Software Design, Modelling and Analysis in UML

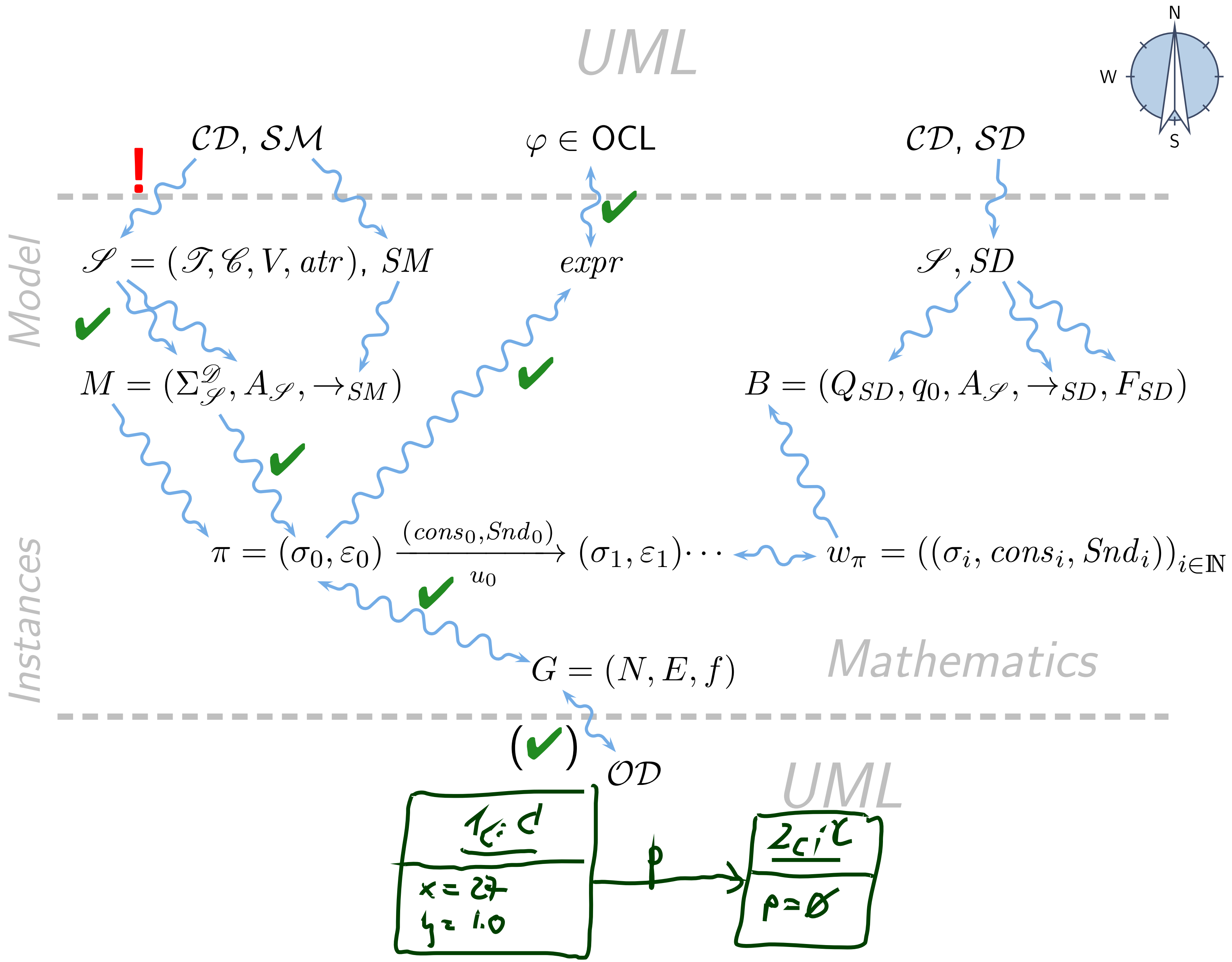
Lecture 06: Class Diagrams I

2014-11-11

Prof. Dr. Andreas Podelski, **Dr. Bernd Westphal**

Albert-Ludwigs-Universität Freiburg, Germany

Course Map


UML
N
W
E
S
$\mathcal{CD}$, $\mathcal{SM}$
$\varphi \in$ OCL
$\mathcal{CD}$, $\mathcal{SD}$
Model
$\mathscr{S} = (\mathscr{T}, \mathscr{C}, V, atr)$, SM
$expr$
$\mathscr{S}$, SD
$M = (\Sigma_{\mathscr{S}}^{\mathscr{D}}, A_{\mathscr{S}}, \to_{SM})$
$B = (Q_{SD}, q_0, A_{\mathscr{S}}, \to_{SD}, F_{SD})$
Instances
$\pi = (\sigma_0, \varepsilon_0) \xrightarrow[u_0]{(cons_0, Snd_0)} (\sigma_1, \varepsilon_1) \cdots$
$w_\pi = ((\sigma_i, cons_i, Snd_i))_{i \in \mathbb{N}}$
$G = (N, E, f)$
Mathematics
$\mathcal{OD}$
UML

Contents & Goals

Last Lecture:

- OCL Semantics
- Object Diagrams

This Lecture:

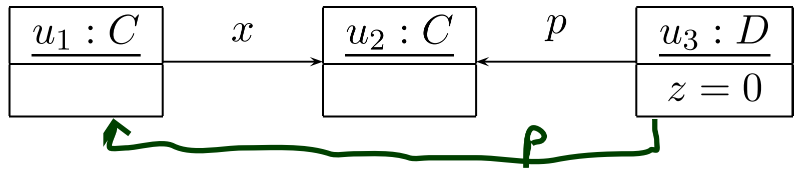
- **Educational Objectives:** Capabilities for following tasks/questions.
  - What is a class diagram?
  - For what purposes are class diagrams useful?
  - Could you please map this class diagram to a signature?
  - Could you please map this signature to a class diagram?

- **Content:**
  - Final notes on object diagrams.
  - Study UML syntax.
  - Prepare (extend) definition of signature.
  - Map class diagram to (extended) signature.
  - Stereotypes – for documentation.

The Other Way Round

The Other Way Round

- If we **only** have a picture as below, we typically assume that it's **meant to be** an object diagram wrt. **some** signature and structure.

