Summary of Haskell classes

For more detail see section 6 of the Haskell 98 Language and Libraries Revised Report.

1. *Input-output*

`Show`
– has `show` function that converts value to string

`Read`
– has `read` function that converts string to value

2. *equality and order*

`Eq`
– has concept of equality
– has `==` and `/=` functions

`Ord`
– extends `Eq`
– has concept of order
– has `<, <=, >, >=` and `compare` functions

3. *numerical classes*

`Num` (for "numeric")
– extends `Eq, Show`
– includes the types `Int, Integer, Float, Double`
– has concept of arithmetic operations
– has `+, -, *, abs,` and `fromInteger` functions

`Integral`
– includes the types `Int, Integer`
– has concept of remainder
– has `quot, rem, div, mod` and `toInteger` functions

`Fractional`
– extends `Num`
– includes the types `Float, Double`
– has concept of division
– has `/, recip,` and `fromRational` functions

`Floating`
– extends `Fractional`
– includes the types `Float, Double`
– has trig functions, exponentials and logarithms, etc.
– has `exp, log, sqrt, sin, cos, ..., asin, ..., sinh, ..., asinh, ...`

4. *classes from category theory*

`Monad`
– includes the types `IO, Maybe`
– a very general kind of type that includes many common design patterns of Haskell

`MonadPlus`
– includes the types `IO, Maybe`
– has a bit of additional structure beyond the basic `Monad` class

`Functor`
– has concept of `map`
– includes types `[a]`, `Maybe`, trees and other data structures
– has a function `fmap :: Functor a => (b -> c) -> a b -> a c`
– `fmap` should satisfy `fmap( f . g ) = fmap f . fmap g`

5. *miscellaneous*

`Enum` (for "enumeration")
– includes many common types
– has concepts of predecessor and successor
– has `pred` and `succ` functions

`Bounded`
– includes many common types
– has concepts of predecessor and successor
– has `maxBound` and `minBound` functions

```
> maxBound :: Int
2147483647
> minBound :: Int
-2147483648
> maxBound :: Integer
ERROR - Cannot infer instance
*** Instance : Bounded Integer
*** Expression : maxBound
```