```java
package project4;

import java.io.Serializable;

/**********************************************************************
 * Double linked list created by me for use in RentalStore.
 * @author Gregory Huizenga
 * @version 7/26/2017
 * @param <E>
 *********************************************************************/
public class MyDoubleLinkedList<E> implements Serializable {

    /** the first node in the list */
    private DNode<E> top;

    /** the last node in the list */
    private DNode<E> tail;

    /** the size of the list */
    public int size;

    /**********************************************************************
     * default constructor for DLinkedList
     *********************************************************************/
    public MyDoubleLinkedList(){
        top = null;
        tail = null;
        size = 0;
    }

    /**********************************************************************
     * gets current size of list
     * @return size size of list
     *********************************************************************/
    public int size(){
        return size;
    }

    /**********************************************************************
     * clears all elements in list
     *********************************************************************/
    public void clear(){
        top = null;
        tail = null;
        size = 0;
    }

    /**********************************************************************
     * returns the element stored at the specified index
     * @param index
     * @return
     *********************************************************************/
    public Object get(int index){
        int i = 0;
        DNode<E> temp = top;

        // only executes if index is legal
```

```
58          if (index >= 0 && index < size){
59              while (i < index){
60                  temp = temp.getNext();
61              }
62          }else{
63              return null;
64          }
65
66          return temp.getData();
67      }
68
69      /**************************************************************
70       * removes all elements in list with specified value
71       * @param s the element to be removed
72       * @return boolean whether at least 1 element was removed
73       *************************************************************/
74      public boolean removeAll(E s){
75          DNode<E> temp = top;
76          int i = 0;
77          boolean removed = false;
78
79          //iterates until end of list
80          while (temp != null){
81              if (temp.getData() == s){
82                  remove(i);
83                  removed = true;
84              }
85              temp = temp.getNext();
86              i++;
87          }
88          return removed;
89      }
90
91      /**************************************************************
92       * searches list for specified element
93       * @param s the element to search for
94       * @return int the index of the element, -1 if not found
95       *************************************************************/
96      public int find(E s){
97          DNode<E> temp = top;
98          int i = 0;
99
100         // iterates until last element in list
101         while (temp != null){
102             if (temp.getData() == s){
103                 return i;
104             }
105             temp = temp.getNext();
106             i++;
107         }
108         return -1;
109     }
110
111     /**************************************************************
112      * removes the object at the specified index
113      * @param index index of object to be removed
114      * @return Object the object that was removed
```

```
115      ****************************************************************/
116     public Object remove(int index){
117         int i = 0;
118         DNode<E> temp = top;
119 
120         //only executes if index is legal
121         if (index >= 0 && index < size){
122             while (i < index){
123                 temp = temp.getNext();
124             }
125 
126             //case 0, first item
127             if (index == 0){
128                 top = top.getNext();
129                 size--;
130                 return temp.getData();
131             }
132 
133             //case 1, last item
134             if (index == size - 1){
135                 temp = tail;
136                 tail = tail.getPrevious();
137                 size--;
138                 return temp.getData();
139             }
140                 //case 2, item with next and previous
141 
142                 temp.getPrevious().setNext(temp.getNext());
143                 temp.getNext().setPrevious(temp.getPrevious());
144                 size--;
145                 return temp.getData();
146 
147         }
148         return null;
149     }
150 
151     /*****************************************************************
152      * adds specified element at beginning of list
153      * @param s the element to add
154      *****************************************************************/
155     public void addFirst (E s){
156 
157         //case 0, 1 element exists in list
158         if (size == 1){
159             tail = top;
160             top.setData(s);
161             top.setNext(tail);
162             tail.setPrevious(top);
163             size++;
164             return;
165         }
166 
167         //case 1, list is empty
168         if (size == 0){
169             top = new DNode<E>(s, null, null);
170             size++;
171             return;
```

```
172         }
173 
174         //case 2, list contains at least 2 elements
175         DNode<E> temp = new DNode(top.getData(), top.getNext(), top);
176         top.getNext().setPrevious(temp);
177         top.setNext(temp);
178         top.setData(s);
179 
180     }
181 
182     /**************************************************************
183      * adds specified element to the end of the list
184      * @param s the element to be added
185      *************************************************************/
186     public void add(E s){
187 
188         //case 0, list is empty
189         if (size == 0){
190             top = new DNode<E>(s, null, null);
191             size++;
192             return;
193         }
194 
195         //case 1, list has 1 element
196         if (size == 1){
197             tail = new DNode<E>(s, null, top);
198             top.setNext(tail);
199             size++;
200             return;
201         }
202 
203         //case 2, list has at least 2 elements
204         DNode<E> temp = new DNode(tail.getData(), null, tail.getPrevious());
205         tail.getPrevious().setNext(temp);
206         tail.setPrevious(temp);
207         temp.setNext(tail);
208         tail.setData(s);
209         size++;
210 
211     }
212 }
213 
```