

# Haskell in razredi tipov

**prejšnjič...**

# **Fleksibilnost tipov** lahko zagotovimo na več načinov

## parametrični polimorfizem

**generiki** (Java, Rust), **polimorfizem** (OCaml, Haskell), ...

## ad-hoc polimorfizem

**značilnosti**/*traits* (Rust), **razredi tipov** (Haskell), ...

## podtipi

implicitne **pretvorbe**, **moduli** (OCaml), **podrazredi**, ...

Vrednosti **manjšega** tipa so kompatibilne z **večjim**

$$\frac{\Gamma \vdash e : A \quad A \leq B}{\Gamma \vdash e : B}$$

Spoznali smo **pravila za določanje** podtipov

$$\frac{A_1 \leq B_1 \quad A_2 \leq B_2}{A_1 \times A_2 \leq B_1 \times B_2}$$

$$\frac{B_1 \leq A_1 \quad A_2 \leq B_2}{A_1 \rightarrow A_2 \leq B_1 \rightarrow B_2}$$

**Zapisni tipi** imajo bogato strukturo podtipov

$$\forall i \leq m. \exists j \leq n. \ell'_i = \ell_j \wedge A_j \leq B_i$$

---

$$\{\ell_1 : A_1; \dots; \ell_n : A_n\} \leq \{\ell'_1 : B_1; \dots; \ell'_m : B'_m\}$$

# Spremenljivi zapisi podtipov ne podpirajo

$$A = \{\text{mutable } x : \text{int}\}$$
$$B = \{\text{mutable } x : \text{float}\}$$

## branje polja

$$\text{let } r_A(p : A) = p.x + 10$$
$$r_A(\{x : 3.14\})$$

$$\text{let } r_B(p : B) = p.x + 3.14$$
$$r_B(\{x : 10\})$$


## pisanje v polje

$$\text{let } w_A(p : A) = p.x \leftarrow 10$$
$$w_A(\{x : 3.14\})$$

$$\text{let } w_B(p : B) = p.x \leftarrow 3.14$$
$$w_B(\{x : 10\})$$


# Objekti so bolj ali manj rekurzivni zapisi

**konstruktorji & metode**

funkcije

**dedovanje**

nominalni podtipi

**prekrivanje** (*overriding*)

senčenje

**preobteževanje** (*overloading*)

ad-hoc polimorfizem

**generične metode & razredi**

parametrični polimorfizem

**abstraktne/virtualne metode & razredi**

specifikacije

**vmesniki**

specifikacije

**enkapsulacija**

modularnost & abstrakcija



# Pri razredih imamo **dve pristopa** k podtipom

```
class C { int x; }  
class D { int x; float y; }  
class E extends C { float y; }
```

## nominalni pristop

Podrazredi so tisti, ki jih programer **navede**.

$E \leq C$       $D \not\leq E$

## strukturni pristop

Podrazredi so vsi s **kompatibilno strukturo**.

$E \leq C$       $D \leq E$       $E \leq D$





nominalni in strukturni

podtipi

v OCamlu



# Uvod v Haskell



# Haskell



# Razredi tipov

**prihodnjič...**

Posvetili se bomo **računskim učinkom**

program

=

funkcija + **učinki**

# Spoznali bomo **monade**

```
T :: Type -> Type  
return :: a -> T a  
>>= :: T a -> (a -> T b) -> T b
```