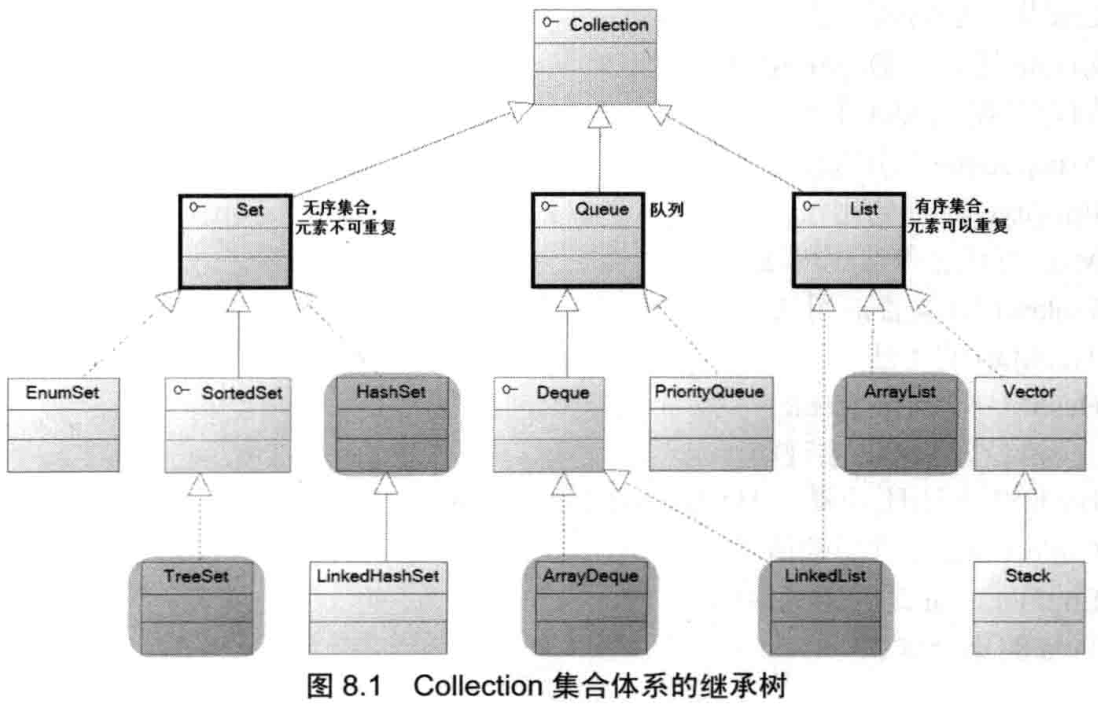
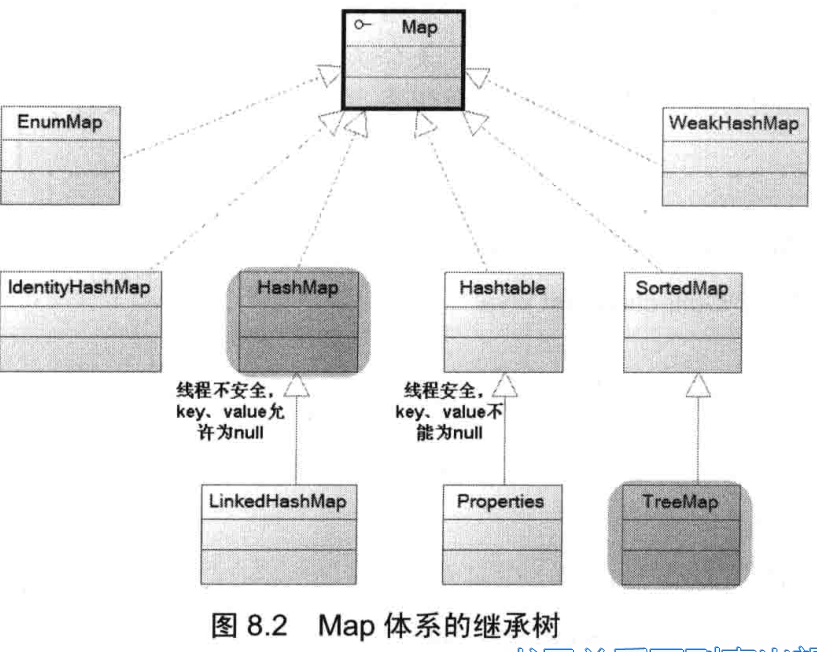
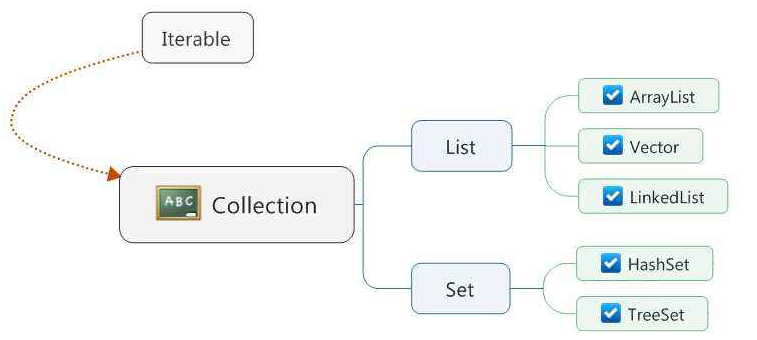
# 集合框架概览