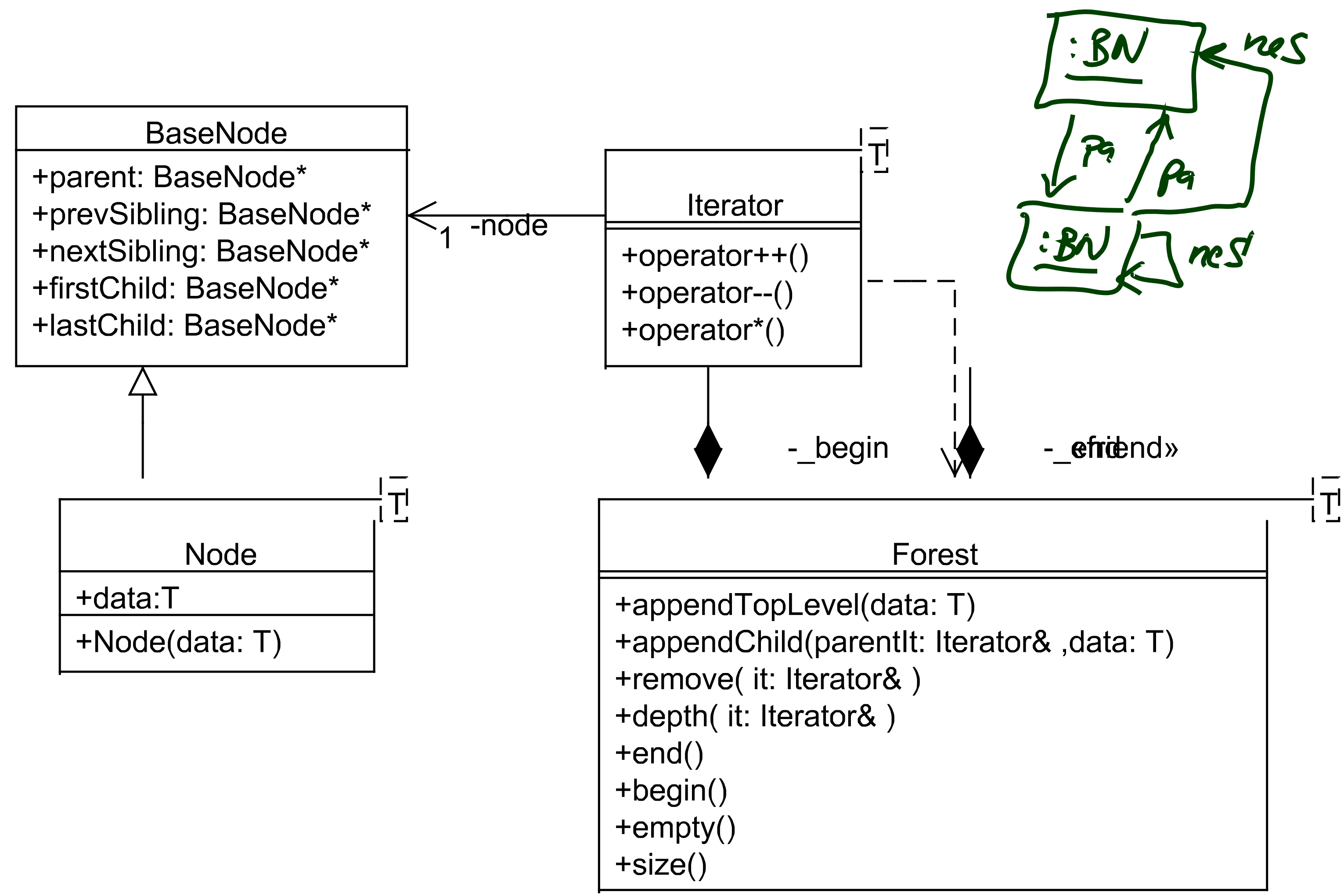


- In the example, we can conclude that the author is referring to **some** signature $\mathscr{S} = (\mathscr{T}, \mathscr{C}, V, atr)$ with at least
  - $\{C, D\} \subseteq \mathscr{C}$ 0,1 also fine, don't know
  - $T \in \mathscr{T}$
  - $\{x : C_*, p : C_*, z : T\} \subseteq V$
  - $\{x\} \subseteq atr(C)$
  - $\{p, z\} \subseteq atr(D)$

