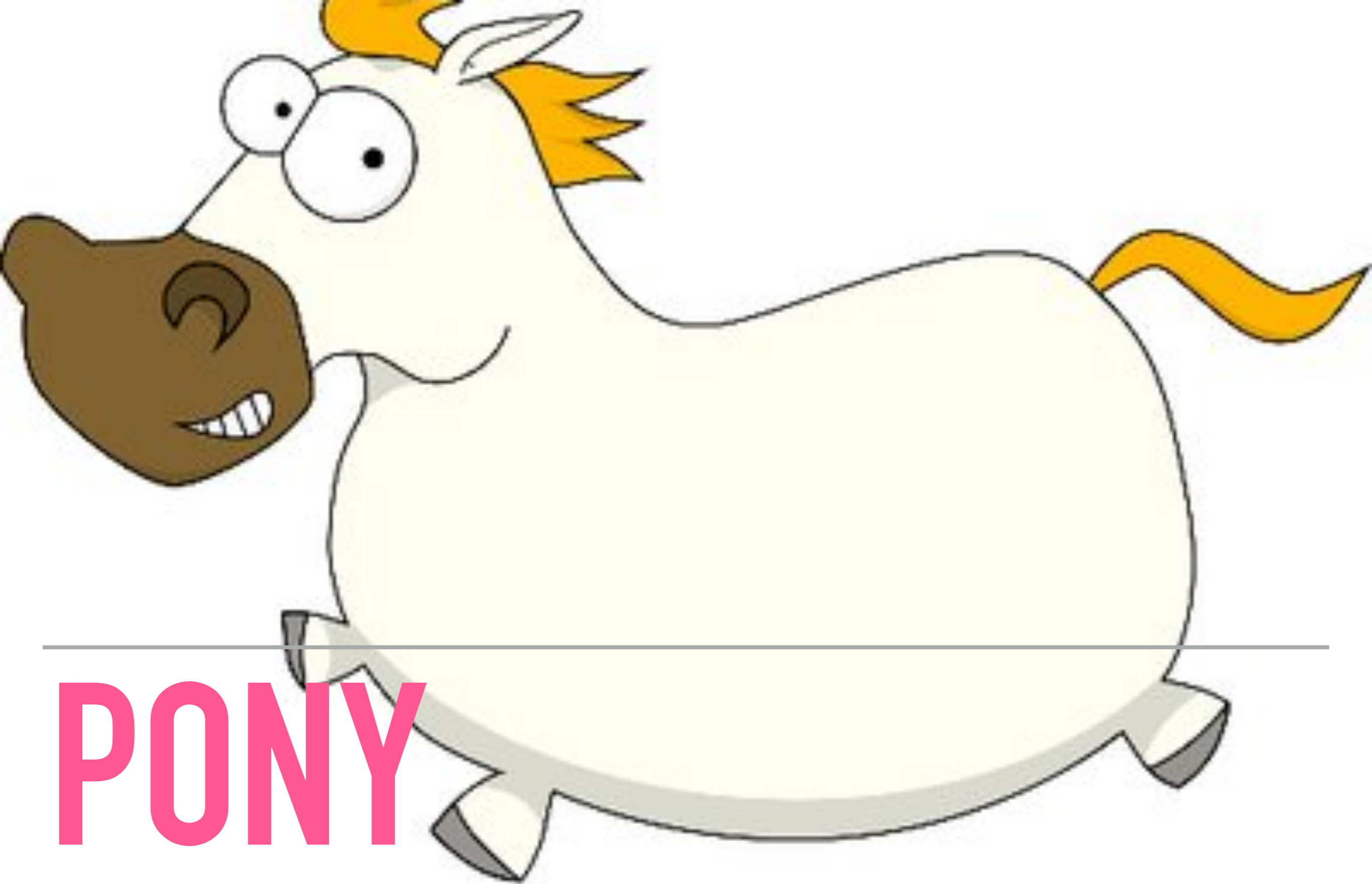

PONY



PONY

PRESEDENS

▶ $1+2*3=?$

▶ $(1+2)*3=(3)*3=9$

▶ $1+(2*3)=1+(6)=7$

PRESEDENS

- ▶ $10 \text{ xor } 3 \text{ or } 2 = ?$
- ▶ $((10 \text{ xor } 3) \text{ or } 2) = 11?$
- ▶ $(10 \text{ xor } (3 \text{ or } 2)) = 9?$
- ▶ Operator precedence is not supported. Parentheses required.

