COMP105 Lecture 14

More Type Classes

Converting to strings

The **show** function converts other types to strings

```
ghci> show 123
"123"

ghci> show [1,2,3]
"[1,2,3]"

ghci> show (True, 2.5)
"(True,2.5)"
```

Converting to strings

The **Show** type class contains types that can be shown

```
ghci> :t show
show :: Show a => a -> String
```

Show contains

- all basic types
- all tuples containing showable types
- all lists that contain showable types

Converting from strings

Read converts strings to other types

```
ghci> read "123" :: Int
123

ghci> read "False" :: Bool
False

ghci> read "[1,2,3,4]" :: [Int]
[1,2,3,4]
```

The use of `::` is necessary to tell Haskell what type it is parsing

Converting from strings

It is not necessary to use `::` when Haskell can deduce the type from the context

```
ghci> not (read "False")
True

ghci> :t not
not :: Bool -> Bool


ghci> read "4" * read "6"
24
```

Converting from strings

The **Read** type class contains all types that can be read

```
ghci> :t read
read :: Read a => String -> a
```

As with show, it contains

- all basic types
- all tuples containing readable types
- all lists that contain readable types

Ordered types

The type class **Ord** contains all types that can be compared

```
ghci> :t (>)
(>) :: Ord a => a -> a -> Bool

ghci> :t (<=)
(<=) :: Ord a => a -> a -> Bool

ghci> :t max
max :: Ord a => a -> a -> a
```

Ordered types

It contains numbers, but also **all** basic types, tuples, and lists

```
ghci> 'a' < 'b'
True

ghci> True > False
True

ghci> (1, 10) <= (1, 11)
True

ghci> [1..10] < [2..11]
True
```

Tuples and lists are compared **lexicographically** (element by element)

Exercises

What are the types of the following functions?

1. `showTuple (x, y) = show x ++ " " ++ show y`

2. `addThree x = read x + 3`

3. `headLt10 (x:xs) = x <= 10`