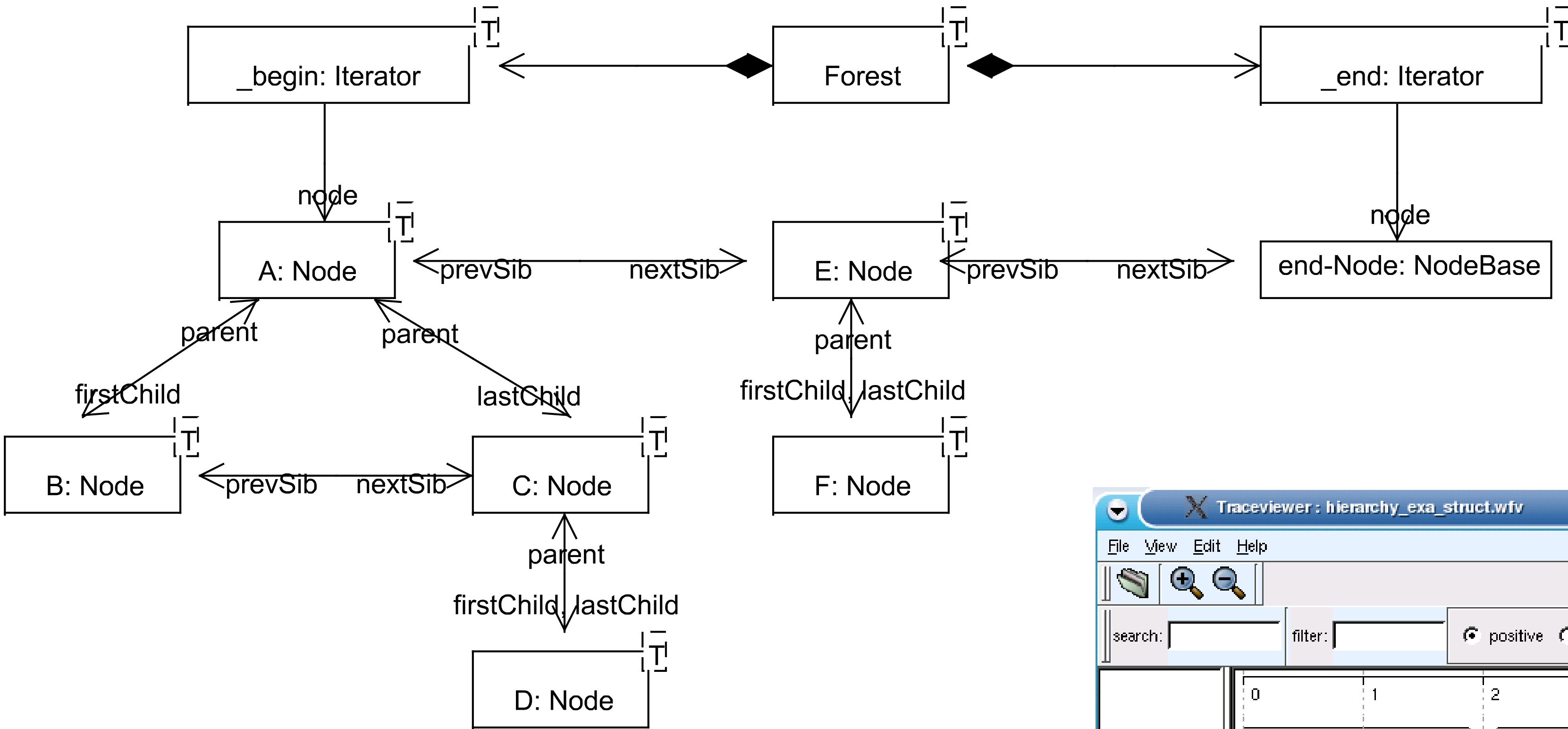
  and a structure with
  - $0 \in \mathscr{D}(T)$
  - $\{u_1, u_2\} \subseteq \mathscr{D}(C)$
  - $\{u_3\} \subseteq \mathscr{D}(D)$

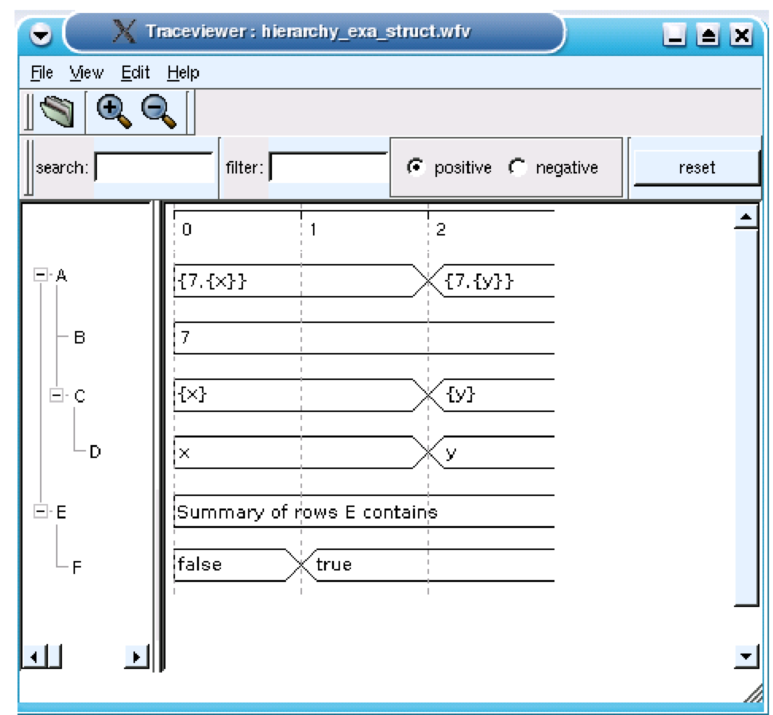
Example: Object Diagrams for Documentation