PRESEDENS

- ▶ Operator precedence is not supported. Parentheses required.
- ▶ Også for $1+2*3$

TEXT

MENS VI SNAKKER OM MATTE...

MENS VI SNAKKER OM MATTE...

- ▶ Dele på null er tull, der tull == 0

BUILDER PATTERN

```
let mann = Person
mann.gi_navn("Snorval")
mann.sett_kaker(5)
mann.sett_briller(2)
```

```
fun ref sett_kaker(kaker: U32) : U32 =>
    _kaker = kaker
```

BUILDER PATTERN

```
let mann = Person
mann.gi_navn("Snorval")
mann.sett_kaker(5)
mann.sett_briller(2)
```

```
let mann = Person
    .gi_navn("Snorval")
    .sett_kaker(5)
    .sett_briller(2)
```

```
fun ref sett_kaker(kaker: U32) : U32 =>
    _kaker = kaker
```

BUILDER PATTERN

```
let mann = Person
mann.gi_navn("Snorval")
mann.sett_kaker(5)
mann.sett_briller(2)
```

```
let mann = Person
    .gi_navn("Snorval")
    .sett_kaker(5)
    .sett_briller(2)
```

```
fun ref sett_kaker(kaker: U32) : Person =>
    _kaker = kaker
    this
```

BUILDER PATTERN

chaining operator



```
let mann = Person
mann.gi_navn("Snorval")
mann.sett_kaker(5)
mann.sett_briller(2)
```

```
let mann = Person
  .>gi_navn("Snorval")
  .>sett_kaker(5)
  .>sett_briller(2)
```

```
fun ref sett_kaker(kaker: U32) : U32 =>
  _kaker = kaker
```

TEXT

INTERFACES

INTERFACES

- ▶ Rust has trait system
- ▶ Go has “duck typed” interfaces.

```
trait HasName {  
    fn name(&self) -> &'static str;  
}  
  
struct Person {  
    name: &'static str;  
}  
  
impl HasName for Person {  
    fn name(&self) -> &'static str {  
        name  
    }  
}
```

```
type hasName interface {  
    name() string  
}  
  
type person struct {  
    name string  
}  
  
func (p person) name() string {  
    return name  
}
```

INTERFACES

- ▶ Pony har begge:
- ▶ nominal subtyping (ala rust)
- ▶ structural subtyping (ala go)

```
trait HasNameT
  fun name(): String

class PersonT is HasNameT
  var _name : String = ""
  fun name(): String => _name
```

```
interface HasNameI
  fun name(): String

class PersonI
  var _name : String = ""
  fun name(): String => _name
```

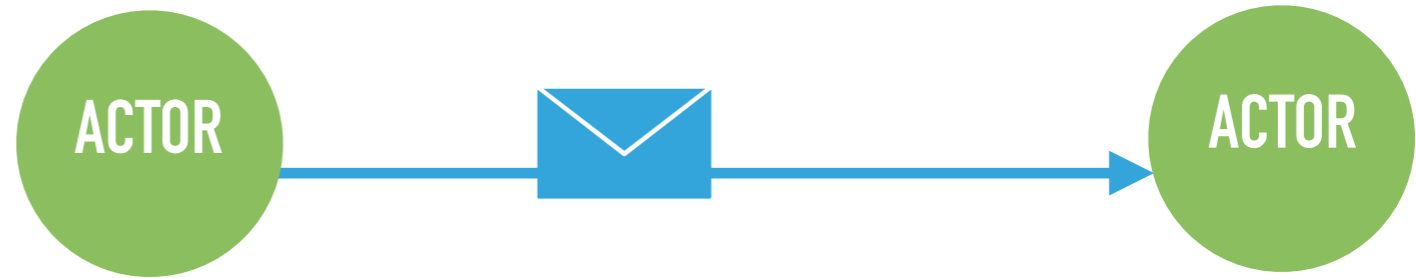
SIGNFIKANTE NAVNEREGLER

- ▶ Typer skrives med stor forbokstav: `class Foo`
- ▶ Variabler skrives med liten forbokstav: `var brille`
- ▶ Private medlemmer begynner med `_`: `var _kake`

