**Introduction to Haskell Type Classes**

Jim Royer

EECS

December 21, 2010

---

**A less ad-hoc approach to ad-hoc polymorphism**

Type classes are “clubs” types can join.
- There are membership requirements
- There are membership benefits
- There are membership cards you can show to get into places

You can find the standard classes described in:

http://haskell.org/onlinereport/basic.html#sect6.3

---

**Our running example**

```haskell
data Duration = Time Integer Integer
  deriving (Show)
```

Our intent: `Time h m ≡ h hours and m minutes`. E.g.,
- Time 0 12
- Time 1 30
- Time 0 90

We would like to write `(Time 1 30) == (Time 0 90)` and have Haskell return True.

---

**The Eq class**

```haskell
class Eq a where
  (==), (/=) :: a -> a -> Bool
  x /= y = not (x == y)
  x == y = not (x /= y)
```

- Provides equality (==) and inequality (/=) methods.
- Instances of Eq can be derived for any user-defined datatype whose constituents are also instances of Eq.
- If an instance declaration for Eq defines neither == nor /=, then both will loop.
- If one is defined, the default method for the other will make use of the one that is defined.
- If both are defined, neither default method is used.
Adding Duration to the Eq Class

instance Eq Duration where
  Time h1 m1 == Time h2 m2 = (60*h1+m1==60*h2+m2)

Now
*Main> Time 0 90 == Time 1 30
True

*Main> Time 0 90 == Time 1 31
False

Adding Duration to the Ord and Num Classes

The Ord Class:
http://www.haskell.org/onlinereport/basic.html#sect6.3.2

instance Ord Duration where
  Time h1 m1 <= Time h2 m2 = (60*h1+m1 <= 60*h2+m2)

The Num Class:
http://www.haskell.org/onlinereport/standard-prelude.html

instance Num Duration where
  Time h1 m1 + Time h2 m2 = Time h m
  where (h,m) = divMod (60*(h1+h2)+m1+m2) 60

  Time h1 m1 - Time h2 m2 = Time h m
  where (h,m) = divMod (60*(h1-h2)+m1-m2) 60

  fromInteger n = Time h m
  where (h,m) = divMod n 60

Adding Duration to the Show Class

The Show and Read Classes:
http://www.haskell.org/onlinereport/basic.html#sect6.3.3

data Duration = Time Integer Integer
               -- deriving (Show)

instance Show Duration where
  show (Time h m)
    = show h ++ " hours and " ++ show m ++ " minutes"

How Do Derived Instances Work

RTFM:
http://www.haskell.org/tutorial/stdclasses.html#sect8.4