```
1 /** This class MySet that implements the the interface from MulitSet
2  * this class implement the cardinality, multiplicity, add, isEmpty
3  * union and intersection of MySet
4  *
5  *    Name: Long Nguyen: Student # 5427059
6  *
7  * @version  1.0 (April. 2014)                                    */
8
9 package MULTISET;
10
11 import java.io.Serializable;
12 import java.util.Arrays;
13 import java.util.Iterator;
14
15 import TestLists.*;
16
17 public class MySet < E extends Keyed > implements MultiSet<E>, Serializable {
18
19    private Node<E>  top;  // top element of the stack
20    private int length;  //Length of the list
21
22       /** This constructor creates a new, empty stack.                    */
23
24       public MySet ( ) {
25
26          setTop(null);
27          setLength(0);
28
29       }; // constructor
30
31     //returns the number of elements in the collection
32    public int cardinality() {
33       return getLength();
34    }
```

```
35
36  //returns the number of elements that match item from this
37  public int multiplicity(E item) {
38
39      int numberOfItems = 0;
40
41      Node<E> referenceLink = this.getTop(); // reference pointer to head
42      String thisNumberValue = "";
43
44      //looping through the linkedlisted to get the values
45          while (this.getTop() != null) {
46
47              if((item.getKey().toString().compareTo(this.getTop().item.getKey()))
    == 0){
48                  thisNumberValue += this.getTop().item.getKey();
49                  numberOfItems++;
50              }
51              this.setTop(this.getTop().next);
52          }
53          this.setTop(referenceLink); // setting the pointer back to the head
54
55      return numberOfItems; //return the number of items
56  }
57  /*Adds an Item to the collection; note this is a mutable operation
58   * and checking not to add duplicate values
59   *
60   * */
61  @Override
62  public void add(E anItem) {
63
64      Node<E> referenceLink = this.getTop(); // reference pointer to head
65      boolean duplicate = false;
66
67      //looping through the linkedlisted to get the values
```

```
68            while (this.getTop() != null) {
69              //Checking for duplicate values
70               if((anItem.getKey().compareTo(this.getTop().item.getKey())) == 0){
71                  //System.out.println(this.top.item.getKey());
72                  duplicate = true;
73                  break;
74               }
75               this.setTop(this.getTop().next); //pointing to the next node
76            }
77            this.setTop(referenceLink); // setting the pointer back to the head
78
79      //if there are not duplicates then add the item to the data
80       if(duplicate == false ){
81          //System.out.println(anItem.getKey());
82          setTop(new Node<E>(anItem,getTop()));
83          setLength(getLength() + 1);
84       }
85
86    }
87
88    //returns true if this is empty
89    @Override
90    public Boolean isEmpty() {
91       return getTop() == null;
92    }
93
94
95    //returns a new MultiSet by taking the union of this and aSet,
96    //the operation is immutable, neither this or aSet is modified
97    @Override
98    public MultiSet<E> union(MultiSet<E> aSet) {
99
100      char[] unsortedSet = null; // unsortedSet
101      String valuesOfStringSet = "";  // initialize the string
```

```
102
103     MultiSet <E> unionSet = new MySet <E>(); //creating a new MySet
104     char b;
105     E values;
106     Object  element;
107
108      Iterator itr = aSet.iterator();
109     //iteratoring through the list to add in the string
110      while(itr.hasNext()) {
111          element = ((Keyed) itr.next()).getKey();
112          //System.out.println(element);
113          valuesOfStringSet += element.toString();
114         }//end while loop
115
116
117       Node<E> referenceLink = this.getTop(); // reference pointer to head
118
119       //looping through the linkedlisted to get the values and adding them
    to a string
120             while (this.getTop() != null) {
121
122                element = this.getTop().item.getKey();
123                valuesOfStringSet += element.toString();
124                 this.setTop(this.getTop().next); //pointing to the next node
125             }