TEXT

ACTOR

ACTOR



- ▶ Kommuniserer med meldinger

ACTOR



- ▶ Kommuniserer med meldinger
- ▶ En FIFO-postkasse for hver skuespiller

ACTOR



- ▶ Kommuniserer med meldinger
- ▶ En FIFO-postkasse for hver skuespiller
- ▶ “Behaviours” mottar meldinger og kjører sekvensielt og asynkront. Kun en om gangen
- ▶ Ikke en tråd! Jobber blir fordelt på de trådene som blir allokert ved oppstart (typisk én per hyperkjerne)

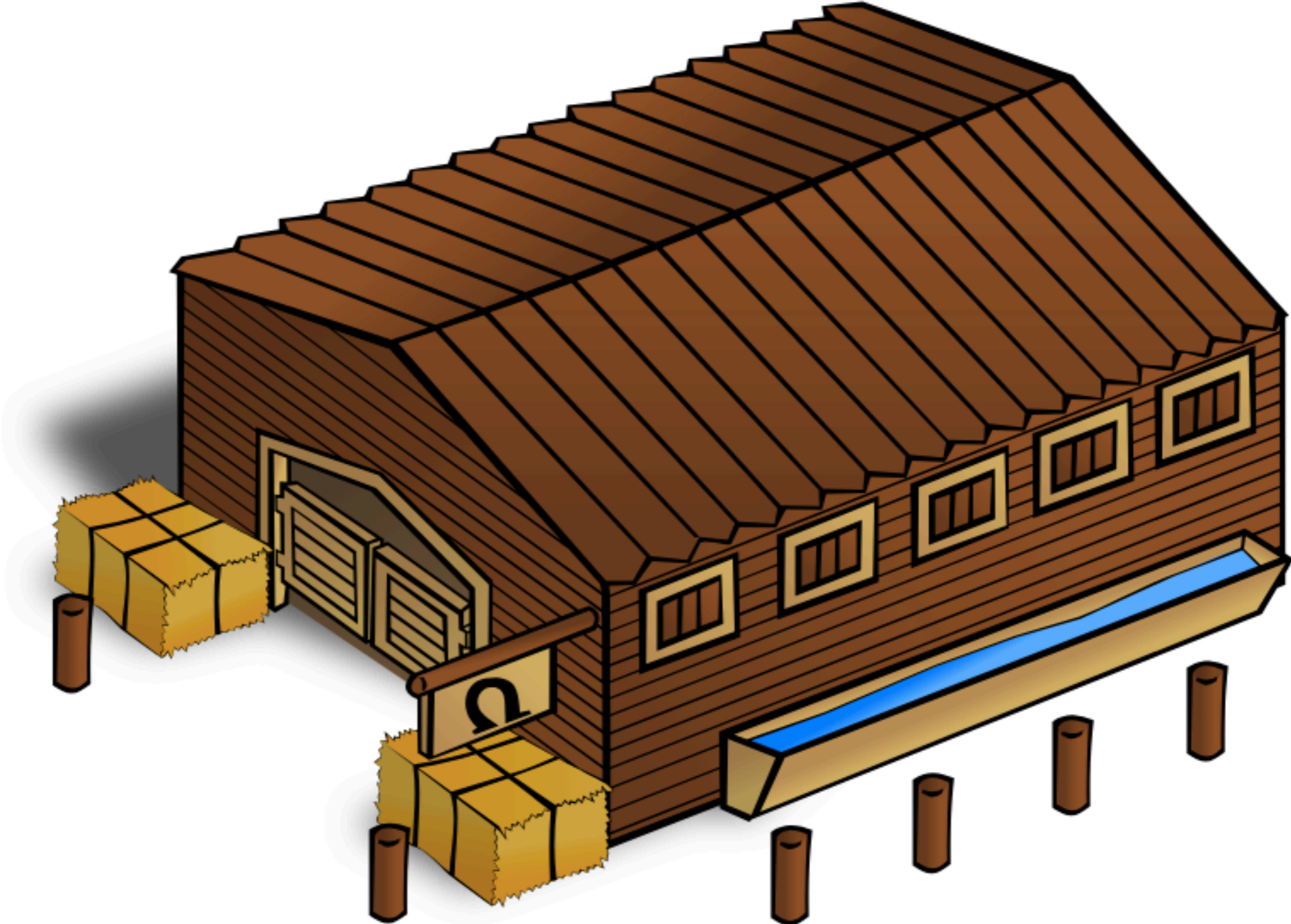
SUMTYPER

- ▶ type Foo is (Hatt | Brille | Mopp)
- ▶ type KanskjeTall is (U64 | None)

