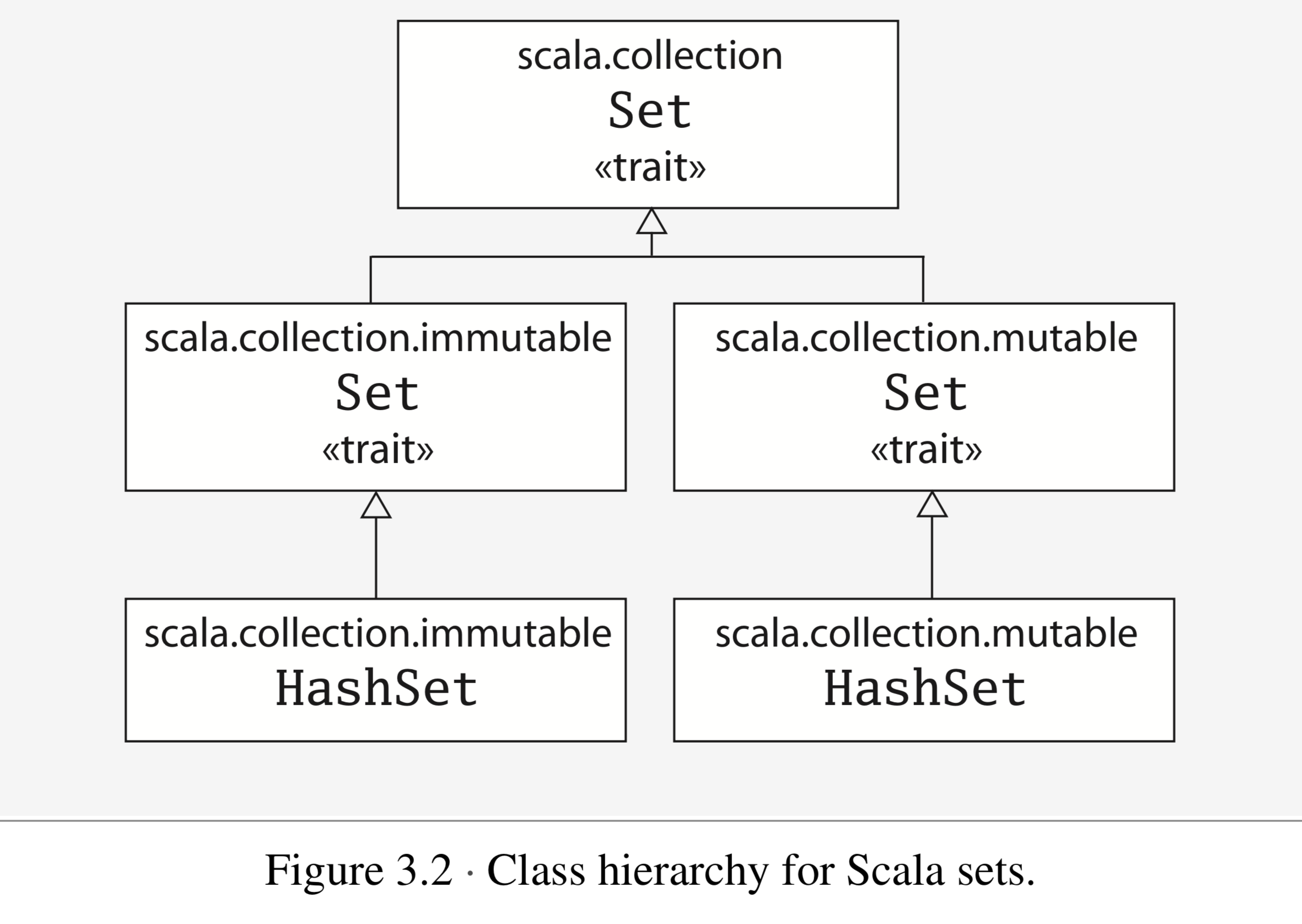
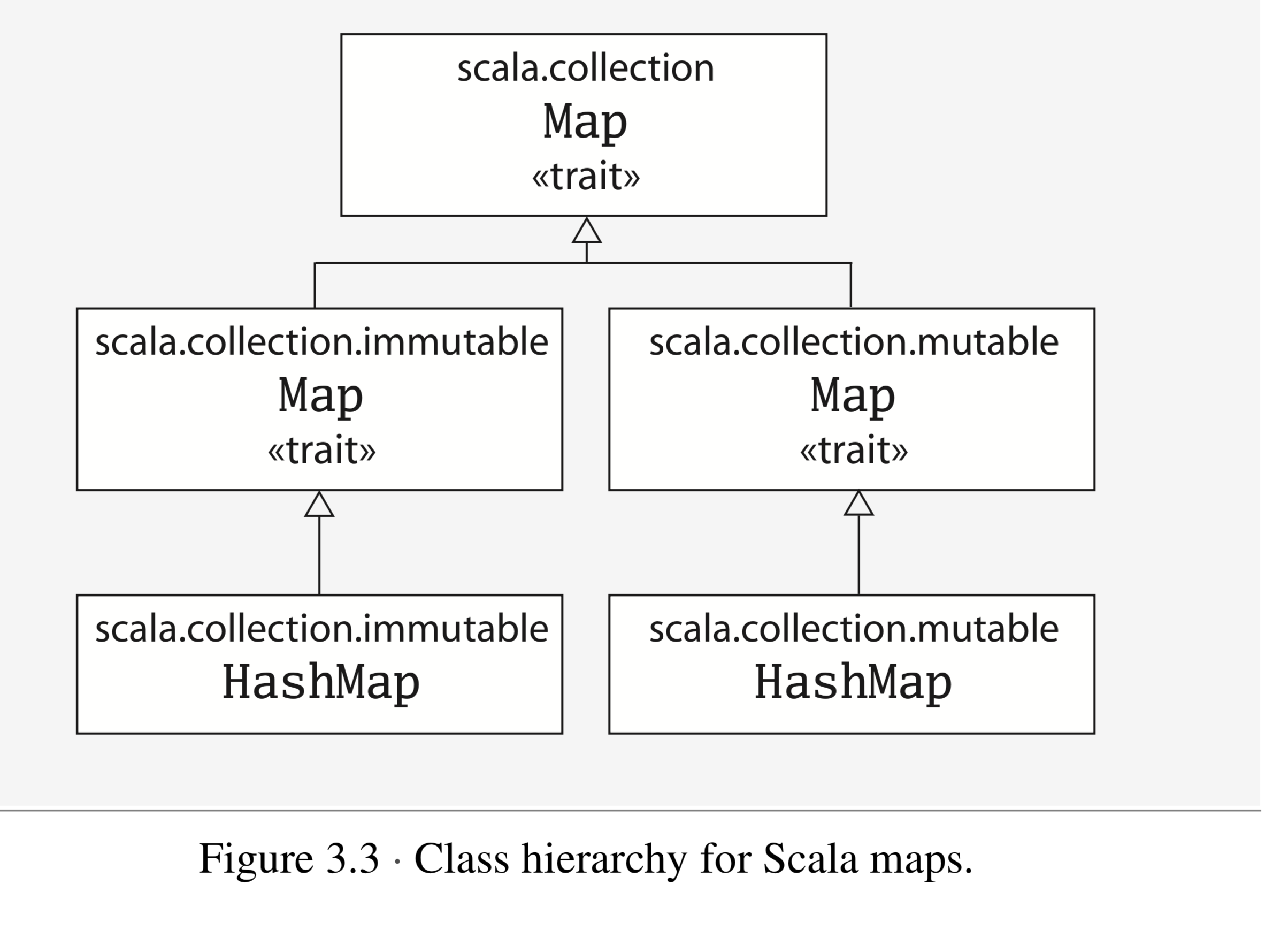
数组和List集合的区别

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.