```
1  public class SeparateChainingHashTable<AnyType>
2  {
3      public SeparateChainingHashTable( )
4        { /* Figure 5.9 */ }
5      public SeparateChainingHashTable( int size )
6        { /* Figure 5.9 */ }
7
8      public void insert( AnyType x )
9        { /* Figure 5.10 */ }
10     public void remove( AnyType x )
11       { /* Figure 5.10 */ }
12     public boolean contains( AnyType x )
13       { /* Figure 5.10 */ }
14     public void makeEmpty( )
15       { /* Figure 5.9 */ }
16
17     private static final int DEFAULT_TABLE_SIZE = 101;
18
19     private List<AnyType> [ ] theLists;
20     private int currentSize;
21
22     private void rehash( )
23       { /* Figure 5.22 */ }
24     private int myhash( AnyType x )
25       { /* Figure 5.7 */ }
26
27     private static int nextPrime( int n )
28       { /* See online code */ }
29     private static boolean isPrime( int n )
30       { /* See online code */ }
31 }
```

图 5-6　分离链接散列表的类架构

就像二叉查找树只对那些是 Comparable 的对象工作一样，本章中的散列表只对遵守确定协议的那些对象工作。在 Java 中这样的对象必须提供适当 equals 方法和返回一个 int 型量的 hashCode 方法，此时，散列表把这个 int 型量通过 myHash 转成适当的数组下标，如图 5-7 所示。图 5-8 解释了 Employee 类，可以将其存放在一个散列表中。类 Employee 提供 equals 方法和基于 Employee 名字的 hashCode 方法。Employee 类的 hashCode 通过使用标准 String 类中定义的 hashCode 来工作。这个标准类中的 hashCode 基本上是图 5-4 中将 14 行到 16 行除去后的程序。

```
1      private int myhash( AnyType x )
2      {
3          int hashVal = x.hashCode( );
4
5          hashVal %= theLists.length;
6          if( hashVal < 0 )
7              hashVal += theLists.length;
8
9          return hashVal;
10     }
```

图 5-7　散列表的 myHash 方法

图 5-9 列出构造方法和方法 makeEmpty。

实现 contains、insert 和 remove 的例程如图 5-10 所示。