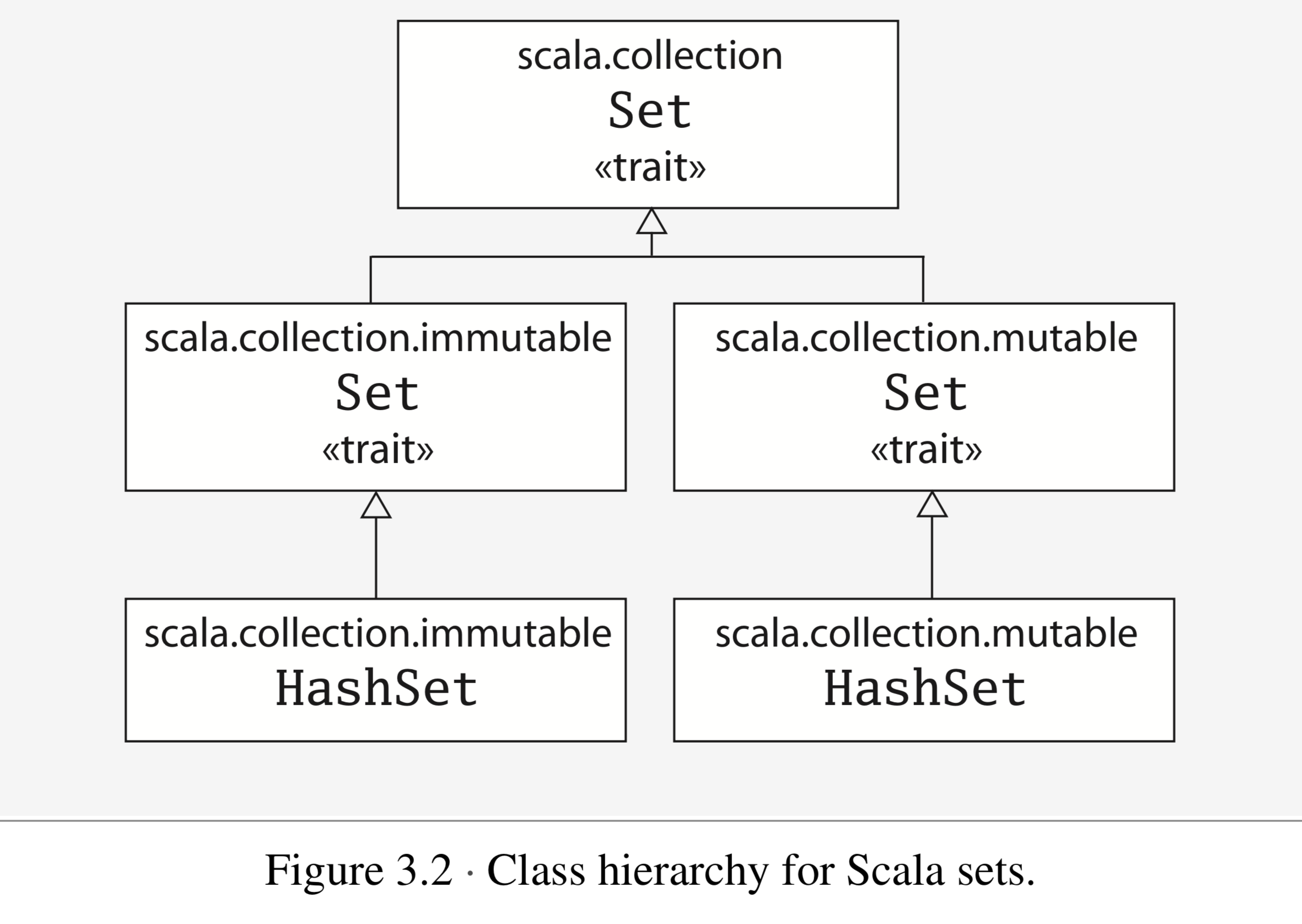
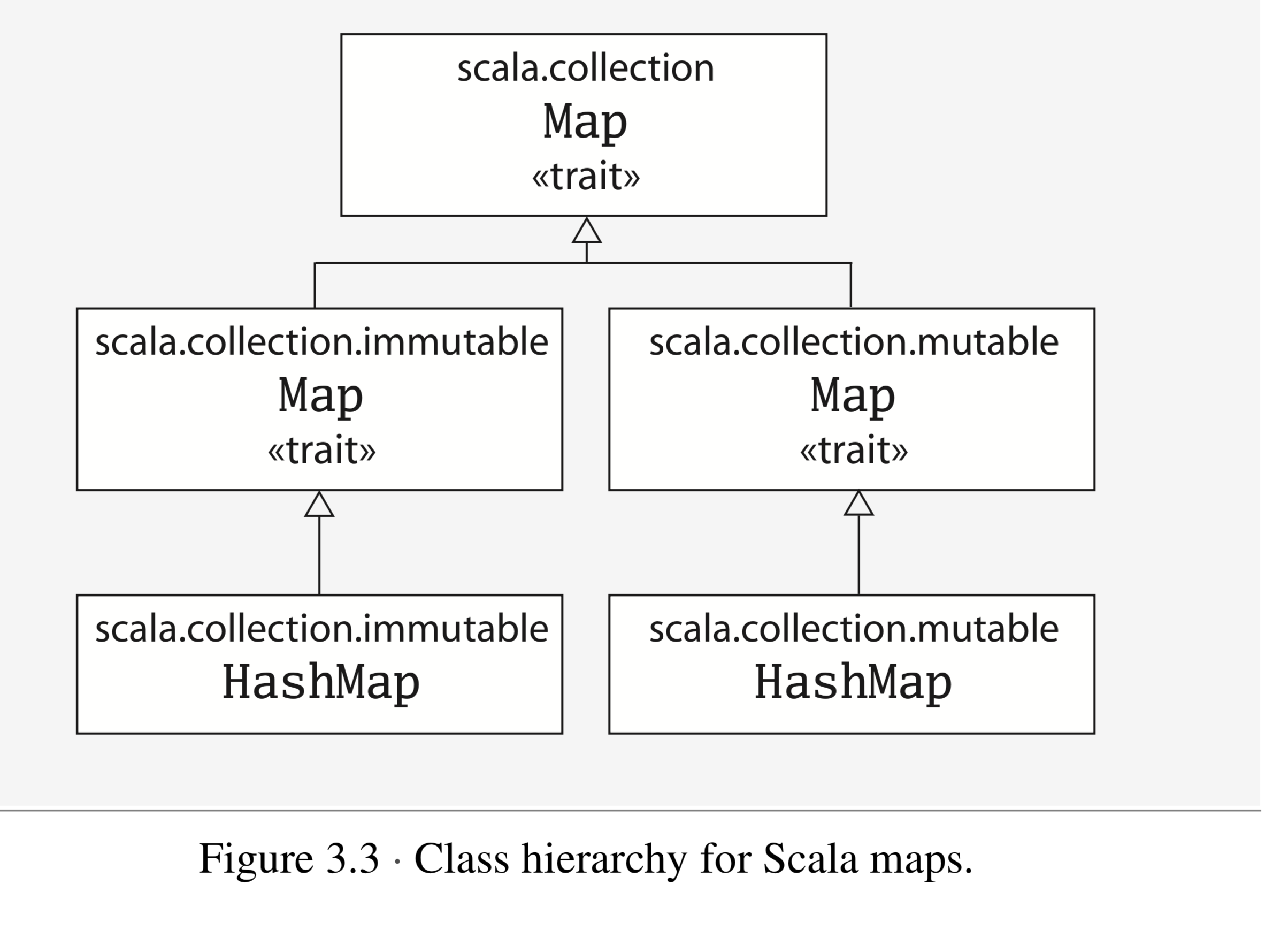
数组和List集合的区别