接口：List，Set，Queue, Deque, SortedSet



接口：map

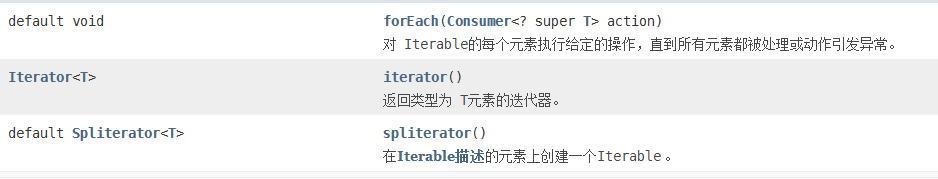


# 迭代

Iterator

|  |
| --- |
| **public** **interface** Iterator<E> {  如果迭代具有更多的元素，则返回true 。 （换句话说，如果[next()](mk:@MSITStore:E:\大学学习\java细节\面试题\jdk+api+1.8_China\jdk%20api%201.8_google.CHM::/java/util/Iterator.html#next--)返回一个元素而不是抛出一个异常，则返回true ）  **boolean** hasNext();  返回迭代中的下一个元素  E next();  此方法从底层集合中溢出此迭代器返回的最后一个元素，每次调用next方法后只能调用此方法一次，如果正在进行迭代的时候，除此方法之外的任何方法任何形式修改集合，则迭代器的内容是不确定的。  **default** **void** remove() {  **throw** **new** UnsupportedOperationException("remove");  }  对每个剩余元素执行给定的操作，直到所有元素都被处理或动作引发异常。 如果指定了该顺序，则按迭代的顺序执行操作。 动作抛出的异常被转发给呼叫者。  **default** **void** forEachRemaining(Consumer<? **super** E> action) {  Objects.*requireNonNull*(action);  **while** (hasNext())  action.accept(next());  }  } |

Iterable



最主要的方法是 **iterator()方法，返回一个类型为T的迭代器**

这样可以保证，实现方法的时候每次都能得到迭代位置从头开始的迭代器

**集合类，List，set都实现了Iterable接口，为什么不直接实现iterator接口呐？**

Iterator的核心就是next和hasNext方法，都是依赖于当前的迭代位置，即实现next和hasNext方法中，需要保存当前迭代位置的值cursor；因此如果collection直接实现了iterator接口，那么就得实现hasNext和next方法，且需要定义一个保存当前迭代值的变量。那么如果该collection作为方法，参数传递的时候，势必会将当前迭代值一同传过去，那么对于当前迭代位置，并没有进行重置，导致next的值不可预知，即使在Iterator接口中添加一个reset方法，一个collection只能保存一个迭代位置的值，如果重置了，就会影响其他正在遍历的值。因此，多个方法调用之间相互干扰。如果实现了Iterable接口，每次调用都会返回一个从头开始计数的迭代器，多个迭代器之间互不干扰。