程序清单 27-6 给出了应用 MyHashSet 的测试程序。

程序清单 27-6 TestMyHashSet.java

```
1  public class TestMyHashSet {
2    public static void main(String[] args) {
3      // Create a MyHashSet
4      MySet<String> set = new MyHashSet<>();
5      set.add("Smith");
6      set.add("Anderson");
7      set.add("Lewis");
8      set.add("Cook");
9      set.add("Smith");
10
11     System.out.println("Elements in set: " + set);
12     System.out.println("Number of elements in set: " + set.size());
13     System.out.println("Is Smith in set? " + set.contains("Smith"));
14
15     set.remove("Smith");
16     System.out.print("Names in set in uppercase are ");
17     for (String s: set)
18       System.out.print(s.toUpperCase() + " ");
19
20     set.clear();
21     System.out.println("\nElements in set: " + set);
22   }
23 }
```

```
Elements in set: [Cook, Anderson, Smith, Lewis]
Number of elements in set: 4
Is Smith in set? true
Names in set in uppercase are COOK ANDERSON LEWIS
Elements in set: []
```

该程序应用 MyHashSet 创建一个集合（第 4 行），并添加 5 个元素到集合中（第 5 ～ 9 行）。第 5 行添加 Smith，第 9 行再次添加 Smith。由于只有不重复的元素可以存储在集合中，Smith 只在集合中出现一次。集合中实际上有 4 个元素。程序显示了元素（第 11 行），得到它的大小（第 12 行），检测集合是否包含某个指定的元素（第 13 行），删除一个元素（第 15 行）。由于集合中的元素是可遍历的，程序使用了 foreach 循环来遍历集合中的所有元素（第 17 ～ 18 行）。最后，程序清除集合（第 20 行）并显示一个空的集合（第 21 行）。

复习题

27.26　为什么可以使用 foreach 循环来遍历集合中的元素？

27.27　描述 MyHashSet 类中的 add(e) 方法是如何实现的？

27.28　程序清单 27-5 中，遍历器中的 remove 方法从集合中删除当前元素。同时它也从内部线性表中删除当前元素（第 161 行）：

```
list.remove(current); // Remove current element from the list
```

这个是必须的吗？

27.29　将程序清单 27-5 中的第 146 ～ 149 行的代码替换为一行语句。

关键术语

associative array（关联数组）
cluster（簇）
dictionary（字典）
linear probing（线性探测）
load factor（装填因子）
open addressing（开放地址法）

double hashing（再哈希）
hash code（散列码）
hash function（散列函数）
hash map（散列映射表）
hash set（散列集合）
hash table（散列表）
perfect hash function（完全散列函数）
polynomial hash code（多项式散列码）
quadratic probing（二次探测法）
rehashing（再散列）
secondary clustering(二次成簇)
separate chaining（链地址法）

本章小结

1. 映射表是存储条目的一种数据结构。每个条目包含两部分：键和值。键也称为搜索键，用于查找相应的值。可以使用散列技术来实现映射表，实现使用 $O(1)$ 的时间复杂度来实现查找、获取、插入以及删除。
2. 集合是一种存储元素的数据结构。可以使用散列技术来实现集合，实现使用 $O(1)$ 的时间复杂度来实现查找、获取、插入以及删除。
3. 散列是一种无须执行搜索，即可从一个键得到的索引获取值的技术。典型的散列函数首先将搜索键转化为一个称为散列码的整数值，然后将散列码压缩为散列表中的一个索引。
4. 当两个键映射到散列表中的同样索引上时，冲突发生。通常有两种方法处理冲突：开放地址法和链地址法。
5. 开放地址法是在发生冲突时，在散列表中找到一个开放位置的过程。开放地址法有几种变体：线性探测、二次探测以及再哈希。
6. 链地址法将具有同样散列索引的条目放到同一个位置中，而不是寻找新的位置。链地址法中每个位置称为一个桶。桶是容纳多个条目的容器。

测试题

回答位于网址 www.cs.armstrong.edu/liang/intro10e/quiz.html 的本章测试题。

编程练习题

**27.1 （应用开放地址法的线性探测法来实现 MyMap）应用开放地址法的线性探测法创建一个实现 MyMap 的新的具体类。简单起见，使用 f(key) = key % size 作为散列函数，这里 size 是散列表的大小。初始的，散列表的大小为 4。当装填因子超过阈值 (0.5) 时，表的大小翻倍。

**27.2 （应用开放地址法的二次探测法来实现 MyMap）应用开放地址法的二次探测法创建一个实现 MyMap 的新的具体类。简单起见，使用 f(key) = key % size 作为散列函数，这里 size 是散列表的大小。初始的，散列表的大小为 4。当装填因子超过阈值 (0.5) 时，表的大小翻倍。

**27.3 （应用开放地址法的再哈希法来实现 MyMap）应用开放地址法的再哈希法创建一个实现 MyMap 的新的具体类。简单起见，使用 f(key) = key % size 作为散列函数，这里 size 是散列表的大小。初始的，散列表的大小为 4。当装填因子超过阈值 (0.5) 时，表的大小翻倍。

**27.4 （修改 MyHashMap 使得可以有重复的键）修改 MyHashMap 从而允许条目可以有重复的键。需要修改 put(key,value) 的实现。同时，添加一个名为 getAll(key) 的新方法，返回一个匹配映射表中键的值的集合。

**27.5 （使用 MyHashMap 实现 MyHashSet）使用 MyHashMap 实现 MyHashSet。注意，可以使用 (key, key) 创建条目，而不是使用 (key, value)。

**27.6 （实现线性探测法的动画）编写程序，实现线性探测法的动画，如图 27-3 所示。可以在程序中修改散列表的初始大小。假设装填因子阈值为 0.75。

**27.7 （实现链地址法的动画）编写程序，实现 MyHashMap 的动画，如图 27-8 所示。可以在程序中修

改散列表的初始大小。假设装填因子阈值为 0.75。

**27.8 （实现二次探测法的动画）编写程序，实现二次探测法的动画，如图 27-5 所示。可以在程序中修改散列表的初始大小。假设装填因子阈值为 0.75。

**27.9 （实现字符串的散列码）编写一个方法，使用 27.3.2 节中描述的方法返回字符串的散列码，其中 b 取值 31。方法头如下：

```
public static int hashCodeForString(String s)
```

**27.10 （比较 MyHashSet 和 MyArrayList）程序清单 24-3 定义了 MyArrayList。编写一个程序，产生 0 到 999999 之间的 1000000 个随机双精度值，并存储在一个 MyArrayList 和 MyHashSet 中。然后产生 0 到 1999999 之间的 1000000 个随机双精度值的线性表。对于线性表中的每个数字，检测是否在数组线性表中以及是否在散列集合中。运行程序，给出对于数组线性表和散列集合的总体测试时间。

**27.11 （SetToList）编写以下方法，从一个集合中返回 ArrayList。

```
public static <E> ArrayList<E> setToList(Set<E> s)
```