1arrays are always mutable; lists are always immutable.





正确对待scala的态度：

1. 优先使用函数式编程，使用val , immutable object ,

a function without side effect.

1. 在某些适合命令式编程的场合用命令式编程。

优先使用函数式编程的理由是：

1. 代码的可读性更高
2. 代码重构时不容易出错。
3. 代码更容易理解

chapter six Functional Objects

functional objects : that do not have any mutable state

immutable object compare to mutable object

Immutable object trade-offs

Immutable objects offer several advantages over mutable objects, and one potential disadvantage. First, immutable objects are often easier to reason about than mutable ones, because they do not have complex state spaces that change over time. Second, you can pass immutable objects around quite freely, whereas you may need to make defensive copies  
of mutable objects before passing them to other code. Third, there is no way for two threads concurrently accessing an immutable to corrupt its state once it has been properly constructed, because no thread can change the state of an immutable. Fourth, immutable objects make safe hash table keys. If a mutable object is mutated after it is placed into a HashSet, for example, that object may not be found the next time you look into the HashSet.

The main disadvantage of immutable objects is that they sometimes require that a large object graph be copied, whereas an update could  
be done in its place. In some cases this can be awkward to express and might also cause a performance bottleneck. As a result, it is not uncommon for libraries to provide mutable alternatives to immutable classes. For example, class StringBuilder is a mutable alternative to the immutable String. We’ll give you more information on designing mutable objects in Scala in Chapter 18.

scala的identifier 有：alphanumeric and operator

You have now seen the two most important ways to form an identifier in Scala: alphanumeric and operator

Functions and Closures

方法和函数的关系：

methods, which are functions that are members of some object

【方法是属于某些对象的函数，可以看出方法是函数的特例，函数包括方法】

the concept of a function in Scala is more general than a method.

Scala supports repeated parameters, named arguments, and default arguments.

Functions which call themselves as their last action, are called *tail recursive*

*函数式编程是跟命令式编程相比较的。*

*面向对象是跟面向过程相比较的。*

*在scala中完全可以用函数递归代替循环*

*higher-order functions*—functions that take functions as parameters—give you extra opportunities to condense and simplify code.

*一个带有函数为参数的函数称为高阶函数*

Composition means one class holds a reference to another, using the referenced class to help it fulfill its mission. Inheritance is the superclass/subclass relationship.

*组合是一个类中引用另一个类*

*继承是父类和子类的关系*

*high-order function definition:*

a function is “higher-order” if it takes one or more other func- tions as a parameters.

trait compare to class

First, a trait cannot have any “class” parameters

The other difference between classes and traits is that whereas in classes, super calls are statically bound, in traits, they are dynamically bound

thin interface compare to rich interface

thin interface里面只有抽象方法

rich interface 里面还有默认实现

You have now seen one major use of traits: turning a thin interface into a rich one. Now we’ll turn to a second major use: providing stackable modifi- cations to classes.