TEXT

STABLE

STABLE



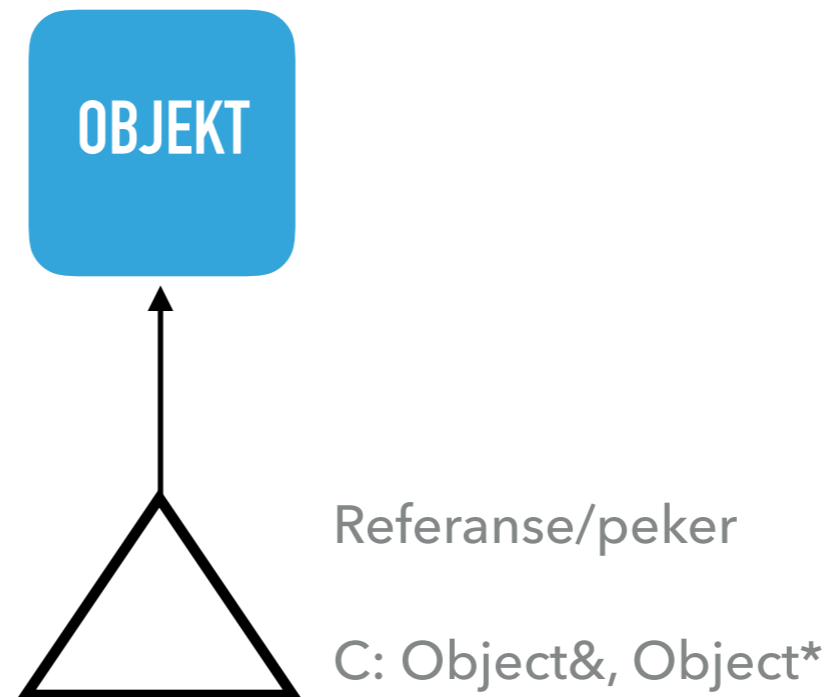
TEXT

LANGUAGE STRANGENESS BUDGET

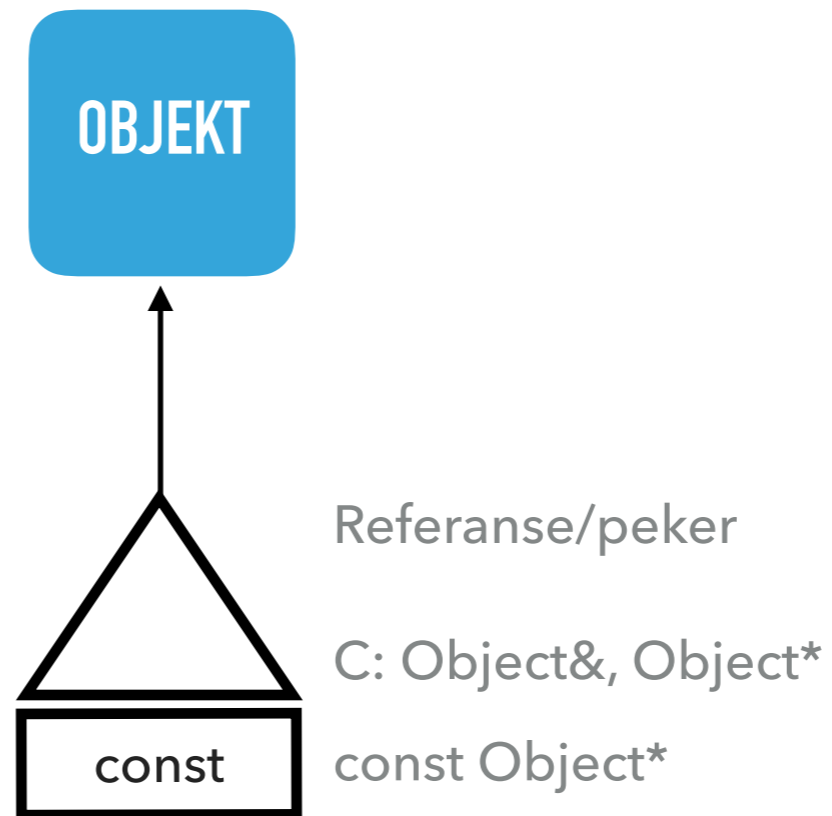
LANGUAGE STRANGENESS BUDGET

- ▶ Brukt opp.

