Objektno programiranje

13. maj 2020

Objekti

Zapis (record): $\{x=3; y="foo"\} : \{x:\text{int}; y:\text{string}\}$

int x string

Podtipi zapisov:

- vrstni red polj ni pomemben
- po širini: $\{x:\text{int}; y:\text{string}; t:\text{bool}\} \leq \{x:\text{int}; y:\text{string}\}$
- po globini: če $A \leq B$ potem $\{x:\text{int}; y:A\} \leq \{x:\text{int}; y:B\}$

OCaml ne uporablja podtipov za zapise.

Objekt je zapis, ki se lahko sklice sam nase (npr. Java "this"):

zapis: $\{x=3; y=5; \text{get\_x} = 3\}$?

objekt: $\text{this} = \{x=3; y=5; \text{get\_x} = \text{this}.x\}$ rekurzivni zapis

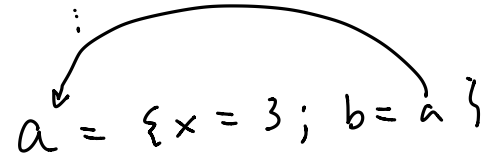
$a = \{x=3; b=a\} =$

$\{x=3; b=\{x=3; b=\{x=3; \cdots\}\}\}$

rekurzivni zapis (koinduktiven)

$a.x$
$a.b.x$
$a.b.b.x$
$\vdots$

$a.b.b = a$
$a.b = a$



Objekt = rekurzivni zapis

11.

2.

objekt = enkapsulirano stanje

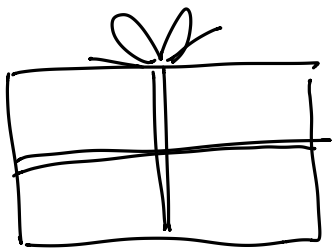
Object

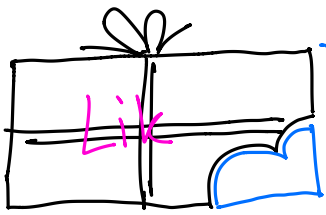
enkapsulirano stanje { atributi → ga ne spreminjamo
↳ stanje lahko spreminjamo

funkcionalnost { metode

end

Abstraktne metode & razredi

class C  funkcionalnost

class C 

manjkajoča funkcionalnost

Primer: Geometrijski liki

```
abstract
class Lik {
    private Color c;          barva
    private int x0, y0;       izhodišče

    public void move(int dx, int dy) { x0 += dx; y0 += dy; }

    abstract
    public float ploscina() { ..... }      ??
}
```

NE VEMO! ODVISNO OD PRIMERA;

~~return 0;~~ 42

throw ErrorAreaNotKnown;

```
class Krog extends Lik {
        private float polmer;
                :
        public float ploscina() { return π * polmer^2; }
}
```

Hierarhija podtipov

① Strukturno $A \leq B$ če to določata strukturi A in B

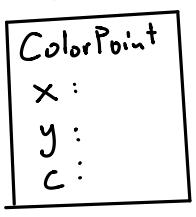
"duck typing"

② Nominalno: določa jo programer s podrazredi

public class A extends B { ..... }

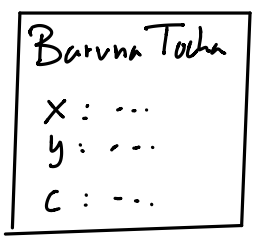
Point { x y }

ColorPoint extends Point { c } ⇝



ga lahko uporabimo kot Point

class BarvnaTocka { x y c }



← ga ne moremo uporabiti kot Point