1 Давайте поговорим о тайпклассах

1.1 Введение

1.1.1 Определения

Тайпкласс - (классы типов) – это способ получить из неограниченного полиморфизма ограниченный. Т.е. есть вот у нас полностью полиморфные функции

```
f :: Int -> Int
f _ = 0
g :: a -> a
g x = x
```

У этих функций сверху тривиальная реализация и все понятно. А давайте мы введем "общее повеение, которое могут реализовывать некоторые типы". Это и будет тайпкласс

```
class CanFoo a where
  toFoo :: a -> String

instance CanFoo Int where
  toFoo 42 = "42!"
  toFoo _ = "Oops!"

main = do
  print $ toFoo (42::Int)
```

Мы получим некий тайпкласс CanFoo. И реализовали его только для инта. При этом нам даже обязательно пришлось указать, что 42 - это именно Int, а не какой-то там другой тип. Теперь мы можем писать так:

```
code :: CanFoo a => a -> String
code x = "!" ++ toFoo x
```

И при этом не можем писать просто вот так:

```
code a -> String
code a = "!" ++ toFoo a
```

Так как во втором случае не понятно, умеет ли x в CanFoo. А в первом варианте мы в сигнатуре функции указали, что да, аргумент умеет в CanFoo.

1.2 Примеры тайпклассов в haskell и их расширения

1.2.1 Пример тайпкласса Ord в Haskell

Вот этот кусок у вас выведется, если в ghci написать : *iOrd*

```
class Eq a => Ord a where
    compare :: a -> a -> Ordering
    (<) :: a -> a -> Bool
    (<=) :: a -> a -> Bool
    (>) :: a -> a -> Bool
    (>=) :: a -> a -> Bool
    max :: a -> a -> a
    min :: a -> a -> a
    {-# MINIMAL compare | (<=) #-}

instance Ord a => Ord [a]
instance Ord Word
instance Ord Ordering
instance Ord Int
instance Ord Float
instance Ord Double
instance Ord Char
instance Ord Bool
```

В первой строчке тут указано, что сравнение возможно только для сущностей, которые также умеют сравниваться между собой (т.е. у них описано поведение для Eq)

1.2.2 Полугруппа

Полугруппа - это свойство типа, о котором известно, что у него есть одна ассоциативная операция. Это не совсем матичетическое определение, но для нас подойдет.

```
class Semigroup a where
  (<>) :: a -> a -> a
```

У нее есть законы. Ассоциативность:

$$x <> (y <> z) == (x <> y) <> z$$

Эта штука, например, работает для минимума, максимума, конкатенации строк.

1.2.3 Моноид

Моноид - это полугруппа с нейтральным элементом.

```
1 class Semigroup a => Monoid a where
2  mempty :: a
```

1.2.4 Функция mconcat

В стандартной библиотеке есть функция mconcat. Она схлопывает спикок элементов полугрупп.

```
1 mconcat :: Monoid a => [a] -> a
```