24/25-1 Nominal vs Structural Typing

```
f(Shape s) {
    s.bounding();
}
```

```
f( r )
f( p )
```

Nominal

r and p must be Shapes

Structural

r and p must support 'bounding'

Nominal is catching some errors

nominal is faster

g : int -> int

f(x) {x.g(17)}

"new type"

---

int ∪ str

int ∩ str

+ that on numbers, adds them
on strings, appends them

X + : (int ∪ str) (int ∪ str) -> (int ∪ str)

X + : (int int -> int) ∪ (str str -> str)

+ : ∩

---

subtyping relation

nats ⊆ integers ⊆ rationals ⊆ reals

int fixed point floating point

Cutecats ⊆ Cats ⊂ Pets ⊆ Mammals

.playpiano .meow .pet .hp

.y.a.

25/25-2

S <: T means S is a sub-type of T

S <: (S ∪ T) (S ∩ T) <: S S <: T > and T <: S and S ≠ T

T <: (S ∪ T) (S ∩ T) <: T

Cat <: Mammal

→ any code that wants a mammal will take a cat

→ not inverse (ie some cat specific could exist)

→ any code that claims to return a mammal might/can ret a cat

Hypo 1: ∀ m ∈ M, f(m) has no error

2: C <: M

conclusion: ∀ c ∈ C, f(c) has no error

Liskov substitution Principle

f : Df → Rf Cat → Cat

g : Dg → Rg Mam → Mam

Dg <: Df covariance

Df ≮: Dg Rf <: Rg

f <: g

g

m(somefun : Mam → Mam) {

somefun(new Chin(chin()) . hp()

}

f

m(fun (c: Cat) { c.meow(); c })

26/25-3

type-of : expr × Env → type

subtype : type × type → bool, or type

$$\frac{\Gamma \vdash f : D \Rightarrow R \qquad \Gamma \vdash a : D}{\Gamma \vdash (f\ a) : R}$$

1) ~~del~~ replace $\Gamma \vdash a : D$ with $\Gamma \vdash a : D'$
   add $D' <: D$

2) add a new rule $$\frac{\Gamma \vdash e : S \qquad S <: T}{\Gamma \vdash e : T}$$

---

object = name + a hierarchy (nominal)

object = map (name → type)

$M = \{ m_0 \mapsto S_0, \cdots, m_j \mapsto S_j \}$

$<:$

$N = \{ n_0 \mapsto T_0, \cdots, n_k \mapsto T_k \}$

constraints: $j \geq k$, $m_i = n_i$ (width-subtyping)

$S_i <: T_i$ (depth sub-typing)

posn : { x ↦ int, y ↦ int, dist ↦ ~~(⇒int)~~, translate ↦ (posn ⇒ ()) }

A (struct person (name eyes))

~~(define (f x) (p x "b"))~~
(f J) == (f J)? YES

B (define (f x)
    (struct p (n e))
    (p x "blue"))

(f "Jay") == (f "Jay")?
NO

7/25-4

What is f's type?

A) f : (str → person)

B) f : str → ~~person~~

f : ∃p . str → p

(∃x. x → int)

class constructor:
(initialize args) →

∃O. O