Example: Data Structure [Schumann et al., 2008]



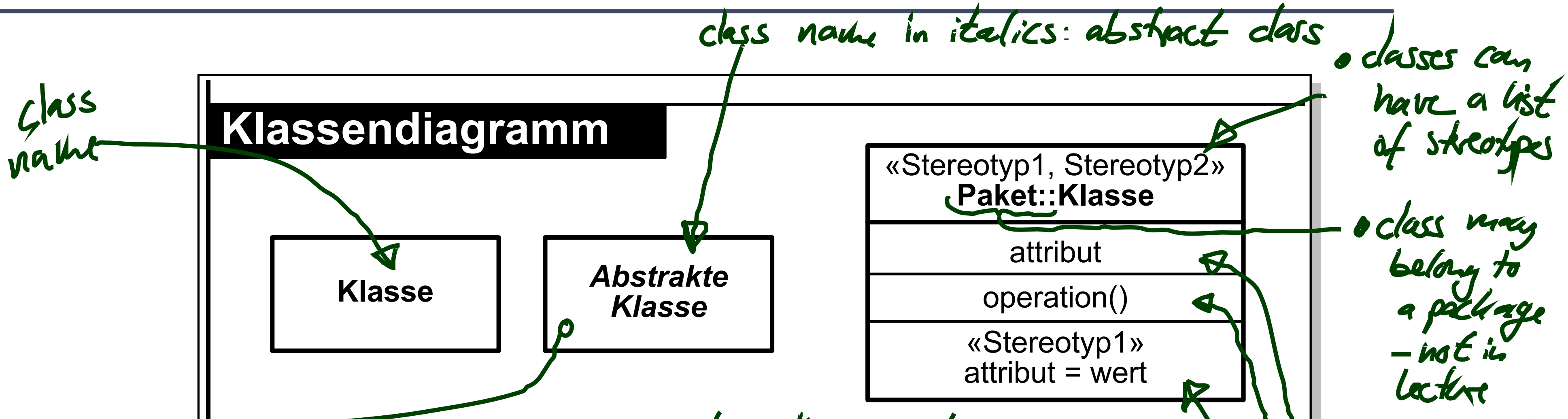
Example: Illustrative Object Diagram [Schumann et al., 2008]





UML Class Diagrams: Stocktaking

UML Class Diagram Syntax [Oestereich, 2006]



Syntax für Attribute:
Sichtbarkeit Attributname : Paket::Typ [Multiplizität Ordnung] = Initialwert {Eigenschaftswerte}
Eigenschaftswerte: {readOnly}, {ordered}, {composite}

Syntax für Operationen:
Sichtbarkeit Operationsname (Parameterliste):Rückgabetyp {Eigenschaftswerte}

Sichtbarkeit:
+ public element
# protected element
– private element
~ package element

