# 标记接口（Marker-Interface）

## Serializable

序列化接口，用于标记一个类的对象是可以进行序列化的。所谓序列化，就是讲一个对象的状态（属性）变为可存储或者可传输的过程。

序列化一个类对象是将一个对象的属性进行序列化，因此被static和transient修饰的属性不能被序列化，因为static修饰的属性属于类，不属于对象；transient修饰的属性直接被当做不能序列化；

参考SerializableDemo类

### 注意事项

* 如果一个class实现了Serializable接口，则其子类自动实现序列化，无需显示实现Serializable接口；
* 被static和transient修饰的属性不能序列化；

## Cloneable

一个标记接口，接口中无任何方法。Java中一个类要想提供 clone功能，必须实现Cloneable接口，否则如果调用clone方法会出现CloneNotSupportedException异常。clone方法来源于java.lang.Object类，默认实现是“浅拷贝”，如果要提高“深拷贝”功能，必须override上面所说的clone方法。

clone方法返回的是一个新对象，不是原有对象的引用。

## Comparable

该接口主要用来比较类的偏序关系，如果一个类实现了该接口，必须实现其中的compareTo方法，该方法给用户提供自定义偏序关系的实现。

参考Demo 项目中的CollectionsDemo类，其中在调用sort方法时，会调用具体类的compareTo接口比较两个元素的偏序关系。

## Comparator

该接口也用于比较类的偏序关系，不过类不需要实现该接口，但是需要重新定义一个比较器类，在需要判断偏序关系的地方需要引用该比较器类的实。

实现该接口的类必须实现其中的compare方法。

参考Demo 项目中的CollectionsDemo类。

## RandomAccess

标记接口，主要用于Collections类中的binarySearch静态方法。如果集合类的子类实现了RandomAccess 接口，使用Collections进行二叉查找时则通过下标二叉查找算法，否则使用迭代器的二叉查找算法（下标二叉查找算法的性能比迭代器二叉查找算法性能高）。

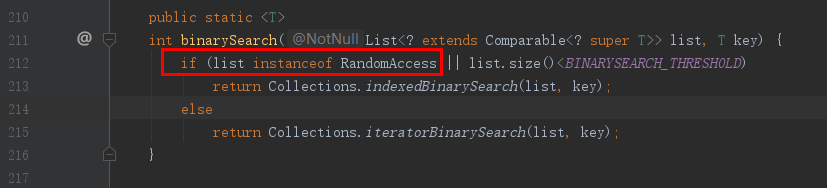


Figure 1 Collections接口二叉查找算法

第212行指出如果list实现了RandomAccess接口或者size小于 BINARYSEARCH\_THRESHOLD（值为5000），都将进入所有的indexBinarySearch，否则进入iteratorBinarySearch。

比如LinkedList虽然没有实现RandomAccess接口，但是链表大小小于5000时仍然会执行indexBinarySearch操作。

有关binarySearch的两个方法使用Demo参考Demo项目里面的CollectionsDemo类。IndexBinarySearch

### IndexBinarySearch

该方法通过下标获取元素，思路很简单。其中重点是：

* List中的元素必须实现Comparable接口
* List必须实现了通过下标获取元素的get方法

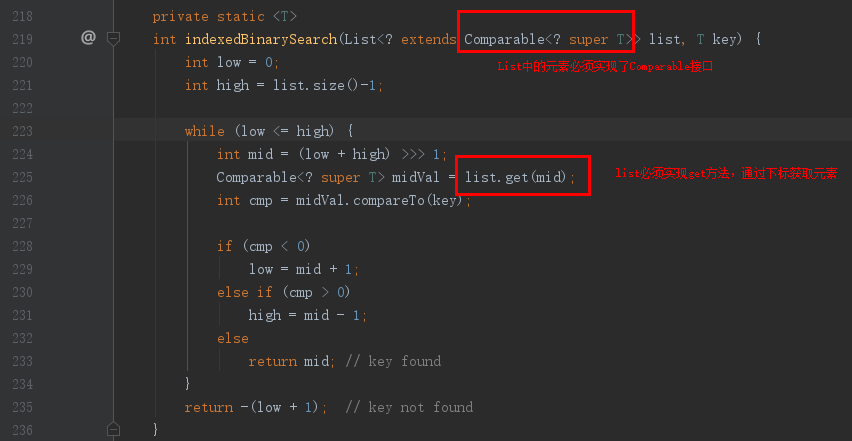


Figure 2 通过下标实现二叉查找算法

### IteratorBinarySearch

该方法的思路同indexBinarySearch，只不过在获取元素是通过iterator获取。

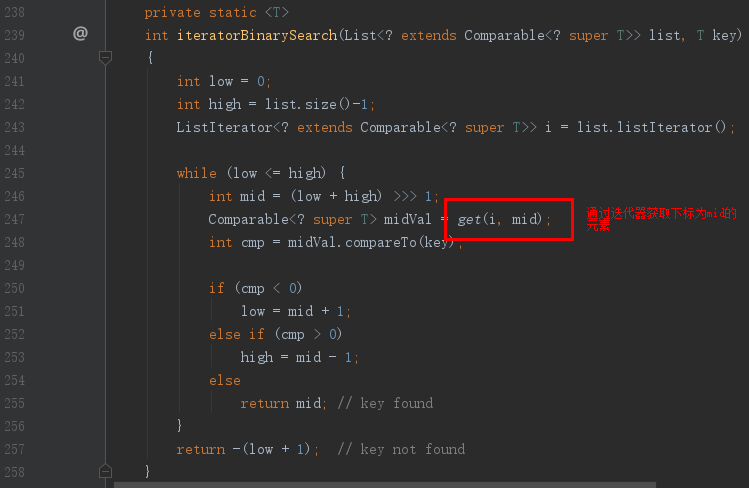


Figure 3 通过迭代器的二叉查找算法

其中的get方法如下：

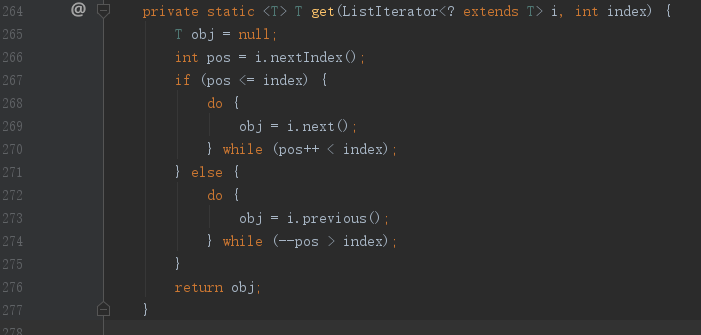


Figure 4 通过迭代器获取元素

# 功能接口

## Iterable

如果一个类实现了该接口，表明该类对象可以作为“for-each”的目标。接口方法iterator返回T类型的迭代器；只有实现了该接口的类才能调用iterator方法获取迭代器。

该接口主要使用在集合类一族，比如list和set结构。Iterable接口作为集合的顶级接口之一，实现了接口的类可以使用如下的循环方式：

For(Object o:list)

# 工具类

## Collections