集合

List

> 在集合中， `List` 是最基础的一种集合：它是一种有序列表。

Map

`Map<K,V>` 是一种键-值映射表，当我们调用 `put(K key, V value)` 方法时，就把 `Key` 和 `Value` 做了映射并放入 `Map` .

> 始终牢记：Map中不存在重复的Key，因为放入相同的Key，只会把原来的Key-Value对应的value替换掉。
> Map中的Key不是一个有序的，也不能假设输出的Key时有序的。

正确使用 `Map` 必须保证：

- 作为 `Key` 的对象必须正确覆写 `equals` 方法，相等的两个 `Key` 实例调用 `equals` 必须返回true;
- 作为 `Key` 的对象还必须正确覆写 `hashCode` 方法，且 `hashCode` 方法要严格遵循下面规范：
  1. 如果两个对象相等，则两个对象的 `hashCode` 必须相等；
  2. 如果两个对象不相等，则两个对象的 `hashCode` 尽量不要相等。

Set

我们知道， `Map` 用于存储 `Key-Value` 的映射，对于充当key的对象，是不能重复的，并且，不当需要正确覆写 `equals` 方法，还要正确覆写 `hashCode` 方法。如果只需要存储不重复的key,并不需要存储映射的value，那么就可以使用 `Set`

Queue

队列 `Queue` 是一种经常使用的集合， `Queue` 实际上实现了一个先进先出(FIFO: First In First Out)的有序表。它和 `List` 的却别在于， `List` 可以在任意位置添加和删除元素，而 `Queue` 只有两个操作：

- 把元素添加到队列末尾；
- 从队列头部取出元素。

Deque

`Queue` 是一个队列，只能一头进，一头出。
`Deque` 实现了一个双端队列，它的功能是：

- 既可以添加到队尾，也可以添加到队首；
- 既可以从队首获取，又可以从队尾获取。

Stack

栈(`Stack`)是一种后进先出（LIFO）的数据结构，操作栈的方法有：

- 把元素压栈:(`push(E)`)
- 把栈顶的元素取出并移除:(`pop(E)`)
- 取出栈顶元素不移除:(`peek(E)`)