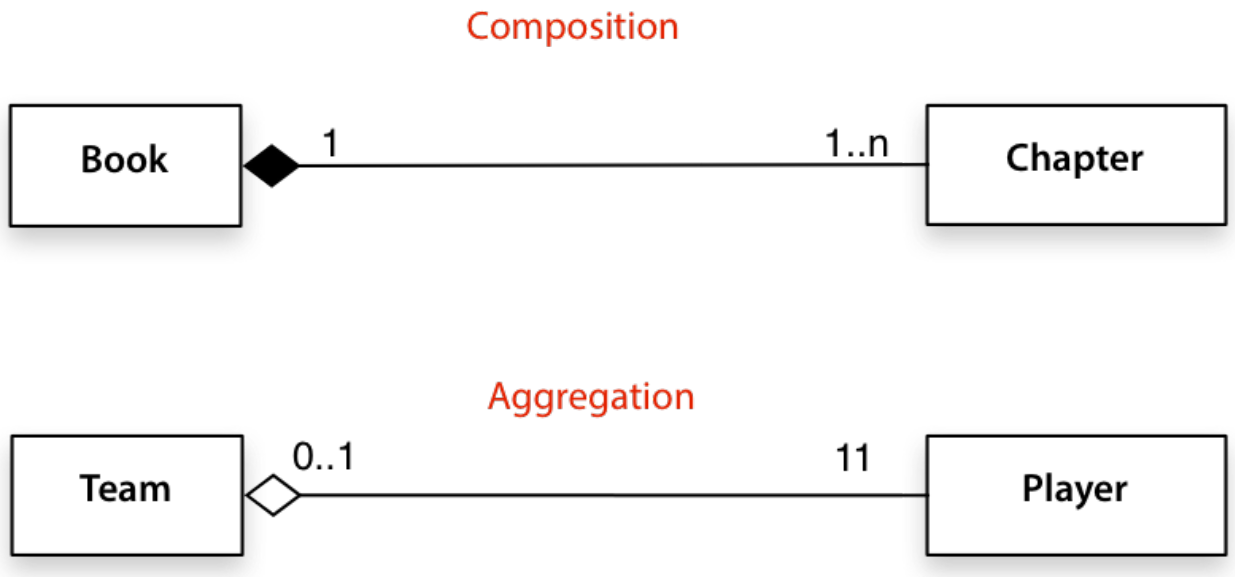
Each attribute definition must specify what other objects are allowed to see the attribute – that is its visibility. Visibility is defined as follows:

- Public (+) visibility allows access to objects of all other classes.
- Private (-) visibility limits access to within the class itself. For example, only operations of the class have access to a private attribute.
- Protected (#) visibility allows access by subclasses. In the case of generalizations (inheritance), subclasses must have access to the attributes and operations of the superclass or they cannot be inherited.
- Package (~) visibility allows access to other objects in the same package.

