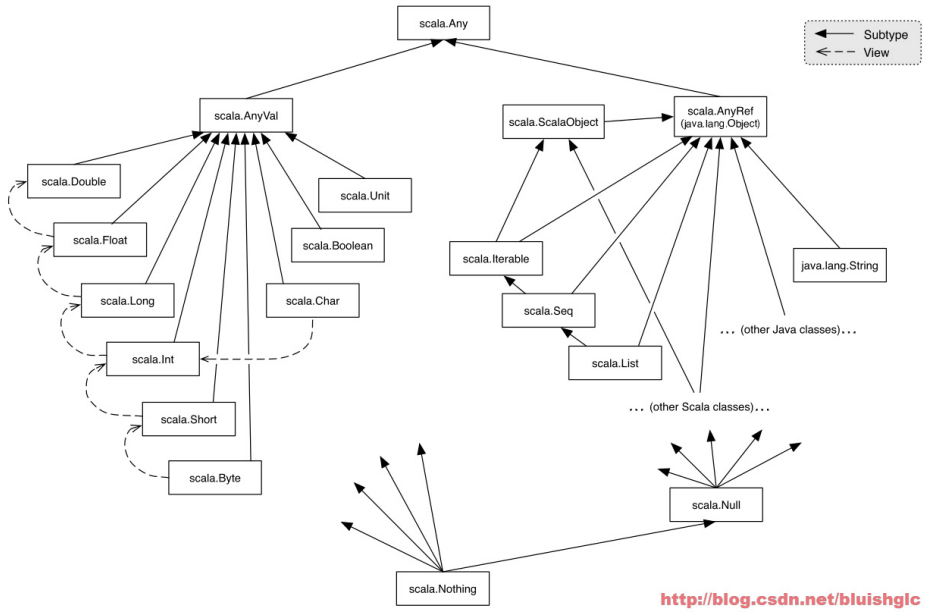
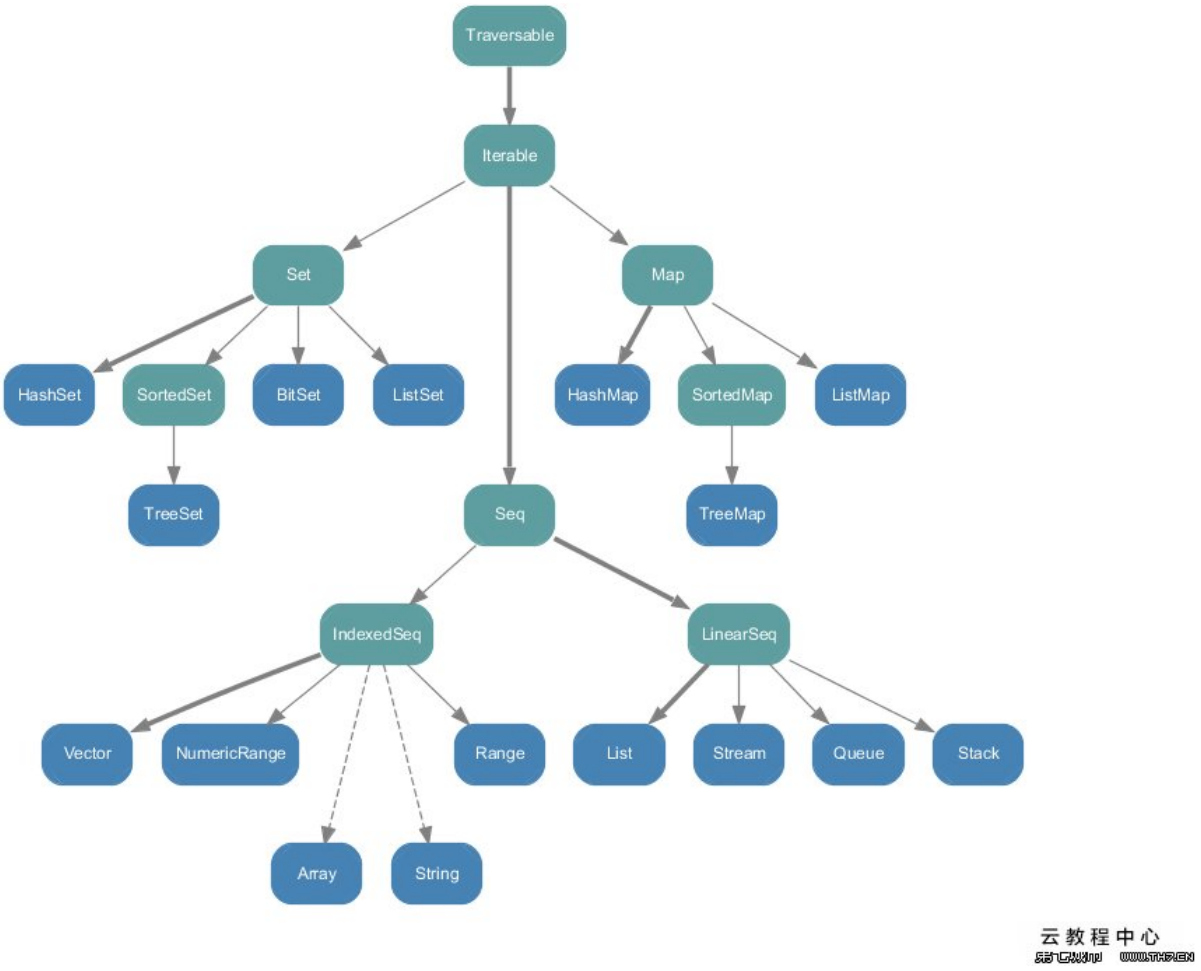
整体类型架构图(待替换)



iterable类型架构图




```
def lazyFunc(x: Int, y: => Int) = 1   // 表示接受的是一个变量
// x 参数为传值(call by value), y 参数是传名称(call by name)
// 传名称相当于惰性参数,使用到该参数时才会求值,用一次求一次; 而传值则会先求值,不管函数内部是否使用或者使用几次
def func(() => Int) =1 // 表示接受的是一个函数,没有入参,返参是Int的函数
```

println "23" // 等同于 println("23")
// 在scala里，函数都可以写成操作符的形式，这使得函数定义更像数学表达式,若参数只有一个，圆括号也是可以省略的。

隐式转换:
1.当调用每个类的方法没有时,编译器会试图隐式转换,从作用域中寻找方法(可能放在伴生对象中)
2.当传函数的参数时,不是函数所需要的类型,会发生隐式转换,在作用域中寻找可以转变类型的函数(不在乎此函数的变量名,只关心入参和返参类型),配合柯西化,可以不传隐式参数