4.8.1 关于 Set 接口

Set 接口代表不允许重复元的 Collection。由接口 SortedSet 给出的一种特殊类型的 Set 保证其中的各项处于有序的状态。因为一个 Set IS-A Collection，所以用于访问继承 Collection 的 List 的项的方法也对 Set 有效。图 3-6 中描述的 print 方法如果传送一个 Set 也将会正常工作。

由 Set 所要求的一些独特的操作是一些插入、删除以及(有效地)执行基本查找的能力。对于 Set，add 方法如果执行成功则返回 true，否则返回 false，因为被添加的项已经存在。保持各项以有序状态的 Set 的实现是 TreeSet。TreeSet 类的基本操作花费对数最坏情形时间。

默认情况下，排序假设 TreeSet 中的项实现 Comparable 接口。另一种排序可以通过用 Comparator 实例化 TreeSet 来确定。例如，我们可以创建一个存储 String 对象的 TreeSet，通过使用图 1-18 中编写的 CaseInsensitiveCompare 函数对象忽略大小写。下面的代码中，Set s 大小为 1。

```
Set<String> s = new TreeSet<String>( new CaseInsensitiveCompare( ) );
s.add( "Hello" ); s.add( "HeLLo" );
System.out.println( "The size is: " + s.size( ) );
```

4.8.2 关于 Map 接口

Map 是一个接口，代表由关键字以及它们的值组成的一些项的集合。关键字必须是唯一的，但是若干关键字可以映射到一些相同的值。因此，值不必是唯一的。在 SortedMap 接口中，映射中的关键字保持逻辑上有序状态。SortedMap 接口的一种实现是 TreeMap 类。Map 的基本操作包括诸如 isEmpty、clear、size 等方法，而且最重要的是包含下列方法：

```
boolean containsKey( KeyType key )
ValueType get( KeyType key )
ValueType put( KeyType key, ValueType value )
```

get 返回 Map 中与 key 相关的值，或当 key 不存在时返回 null。如果在 Map 中不存在 null 值，那么由 get 返回的值可以用来确定 key 是否在 Map 中。然而，如果存在 null 值，那么必须使用 containsKey。方法 put 把关键字/值对置入 Map 中，或者返回 null，或者返回与 key 相联系的老值。

通过一个 Map 进行迭代要比 Collection 复杂，因为 Map 不提供迭代器，而是提供 3 种方法，将 Map 对象的视图作为 Collection 对象返回。由于这些视图本身就是 Collection，因此它们可以被迭代。所提供的 3 种方法如下：

```
Set<KeyType> keySet( )
Collection<ValueType> values( )
Set<Map.Entry<KeyType,ValueType>> entrySet( )
```

方法 keySet 和 values 返回简单的集合(这些关键字不包含重复元，因此以一个 Set 对象的形式返回)。这里的 entrySet 方法是作为一些项而形成的 Set 对象被返回的(由于关键字是唯一的，因此不存在重复项)。每一项均由被嵌套的接口 Map.Entry 表示。对于类型 Map.Entry 的对象，其现有的方法包括访问关键字、关键字的值，以及改变关键字的值：

```
KeyType getKey( )
ValueType getValue( )
ValueType setValue( ValueType newValue )
```

4.8.3 TreeSet 类和 TreeMap 类的实现

Java 要求 TreeSet 和 TreeMap 支持基本的 add、remove 和 contains 操作以对数最坏情形时间完成。因此，基本的实现方法就是平衡二叉查找树。一般说来，我们并不使用 AVL 树，而是经