```java
/**
 * EECS233 HW6 Programming Project 3
 * Tung Ho Lin
 */

import java.math.BigDecimal;
import java.math.MathContext;

public class MyHashTable {

  private MyNode[] data;

  private int maxSize;

  private int curSize;

  public MyHashTable() {
    data = new MyNode[64];  //starts with an initial length of 64
    maxSize = 64;
    curSize = 0;
  }

  public MyNode[] getData() {
    return data;
  }

  public int getMaxSize() {
    return maxSize;
  }

  //a method to calculate the loadfactor and round it to 3 sigfig
  public double loadfactor() {
    double d = (double) curSize/maxSize;
    BigDecimal bd = new BigDecimal(d);
    bd = bd.round(new MathContext(3));
    double rounded = bd.doubleValue();
    return rounded;
  }

  public int hash(String word) {
    int num = word.hashCode();
    if(num < 0)
      num = num * -1;
    return num % maxSize;
  }

  public void rehash() {
    int oldSize = maxSize;
    MyNode[] oldData = data;
    maxSize = oldSize * 2;
    data = new MyNode[maxSize];
    for(int i=0; i<oldSize; i++) {
      if(oldData[i] != null) {
        MyNode curNode = oldData[i];
        while(curNode != null) {
          put(curNode.data);
          curNode = curNode.next;
        }
      }
    }
  }

  public void put(String word) {
    if(contains(word))
      getNode(word).increment();  //increment the occur if the word already exists
    else {
      int index = hash(word);
      if(data[index] != null)
        data[index].getLast().next = new MyNode(word, null);
      else
```

```
71          data[index] = new MyNode(word, null);
72        curSize++;
73      }
74    }
75
76    //get the Node that contains the input word
77    public MyNode getNode(String word) {
78      int index = hash(word);
79      if(data[index] == null)   //the word has not been hashed into the table before,
    return null
80        return null;
81      else {
82        MyNode curNode = data[index];
83        while(curNode!= null){
84          if(curNode.data.equals(word))
85            return curNode;   //find the word
86          curNode = curNode.next;
87        }
88        return null;   //return null if the word doesn't exist
89      }
90    }
91
92    public boolean contains(String word) {
93      if(getNode(word) == null)
94        return false;
95      else
96        return true;
97    }
98
99    //inner class MyNode
100   public class MyNode {
101
102     private String data;
103
104     private MyNode next;
105
106     private int occur;  //the number of times this word occur
107
108     public MyNode(String data, MyNode next) {
109       this.data = data;
110       this.next = next;
111       this.occur = 1;
112     }
113
114     public MyNode getNext() {
115       return next;
116     }
117
118     public String getData() {
119       return data;
120     }
121
122     public void setData(String data) {
123       this.data = data;
124     }
125
126     public int getOccur() {
127       return occur;
128     }
129
130     //increment the number of occurences of the word
131     public void increment() {
132       occur++;
133     }
134
135     //to get the last node directly/indirectly connected to this node
136     public MyNode getLast() {
137       MyNode curNode = this;
138       while(curNode.next != null)
139         curNode = curNode.next;
```

```
140         return curNode;
141       }
142
143     }
144 }
```