Lecture 14

Type classes

Consider the following interaction with the `ghci`

```
GHCi, version 7.0.3: http://www.haskell.org/ghc/  :? for help
Loading package ghc-prim ... linking ... done.
Loading package integer-gmp ... linking ... done.
Loading package base ... linking ... done.
Loading package ffi-1.0 ... linking ... done.
Prelude> (+) 1

<interactive>:1:1:
    No instance for (Show (a0 -> a0))
      arising from a use of ‘print’
    Possible fix: add an instance declaration for (Show (a0 -> a0))
    In a stmt of an interactive GHCi command: print it
Prelude> (+) 1 2
3
Prelude>
```

Here the interpreter is able to display the value `3` but not the function `(+) 1`. The error message is instructive. It says that there is no instance of `Show (a0 -> a0)` for use with `print`. Let us see what `Show` and `print` are by using the `:info` command of the interpreter.

```
$ ghci
GHCi, version 7.0.3: http://www.haskell.org/ghc/  :? for help
Loading package ghc-prim ... linking ... done.
Loading package integer-gmp ... linking ... done.
Loading package base ... linking ... done.
```

```
Loading package ffi-1.0 ... linking ... done.
Prelude> :info Show
class Show a where
  showsPrec :: Int -> a -> ShowS
  show :: a -> String
  showList :: [a] -> ShowS
        -- Defined in GHC.Show
instance (Show a, Show b) => Show (Either a b)
  -- Defined in Data.Either
instance Show a => Show [a] -- Defined in GHC.Show
instance Show Ordering -- Defined in GHC.Show
instance Show a => Show (Maybe a) -- Defined in GHC.Show
instance Show Int -- Defined in GHC.Show
instance Show Char -- Defined in GHC.Show
instance Show Bool -- Defined in GHC.Show

[lots of similar lines delete]

Prelude> :info print
print :: Show a => a -> IO ()   -- Defined in System.IO
Prelude>
```

To understand what it means we need to know how the interpreter behaves. When even you enter an expression on the command line the interpreter evaluates and tries to print it. However not all haskell types are printable. The function `print` has the type `Show a => a -> IO ()` which means `print` argument type `a` is not a general type variable but is constranined to take types which are an instance of `Show`. Here `Show` is a type class.

Note that from `:info Show` it is clear that the standard types like `Int`, `Bool` etc are all instances of the class `Show`. Therefore the interpreter could print any value of those type. However, a function type is not an instance of `Show` and hence could not be printed.