Parameterliste: Richtung Name : Typ = Standardwert
Eigenschaftswerte: {query}
Richtung: in, out, inout

only with association

initial value

attribute name

type

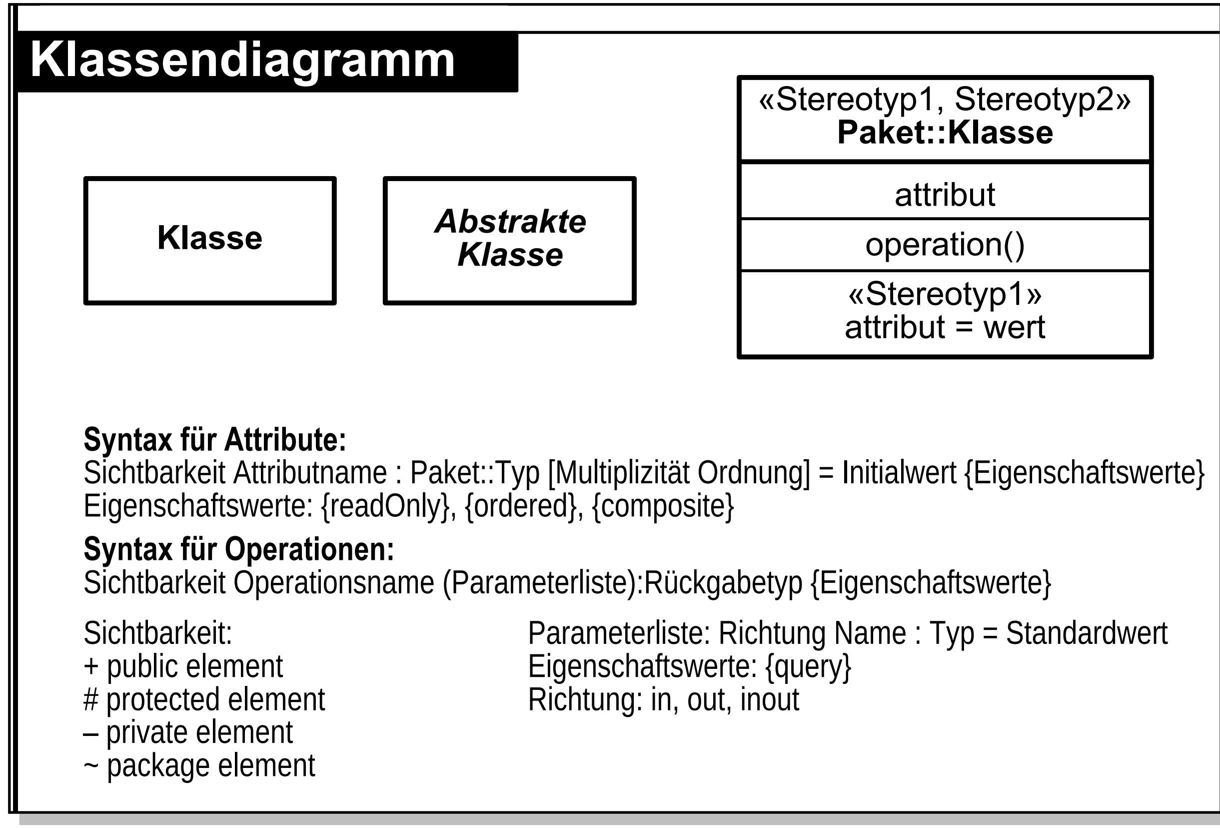
a list of properties

alternative rendering (e.g. for handwriting): Class {A}

What Do We (Have to) Cover?

A **class**

- has a set of **stereotypes**,
- has a **name**,
- belongs to a **package**, (NOT!)
- can be **abstract**,
- can be **active**, e.g. Class
- has a set of **operations**, (Well)
- has a set of **attributes**.



Each **attribute** has

- a **visibility**,
- a **name**, a **type**,
- a **multiplicity**, an **order**, (later) (NOT)
- an **initial value**, and
- a set of **properties**, such as **readOnly**, **ordered**, etc.

Wanted: places in the signature to represent the information from the picture.

Extended Signature

Recall: Signature

$\mathscr{S} = (\mathscr{T}, \mathscr{C}, V, atr)$ where

- (basic) types $\mathscr{T}$ and classes $\mathscr{C}$, (both finite),
- typed attributes V, τ from $\mathscr{T}$ or $C_{0,1}$ or C_*, $C \in \mathscr{C}$,
- $atr : \mathscr{C} \to 2^V$ mapping classes to attributes.