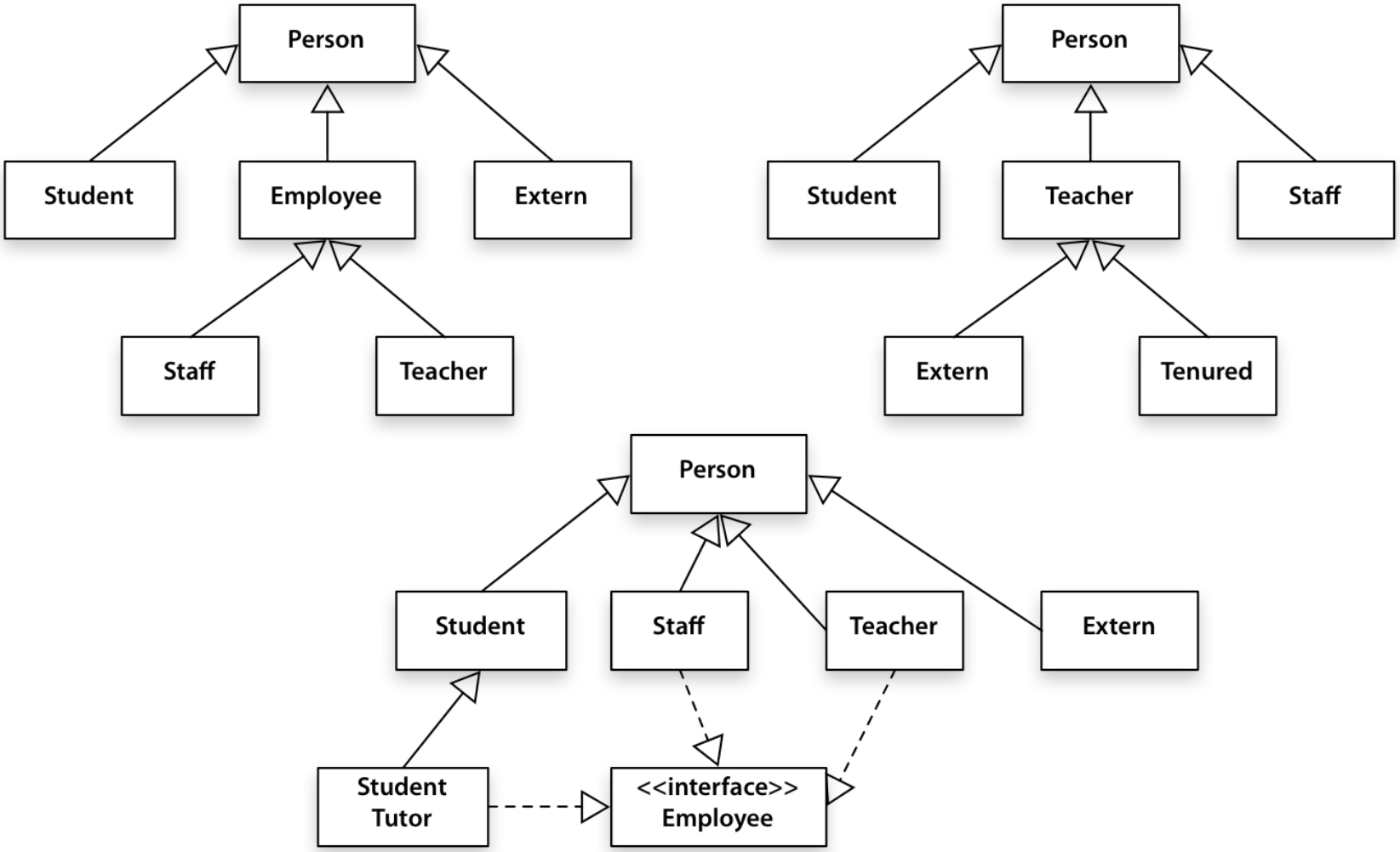
Attribute Element Description	Attribute Element Example
Create an attribute name	company
Add the attribute data type	company:character
Add the attribute's default value, if any	company:character = spaces
Set the constraints on the attribute value. For this example, first identify the field length.	company:character = spaces {1 to 30 characters}
Next identify the types of data that can be used in the attribute. Add this information within the brackets.	company:character = spaces {1 to 30 characters including alphabetic, spaces, and punctuation characters; no special characters allowed}
Set the attribute visibility (designate private visibility with a minus (-) sign in front of the attribute).	- company:character = spaces {1 to 30 characters including alphabetic, spaces, and punctuation characters; no special characters allowed}

- Operation name: Required. The combination of name and parameters does need to be unique within a class.
- Arguments/parameters: Each argument requires an identifier and a data type.
- Return data type: Required for a return value, but return values are optional.
- Visibility (+, -, #, ~): Required before code generation.
- Class level operation (underlined operation declaration): Optional. In Java: static methods.
- Argument name: Required for each parameter, but parameters are optional. Any number of arguments is allowed.
- Argument data type: Required for each parameter, but parameters are optional.





Class Diagram: Generalization



The system to be designed must be

- correct
- extensible
- understandable

To that end we use object oriented design principles

Principles for Single Class Design

- Encapsulation, abstraction, and information hiding
- Separation of concerns and the single-responsibility principle
- Interface segregation principle

Principles for Design of Class Cooperation

- Loose Coupling
- Liskov substitution principle
- Design by contract
- Open-close principle
- Dependency inversion principle

Im Ordner `figures` befinden sich die folgenden Bilder im eps- und png-Format sowie eine avi-Animation.

HEGoeppingen	Das Hochschulgebäude
HEGrid	Das Grid (Titelseite rechts oben)
HELogo	Das Logo der Hochschule Esslingen
TubeKnot	Das Standbild der Knoten-Animation
TubeKnot.avi	Eine kurze Animation eines Knotens

Die Bilder `HELogo`, `HEGoeppingen` und `HEGrid` werden für die Titelseite benutzt. `HELogo` erscheint außerdem in jeder Titelzeile eines Frames.