126             this.setTop(referenceLink); // setting the pointer back to the head
127
128     //making the string into a char array and sorting it
129      unsortedSet = valuesOfStringSet.toCharArray();
130       Arrays.sort(unsortedSet);
131
132      //adding the values to unionSet backward
133      for(int i = unsortedSet.length-1; i >=0 ; i--){
134          b = (char) unsortedSet[i];
```

```
135             values = (E) new KeyedChar(b);
136             unionSet.add(values);
137           }
138
139       return unionSet;    //returning the union of the collection
140   }
141
142   /*This method checks if both sets are equal*/
143   public Boolean equal(MultiSet<E> aSet) {
144
145       boolean isEqual = false; // setting the initial state to false
146
147       Node<E> referenceLink = this.getTop(); // reference pointer to head
148
149       //while the pointer is not null and iterator has next
150        while(aSet.iterator().hasNext() && this.getTop() != null) {
151          //Iterator through the aSet to get the key
152             Object element = aSet.iterator().next().getKey();
153             //System.out.println(element);
154
155             //if comparing each values one at a time to check if they are equal
156              if((element.toString().compareTo(this.getTop().item.getKey())) == 0){
157                 isEqual = true;
158                 //else set it to false if they are not equal
159              }else{
160                 isEqual = false;
161                 break;
162              }//end else
163
164             this.setTop(this.getTop().next);//pointing to the next node in the
    linked list
165        }
166        this.setTop(referenceLink); // setting the pointer back to the head
167
```

```
168     return isEqual; //return true of false if its equal
169   }
170
171   /*This method gets the intersection between the two sets */
172   @Override
173   public MultiSet<E> intersection(MultiSet<E> aSet) {
174
175     char[] unsortedSetA = null; // unsortedSet char array
176     char[] unsortedSetB = null; // unsortedSet char array
177
178     String valuesOfStringSetA = "";   // initialize the string
179     String valuesOfStringSetB = "";   // initialize the string
180
181     MultiSet <E> intersectionSet = new MyBag <E>(100);  //creating a new
    MyBag
182     char b;
183     E values;
184     Object  element;
185
186      Iterator itr = aSet.iterator();
187     //iteratoring through the aSet and adding it to a string
188      while(itr.hasNext()) {
189          element = ((Keyed) itr.next()).getKey();
190          valuesOfStringSetB += element.toString();
191         }
192
193       Node<E> referenceLink = this.getTop(); // reference pointer to head
194
195       //looping through the linkedlisted to get the values
196            while (this.getTop() != null) {
197
198               element = this.getTop().item.getKey();
199                valuesOfStringSetA += element.toString();
200
```

```
201                 this.setTop(this.getTop().next); //pointing to the next node
202               }
203               this.setTop(referenceLink); // setting the pointer back to the head
204
205       /*Sorting the Array of both sets*/
206       unsortedSetA = valuesOfStringSetA.toCharArray();
207       unsortedSetB = valuesOfStringSetB.toCharArray();
208        Arrays.sort(unsortedSetA);
209        Arrays.sort(unsortedSetB);
210
211       //Checking the Array and check for duplicate values for the intersection
212       for(int i =unsortedSetA.length-1; i>=0; i--){
213          for(int j = unsortedSetA.length-1; j >=0; j--){
214             if(unsortedSetA[i] == unsortedSetB[j]){
215                  b = (char) unsortedSetA[i];
216                  values = (E) new KeyedChar(b);
217                  intersectionSet.add(values);
218
219             }//end if
220          }//end for
221         }
222      return intersectionSet;
223
224   }
225
226   @Override
227   public Iterator<E> iterator() {
228      return new LinkIterator<E>(this);
229   }
230
231   public Node<E> getTop() {
232      return top;
233   }
234
```

```
235    public void setTop(Node<E> top) {
236        this.top = top;
237    }
238
239    public int getLength() {
240        return length;
241    }
242
243    public void setLength(int length) {
244        this.length = length;
245    }
246
247 }
248
```