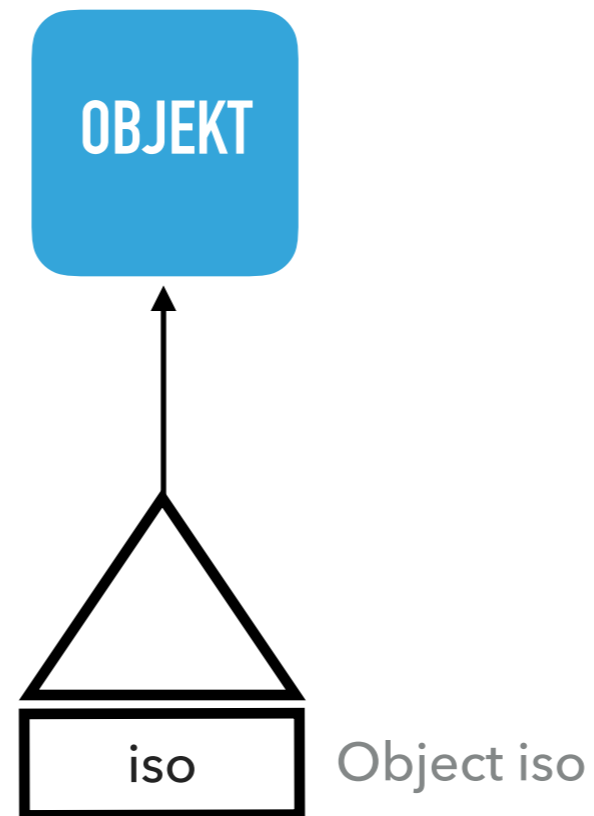
REFERENCE CAPABILITIES



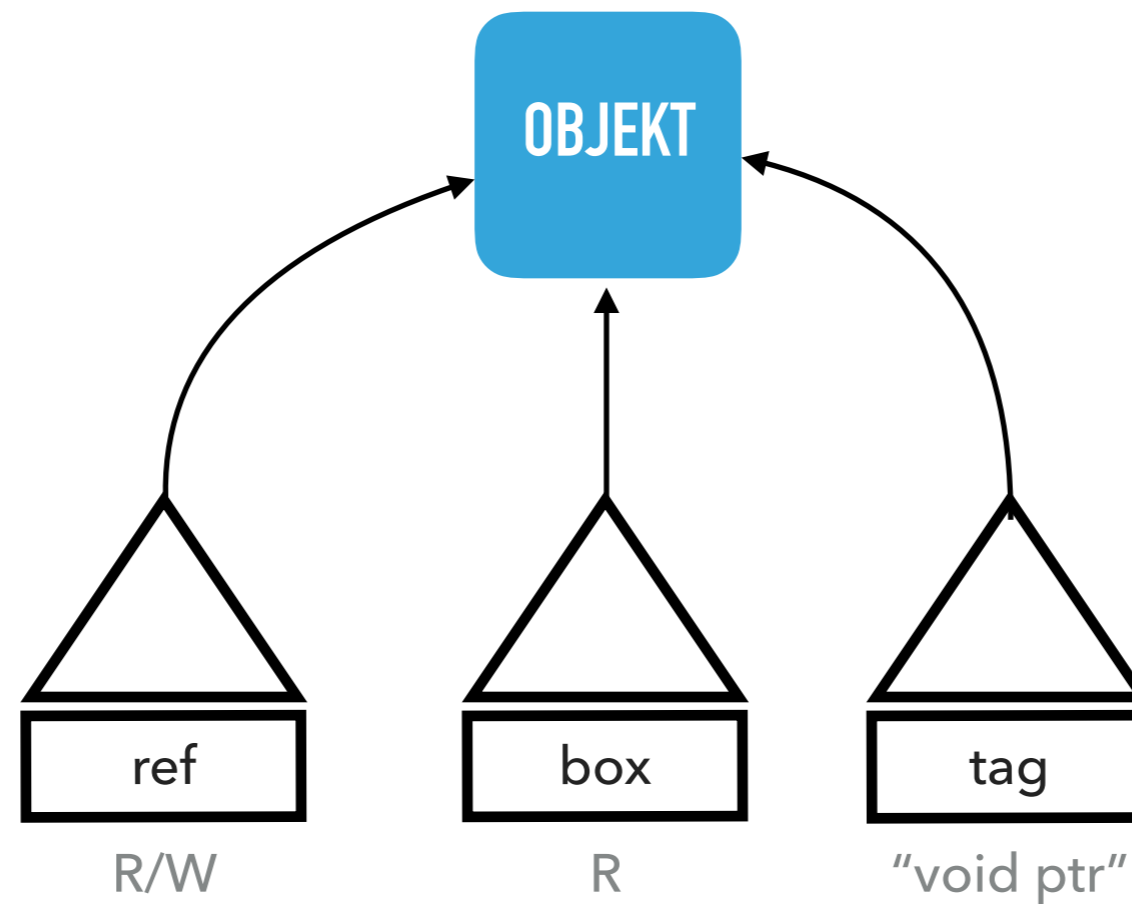
REFERENCE CAPABILITIES



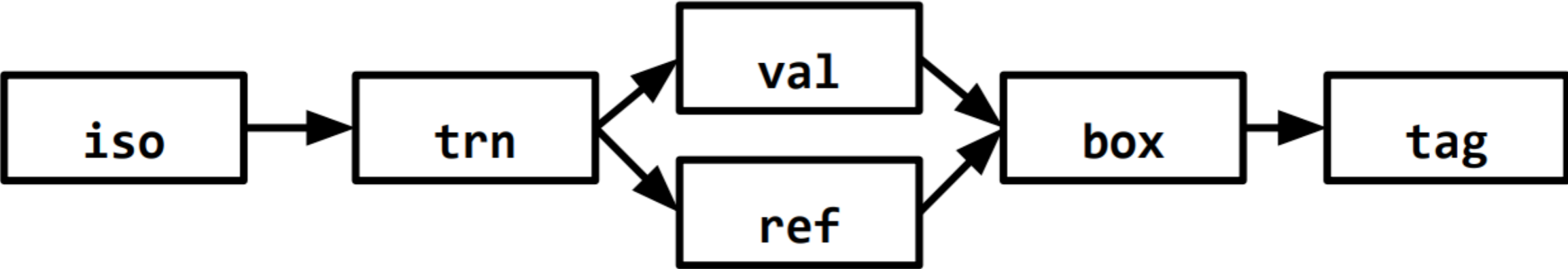
REFERENCE CAPABILITIES



REFERENCE CAPABILITIES



REFERENCE CAPABILITIES



KONKLUSJON

- ▶ Pony går sine egne veier: Presedens, deling på null, signifikante navneregler
- ▶ Chaining operator
- ▶ Actors
- ▶ Reference capabilities

TEXT

TAKK

- ▶ Takk
- ▶ for
- ▶ meg