Objektno programiranje

Ponovimo: $A \leq B$ podtip

$$\frac{e : A \qquad A \leq B}{e : B}$$

Zapisi: $\{l_1 : \tau_1, \ldots, l_n : \tau_n\}$ tip zapisa $\tau_1 \times \cdots \times \tau_n$

$\{l_1 = e_1, \ldots, l_n = e_n\}$

Ocaml: record types

SQL: tip zapisa = shema tabele
polje = stolpec
zapis = vrstica

Ocaml: moduli → zapisi "en nivo višje"

signatura = tip zapisa

```
sig
  type t
  val empty : t
  val add : t -> t -> t
end
```

$\{t : Type, empty : t, add : t \to t \to t\}$

struktura/modul = zapis

```
struct
  type t = int
  let empty = 0
  let add x y = x+2*y
```

$\{t = int, empty = 0, add = fun\ x\ y \to x + 2*y\}$

Strukturni podtipi zapis:

- po širini $\{l_i : \tau_i\}_{i=1}^{n} \leq \{k_j : \rho_j\}_{j=1}^{m}$
  (A ≤ B)
  - vsako polje iz B se pojavi tudi v A (z istim tipom)
  - A je "širši" kot B

  ↘ vsak $k_j : \rho_j$ se pojavi kot $l_i : \tau_i$ da $k_j = l_i$ in $\rho_j = \tau_i$

- po globini & širini

  uporabimo relacijo $\leq$ tudi na tipih polj

$$\underbrace{\{l_i : \tau_i\}_{i=1}^{n}}_{A} \leq \underbrace{\{k_j : \rho_j\}_{j=1}^{m}}_{B}$$

Vsako polje $k_j : \rho_j$ se pojavi v A kot polje $l_i : \tau_i$, da velja

$$k_j = l_i \quad \text{in} \quad \tau_i \leq \rho_j$$

OCaml:

- ne uporablja podtipov za zapise
- jih uporablja za signature & strukture

Objekti

objekt this = { $l_1 = v_1$, $l_2 = v_2$, ⋮, $l_n = v_n$ } } atributi / polja

$m_1 =$ this.l_1

$m_2 =$ ... this.m_1(42) } metode

}

this = { $l_1 = v_1, \ldots, l_n = v_n$,

$m_1 = f_1(\text{this})$, $m_2 = f_2(\text{this})$ }

This je negibna točka,

objekt = rekurzivni zapis

(ker se sklicuje sam nase z this)

Razredi

Razred:
- specifikacija atributov (kateri so in kakšne tipe imajo)
- metode

Strukturni in nominalni podtipi

strukturni: glede na polja in njihove tipe

nominalni: glede na poimenovanje tipov/razredov

Ocaml

```
type a = {x : float; y : float}
type b = {x : float}
type c = {x : float; z : float}

a ≤ b
c ≤ b
```

} Strukturno

Java

```
class B { float x; }
class A extends B { float y; }
class C { float x; float z; }
```

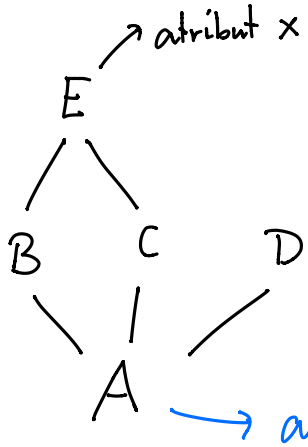
$A \leq B$ ker je tako odrjeno z "extends"

$C \not\leq B$ ker ni tako odrjeno (struktura ni pomembna)

Enkratno & večkratno dedovanje

class A extends B — enkratno

class A extends B, C, D — večkratno



→ ali podeduje x enkrat ali dvakrat?

Prekrivanje

```
class A {

    ... int f(int x){....}

}
```

```
class B extends A {
    :
    int f(int x) {....}
```

prekrije f

this.f