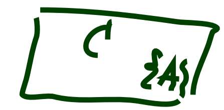
Too abstract to represent class diagram, e.g. no "place" to put class **stereotypes** or attribute **visibility**.

So: **Extend** definition for classes and attributes: Just as attributes already have types, we will assume that

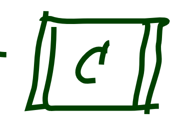
- classes have (among other things) **stereotypes** and
- attributes have (in addition to a type and other things) a **visibility**.

Extended Classes

From now on, we assume that each class $C \in \mathscr{C}$ has:



- a finite (possibly empty) set S_C of **stereotypes**,
- a boolean flag $a \in \mathbb{B}$ indicating whether C is **abstract**, (a=true iff abstract)
- a boolean flag $t \in \mathbb{B}$ indicating whether C is **active**.



We use $S_{\mathscr{C}}$ to denote the set $\bigcup_{C \in \mathscr{C}} S_C$ of stereotypes in $\mathscr{S}$.

(Alternatively, we could add a set St as 5-th component to $\mathscr{S}$ to provides the stereotypes (names of stereotypes) to choose from. But: too unimportant to care.)

Convention:

- We write

$$\langle C, S_C, a, t \rangle \in \mathscr{C}$$

  when we want to refer to all aspects of C.

- If the new aspects are irrelevant (for a given context), we simply write $C \in \mathscr{C}$ i.e. old definitions are still valid.

Extended Attributes

- From now on, we assume that each attribute $v \in V$ has (in addition to the type):
  - a **visibility**

$$\xi \in \{\underbrace{\mathsf{public}}_{:=+}, \underbrace{\mathsf{private}}_{:=-}, \underbrace{\mathsf{protected}}_{:=\#}, \underbrace{\mathsf{package}}_{:=\sim}\}$$

  - an **initial value** $expr_0$ given as a word from **language for initial values**, e.g. OCL expresions.

    (If using Java as **action language** (later) Java expressions would be fine.)
  - a finite (possibly empty) set of **properties** P_v.

    We define $P_{\mathscr{C}}$ analogously to stereotypes.

Convention:

- We write $\langle v : \tau, \xi, expr_0, P_v \rangle \in V$ when we want to refer to all aspects of v.
- Write only $v : \tau$ or v if details are irrelevant.

And?

- **Note**:
  All definitions we have up to now **principally still apply** as they are stated in terms of, e.g., $C \in \mathscr{C}$ — which still has a meaning with the extended view.

  For instance, system states and object diagrams remain mostly unchanged.

- **The other way round**: **most** of the newly added aspects **don't contribute** to the constitution of system states or object diagrams.

- Then what **are** they useful for...?
- First of all, to represent class diagrams.
- And then we'll see.

Mapping UML CDs to Extended Signatures

From Class Boxes to Extended Signatures

A class box n **induces** an (extended) signature class as follows:

n:

$\langle\langle S_1, \dots, S_k \rangle\rangle$ C
$\xi_1\ v_1 : \tau_1 = v_{0,1}\ \{P_{1,1}, \dots, P_{1,m_1}\}$ $\vdots$ $\xi_\ell\ v_\ell : \tau_\ell = v_{0,\ell}\ \{P_{\ell,1}, \dots, P_{\ell,m_\ell}\}$

$$\rightsquigarrow$$

$$\mathscr{C}(n) := \langle C, \{S_1, \dots, S_k\}, a(n), t(n) \rangle$$

$$V(n) := \{\langle v_1 : \tau_1, \xi_1, v_{0,1}, \{P_{1,1}, \dots, P_{1,m_1}\}\rangle, \dots, \langle v_\ell : \tau_\ell, \xi_\ell, v_{0,\ell}, \{P_{\ell,1}, \dots, P_{\ell,m_\ell}\}\rangle\}$$