Es liegt eine Beispiel-Datei in drei Formen bei.

example-beamer.tex	die Beamer-Präsentation
example-handout.tex	A4-Handout der Beamer-Frames
example-article.tex	Article-Handout

Die drei PDF-Dateien können entweder gemeinsam über das beiliegende make-File oder einzeln mit den Befehlen

```
pdflatex example-beamer.tex
pdflatex example-handout.tex
pdflatex example-article.tex
```

erzeugt werden. Die drei LaTex-Dateien binden jeweils die Dateien `preamble.tex` und `example.tex` ein.

- Die Einträge in der Fußzeile werden aus den optionalen Argumenten der Befehle `title`, `subtitle`, `author`, und `date` in der Titelseite zusammengesetzt.
- Die Fußzeile kann zwischen den Frames mit dem Befehl

  ```
  \setbeamertemplate{footline}{}
  ```

  für die nachfolgenden Frames ausgeschaltet werden. Dadurch gewinnt man etwas Platz auf der Seite.

Table: Vordefinierte Farben entsprechend Corporate Design

Farbe	Kürzel	RGB-Wert	Verwendungsbeispiel
	HEblue6	0, 70, 102	Überschriften
	HEblue5	85, 146, 172	
	HEblue4	168, 203, 221	Hauptüberschriften
	HEblue3	206, 226, 236	
	HEblue2	236, 242, 245	
	HEblue1	245, 248, 249	Abbildungshintergrund
	HEgray1	112, 113, 114	Aufzählungszeichen
	HEred2	212, 0, 50	Überschriften, Logo
	HEred1	230, 143, 126	

Mit dem Befehl

```
\setbeamertemplate{footline}[Esslingen theme]
```

wird die Fußzeile wieder auf die voreingestellte Variante zurückgesetzt.

Die Itemize-Liste

- Erste Ebene.
  - Zweite Ebene.

Die Enumerate-Liste

1. Erster Eintrag
   1.1 Erster Eintrag, zweite Ebene

Die Description-Liste

Eintrag Erläuterung

Das Layout der Zähler und Listen-Items lässt sich leicht anpassen. Diese Möglichkeit sollte aber nur zurückhaltend genutzt werden.

- Erste Ebene.
  - Zweite Ebene.
    - Dritte Ebene.

1. Erster Eintrag
   (i) Erster Eintrag, zweite Ebene
   a) Erster Eintrag, dritte Ebene

Ob die Blöcke mit farbigem Hintergrund ausgegeben werden und wie kräftig dieser ist, kann mit den folgenden Color-Themes in der Datei preamble.text gesteuert werden. (Voreingestellt ist `rose`.)

- `% \usecolortheme{lily}% kein Farb-Hintergrund`
- `\usecolortheme{rose}% dezenter Farb-Hintergrund`
- `% \usecolortheme{orchid}% kräftiger Farb-Hintergrund`

Überschrift

Ein Block.

Überschrift

Ein Alert-Block.

Example

Ein Beispiel-Block.

Theorem

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam

$$\sum_{i=1}^{\infty} \frac{1}{x^2} \tag{1}$$

Proof.

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. □

Quellcode-Listings lassen sich in-line `print "hello world"` und in der lstlisting-Umgebung des Pakets listing.sty darstellen.

```
    cout << "Hello world!";
```

huuhu

```
   import com.aurel.track;

public class TestClass {
    private String dummy;
}
   cout << "Hello world!";
```

Wie und ob Animationen angezeigt werden, ist abhängig vom Betriebssystem und vom PDF-Betrachter.

Klicken Sie auf das Bild, um die Animation mit einem externen Betrachter anzuzeigen.

Richtlinien für die Anfertigung von Seminar-, Bachelor- und Masterarbeiten
Fakultät Betriebswirtschaft; Hochschule Esslingen; Flandernstraße 101; 73732 Esslingen

LaTeX Beamer class development repository.
`https://bitbucket.org/rivanvx/beamer/wiki/Home`

Till Tantau, Joseph Wright, and Vedran Miletić
The beamer class – User Guide for version 3.10.
`http://www.ctan.org/tex-archive/macros/latex/contrib/beamer/doc/beameruserguide.pdf`

Happy LaTeX-ing!