Functional Programming

Classes

Overview

Classes

```
instance + String
where
      (+) s1 s2 = s1 +++ s2

Start = "Hello" + " world!" // "Hello world!"

instance + (a,b) | + a & + b
where
      (+) (x1,y1) (x2,y2) = (x1+x2,y1+y2)

Start = (1,2) + (3,4) // (4,6)
```

Classes

```
//in StdTuple.dcl
instance == (a,b)     | Eq a & Eq b
instance == (a,b,c) | Eq a & Eq b & Eq c

//in StdTuple.icl
instance == (a,b) | Eq a & Eq b
where
(==) ::!(a,b) !(a,b) → Bool | Eq a & Eq b
(==) (x1,y1) (x2,y2) = x1==x2 && y1==y2

instance == (a,b,c) | Eq a & Eq b & Eq c
where
(==) ::!(a,b,c) !(a,b,c) → Bool | Eq a & Eq b & Eq c
(==) (x1,y1,z1) (x2,y2,z2) = x1==x2 && y1==y2 && z1==z2

Start = (1,2) == (3,4) // False == overloading
```

Classes

```
increment n = n+1

Start = increment 4

double :: a → a | + a
double x = x + x

Start = double 3

Start = double 3.3
```

Classes

```
delta :: a a a → a | *,-,fromInt a
delta a b c = b*b - (fromInt 4)*a*c

Start = delta 1.0 2.0 1.0

class Delta a | *,-,fromInt a

delta1 :: a a a → a | Delta a
delta1 a b c = b*b - (fromInt 4)*a*c

Start = delta1 1.0 2.0 1.0
```

Classes

```
class PlusMinx a
 where
       (+¬)  infixl 6   :: !a   !a      →       a
       (-¬)  infixl 6   :: !a   !a      →       a
       zerox            :: a

instance PlusMinx Char
 where
       (+¬) :: !Char !Char → Char
       (+¬) x y =  toChar (toInt(x) + toInt(y))
       (-¬) x y =  toChar (toInt(x) - toInt(y))
       zerox = toChar 0

Start = 'a' +¬ 'e'
```

Classes

```
Start :: Char
Start = zerox

double1 :: a → a | PlusMin a
double1 x = x + x

Start = double1 2 // 4
```

Classes

```
:: Q = { nom :: Int
       , den :: Int
       }

simplify {nom=n,den=d}
  | d == 0 = abort " denominator is 0"
  | d < 0  = { nom = -n/g, den = -d/g}
  | otherwise =  { nom = n/g, den = d/g}
  where g = gcdm n d

gcdm x y = gcdnat (abs x) (abs y)
  where gcdnat x 0 = x
        gcdnat x y = gcdnat y (x rem y)

mkQ n d = simplify { nom = n, den = d }
Start = mkQ 81 90
```

Classes

```
instance + Q
where
    (+) x y = mkQ (x.nom*y.den+y.nom*x.den) (x.den*y.den)

Start = mkQ 2 4 + mkQ 5 6 // (Q 4 3)

instance - Q
where
    (-) x y = mkQ (x.nom*y.den-y.nom*x.den) (x.den*y.den)

Start = mkQ 2 4 - mkQ 5 6 // (Q -1 3)
```

Classes

```
instance fromInt Q
where
    fromInt i = mkQ i 1

Start :: Q
Start = fromInt 3 // (Q 4 3)

instance zero Q
where
    zero = fromInt 0

Start :: Q
Start = zero // (Q 0 1)
```

Classes

```
instance one Q
where
    one = fromInt 1 //
Start :: Q
Start = one // (Q 1 1)

instance toString Q
where
    toString q
        | xq.den == 1 = toString xq.nom
        | otherwise = toString xq.nom +++"/"+++ toString xq.den
    where xq = simplify q

Start = toString (mkQ 3 4) // "3/4"
```

Classes

```
instance < Q
where
    (<) x y = x.nom*y.den < y.nom*x.den

Start = mkQ 1 2 < mkQ 3 4  // True

ls = [toString q \\ q ← [zero, mkQ 1 3 .. mkQ 3 2]]
Start :: [String]
Start = ls // ["0","1/3","2/3","1","4/3"]
```

Classes

```
//overloading can not be solved
Start = toString zero+zero

Start :: String
Start = toString sum  // "0"
where sum :: Q
      sum = zero + zero
```