ALP3

Algorithmen
Datenstrukturen
Datenabstraktion

Freie Universität Berlin

Modellierende Spezifikation von abstrakten Datentypen

Beispiel: Mengen

- Erstellen — `X = new Menge()`
- Einfügen — `boolean X.insert(i)` (i: int)
- Löschen — `boolean X.remove(i)`
- ist enthalten? — `boolean X.contains(i)` (i: int)

Mengen von ganzen Zahlen (vom Typ int)
(der Einfachheit halber.)

Jede Operation ist spezifiziert durch

- Vorbedingungen (V)
- Nachbedingungen (N)

Erstellen:
- Vorbedingungen — (keine)
- Nachbedingungen: $M = \emptyset$

contains(i):
- Vorbedingungen: —
- Nachbedingungen: Ergebnis $=$ $i \in M$

insert(i):
- Vorbedingungen: —
- Nachbedingungen: $M = M^{alt} \cup \{i\}$
  Ergebnis = $i \notin M^{alt}$

GLEICHUNGEN! keine ZUWEISUNGEN M := M ∪ {i}

remove(i):
- Vorbedingungen: — [Variante: $i \in M$]
- Nachbedingungen: $M = M^{alt} \setminus \{i\}$
  Ergebnis = $i \in M^{alt}$

Implementierung eines abstrakten Datentyps

(konkrete) Implementierung	ADT abstraktes Modell
Felder, Listen, Bäume, ...	Mengen, Relationen

Abstraktionsfunktion A

$$A(U,m) = \{U[0], U[1], \ldots, U[m-1]\}$$

int [100] int

$$= \{U[j] \mid 0 \leq j < m\}$$

Menge

[Haskell-Implementierung: Lö 4 (Ei 4 (Ei 2 (Ei 4 Le)))]

Beispiel 2:

Darstellung:
(verkettete) Liste L aus Paaren

$\in \{Ei, Lö\} \times Int$.

(Lö,4), (Ei,4), (Ei,2), (Ei,4)

$A(L) = \{i \in Int \mid i$ kommt in der Liste L vor und das erste Vorkommen ist von der Form $(Ei, i)\}$

Darstellungsinvariante (D)

- Die Werte $U[0], U[1], \ldots, U[m-1]$ sind verschieden.
- $0 \leq m \leq 100$

→ Varianten: • sortiert
• (weglassen)

Nebenbedingungen (Zusatzrestriktionen) (Z)

- $|M| \leq 100$

---

korrekte Implementierung:

$$(V) \wedge (D) \wedge (Z) \xRightarrow{\text{Operation}} (N) \wedge (D)$$

müssen mit Hilfe von A in die Sprache der Implementierung übersetzt werden.

Bsp. remove

(V) —

(D) $0 \leq m^{alt} \leq 100$, $U^{alt}[0], \ldots, U^{alt}[m^{alt}-1]$ versch.

(Z) $|M^{alt}| \leq 100$ $|M^{neu}| \leq 100$ $\xrightarrow{A}$ $m^{alt} \leq 100$, $m^{neu} \leq 100$

(N) $M^{neu} = M^{alt} \setminus \{i\}$ $\xrightarrow{A}$ $\{U[0], \ldots, U[m-1]\} =$

Ergebnis: $i \in M^{alt}$. $\{U^{alt}[0], \ldots, U^{alt}[m^{alt}-1]\} \setminus \{i\}$

(D) $0 \leq m \leq 100$ ✓ ✓, $U[0], \ldots, U[m-1]$ verschieden.

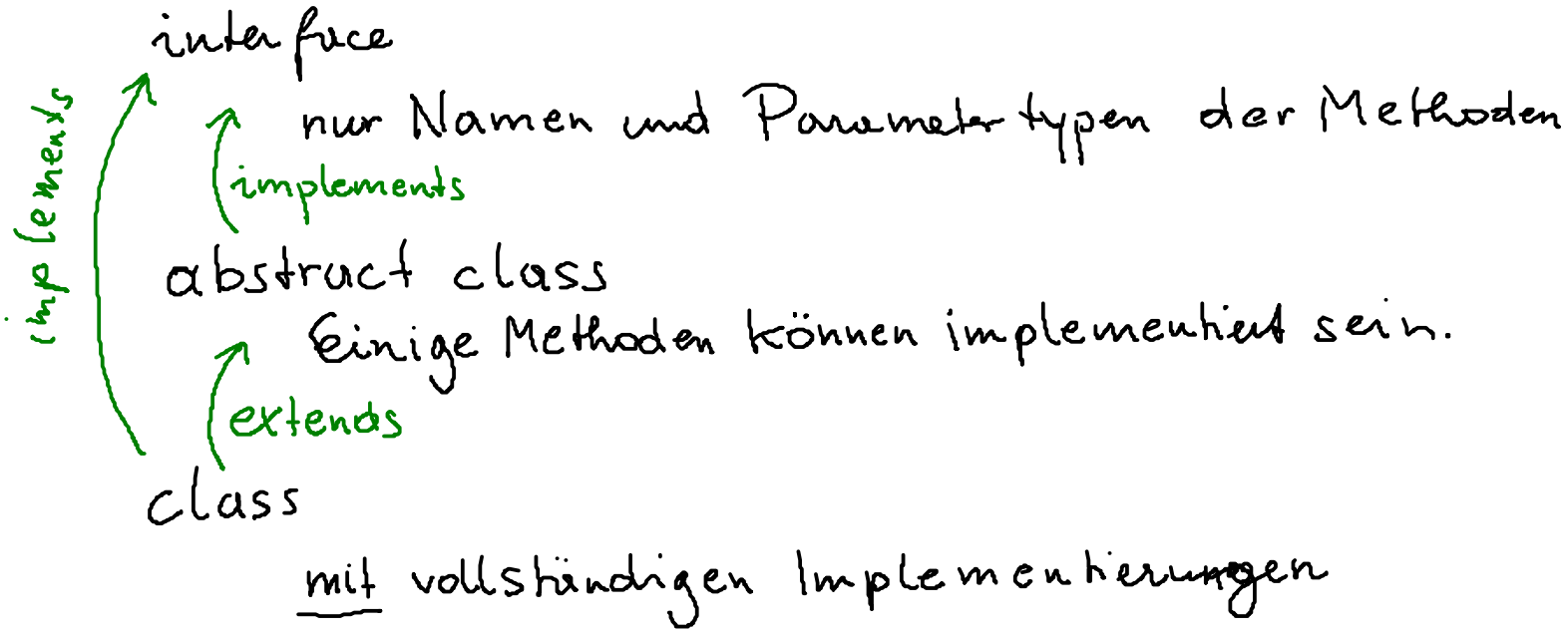
Anwendung auf höherer Ebene.

{Vorbedingung} $U[k-1] := U[k];$ {Nachbedingung}

(Hoare-Kalkül)

$\{(V) \wedge (Z)\}$ M.remove(i); $\{(N)\}$

$M := M \setminus \{i\};$



mit vollständigen Implementierungen

Eifel. Vorbedingungen, Nachbedingungen

„Design by contract"