The Brave New World of Haskell Type Classes

František Farka

May 24, 2015

```haskell
-- | Computes the smaller of two elements
-- min True False ⟶ False
-- min 3    5     ⟶ 3

min a b = _
```

```
-- | Computes the smaller of two elements
-- min True False ⟶ False
-- min 3    5     ⟶ 3
min ::           a → a → a
min a b = _
```

```haskell
-- | Computes the smaller of two elements
-- min True False ⟶ False
-- min 3    5     ⟶ 3
min ::(Ord a) ⇒ a → a → a
min a b = _
```

```haskell
-- | Linear order

class           Ord a where
    (<) ::  a -> a -> Bool
```

```haskell
-- | Computes the smaller of two elements
-- min True False ⟶ False
-- min 3    5     ⟶ 3
min ::(Ord a) ⇒ a → a → a
min a b = _
```

```haskell
-- | Linear order

class           Ord a where
    (<) ::  a -> a -> Bool
```

```haskell
instance Ord Bool where
    False < True = True
    _     < _    = False
```

```haskell
-- | Computes the smaller of two elements
-- min True False ⟶ False
-- min 3    5     ⟶ 3
min ::(Ord a) ⇒ a → a → a
min a b =  if a < b then a else b
```

```haskell
-- | Linear order

class           Ord a where
    (<) ::  a -> a -> Bool
```

```haskell
instance Ord Bool where
    False < True = True
    _     < _    = False
```

```haskell
-- | Computes the smaller of two elements
-- min True False ⟶ False
-- min 3    5     ⟶ 3
min ::(Ord a) ⇒ a → a → a
min a b =  if a < b then a else b
```

```haskell
-- | Linear order

class            Ord a where
    (<) ::  a -> a -> Bool
    (≤) ::  a -> a -> Bool
```

```haskell
instance Ord Bool where
    False < True = True
    _     < _    = False
```

```haskell
-- | Computes the smaller of two elements
-- min True False ⟶ False
-- min 3    5     ⟶ 3
min ::(Ord a) ⇒ a → a → a
min a b =  if a < b then a else b

-- | Equality
class Eq a where
    (≡) ::  a -> a -> Bool

-- | Linear order

class (Eq a) ⇒ Ord a where
    (<) ::  a -> a -> Bool
    (≤) ::  a -> a -> Bool



instance Ord Bool where
    False < True = True
    _     < _    = False
```

```haskell
-- | Computes the smaller of two elements
-- min True False ⟶ False
-- min 3    5     ⟶ 3
min ::(Ord a) ⇒ a → a → a
min a b =  if a < b then a else b

-- | Equality
class Eq a where
    (≡) ::  a -> a -> Bool

-- | Linear order
-- * a ≡ b = ¬(a < b || b < a)
class (Eq a) ⇒ Ord a where
    (<) ::  a -> a -> Bool
    (≤) ::  a -> a -> Bool



instance Ord Bool where
    False < True = True
    _     < _    = False
```

```
-- | Computes the smaller of two elements
-- min True False ⟶ False
-- min 3    5     ⟶ 3
min ::(Ord a) ⇒ a → a → a
min a b =  if a < b then a else b

-- | Equality
class Eq a where
    (≡) ::  a -> a -> Bool

-- | Linear order
-- * a ≡ b = ¬(a < b || b < a)
class (Eq a) ⇒ Ord a where
    (<) ::  a -> a -> Bool
    (≤) ::  a -> a -> Bool
    default instance Eq a where
        a ≡ b = ¬(a < b || b < a)

instance Ord Bool where
    False < True = True
    _     < _    = False
```

```
-- | Computes the smaller of two elements
-- min True False ⟶ False
-- min 3    5     ⟶ 3
min ::(Ord a) ⇒ a → a → a
min a b =  if a < b then a else b

-- | Equality
class Eq a where
    (≡) ::  a -> a -> Bool

-- | Linear order
-- * a ≡ b = ¬(a < b || b < a)
class (Eq a) ⇒ Ord a where
    (<) ::  a -> a -> Bool
    (≤) ::  a -> a -> Bool
    default instance Eq a where
        a ≡ b = ¬(a < b || b < a)

instance Ord Bool where
    False < True = True
    _     < _    = False
```

Contributions:

- reduced boilerplate
- more consistent
- better refactoring