HASKELL'S TYPECLASSES

Curt Clifton
Rose-Hulman Institute of Technology

Please SVN Update your *HaskellInClass* folder, then open *typeClasses.hs*

SOME ADMINISTRATIVE FOO

ERLANG TEXT

- On-line resources are kinda sucky
- Using *Programming Erlang, Software for a Concurrent World*, by Joe Armstrong
  - PDF and paper versions available for purchase at: http://www.pragprog.com/titles/jaerlang
  - Cost for PDF + ePub + mobi is $22.50

CARTOON OF LAST THURSDAY

CARTOON OF LAST THURSDAY

HASKELL *TYPECLASSES*

- Like interfaces in Java
  - Provide polymorphism by specifying that a type supports certain operations
  - But more powerful…

EXAMPLE

instance type name, think "self" but for types

```
class MyEq a where
    isEqual :: a -> a -> Bool
```

declares a typeclass, but think "interface"

Any type that claims to be an instance of MyEq (think "implements the MyEq interface") must provide a function that takes two things of it's type and returns a Bool.

INSTANCE DECLARATIONS

- Syntax:
  instance *TypeClassName DataType* ***where***
  <Required and optional function declarations>

- Example:

```
instance MyEq String where
  isEqual "" "" = True
  isEqual "" _  = False
  isEqual _  "" = False
  isEqual (c:cs) (c':cs') =
    (c == c') && isEqual cs cs'
```

think "String implements MyEq"

MORE POWER!

```
class MyEq2 a where
  isEqual2 :: a -> a -> Bool


  isNotEqual2 :: a -> a -> Bool
```

MORE POWER!

```
class MyEq2 a where
  isEqual2 :: a -> a -> Bool
  isEqual2 x y =
    not (isNotEqual2 x y)

  isNotEqual2 :: a -> a -> Bool
  isNotEqual2 x y =
    not (isEqual2 x y)
```

Q2

SOME BUILT-IN TYPECLASSES

- Show: converts values to Strings
  - show :: (Show a) => a -> String
- Read: the opposite of Show, provides simple parsing
  - read :: (Read a) => String -> a
  - readsPrec :: (Read a) => Int -> String -> [(a, String)]
- Eq, Ord, Num, Double, Float, Int, Integer, Rational, …

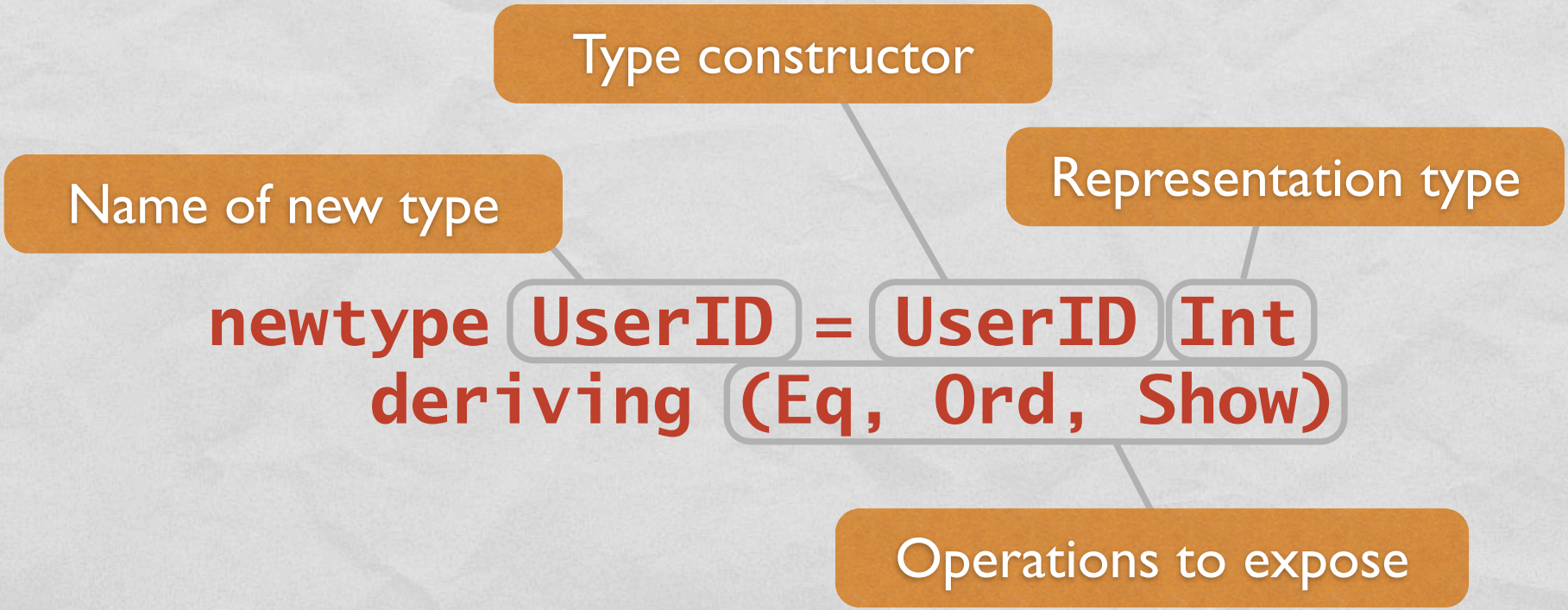
I NEED MORE POWER!

```
data Color = Red | Yellow | Blue
  deriving
    (Read, Show, Eq, Ord, Enum)
```

Collectible Scotty Plate

get yours at
http://collectibleshop.tripod.com/star-trek-plates.html

ONE MORE WAY TO NAME TYPES



THREE WAYS TO NAME TYPES

- *data BinTree a = ExtNode | IntNode a (BinTree a) (BinTree a)*
  - A brand new, structured, algebraic datatype
- *type String = [Char]*
  - Just synonyms, *String* and *[Char]* interchangeable
- *newtype UserID = UserID Int deriving (Eq, Show, Ord)*
  - Distinct type, represented as underlying type, but only supports some operations, not interchangeable