$$atr(n) := \{C \mapsto \{v_1, \dots, v_\ell\}\}$$

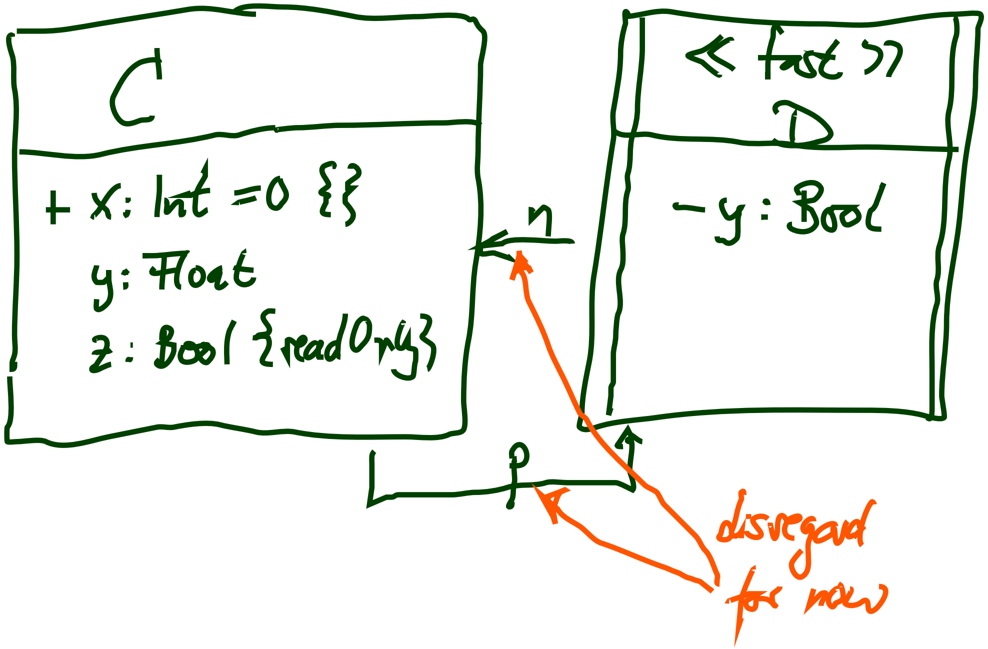
where

- "abstract" is determined by the font:

$$a(n) = \begin{cases} \mathit{true} & \text{, if } n = \boxed{C} \text{ or } n = \boxed{\mathit{C}_{\{A\}}} \\ \mathit{false} & \text{, otherwise} \end{cases}$$

- "active" is determined by the frame:

$$t(n) = \begin{cases} \mathit{true} & \text{, if } n = \boxed{\mathbf{C}} \text{ or } n = \|\, \text{C} \,\| \\ \mathit{false} & \text{, otherwise} \end{cases}$$



$$\mathcal{S} = (\{Int, Float, Bool\},$$
$$\{\langle C, \emptyset, false, false\rangle,$$
$$\langle D, \{fast\}, false, true\rangle\},$$
$$\{\langle x : Int, +, 0, \emptyset\rangle,$$
$$\langle y : Float, +, \star, \emptyset\rangle,$$
$$\langle z : Bool, +, \star, \{readOnly\}\rangle,$$
$$\langle y : Bool, -, \star, \emptyset\rangle$$

[Oestereich, 2006] Oestereich, B. (2006). *Analyse und Design mit UML 2.1, 8. Auflage*. Oldenbourg, 8. edition.

[OMG, 2007a] OMG (2007a). Unified modeling language: Infrastructure, version 2.1.2. Technical Report formal/07-11-04.

[OMG, 2007b] OMG (2007b). Unified modeling language: Superstructure, version 2.1.2. Technical Report formal/07-11-02.

[Schumann et al., 2008] Schumann, M., Steinke, J., Deck, A., and Westphal, B. (2008). Traceviewer technical documentation, version 1.0. Technical report, Carl von Ossietzky Universität Oldenburg und OFFIS.