```java
public class UnorderedArrayList extends ArrayListClass
{

    public UnorderedArrayList(int size)
    {
          super(size);
    }

    public UnorderedArrayList()
    {
        super();
    }

        //Copy constructor
    public UnorderedArrayList(UnorderedArrayList otherList)
    {
        super(otherList);
    }

        //Method to determine whether searchItem is in the list.
        //Postcondition: If searchItem is found, returns the location
        //               in the array where the searchItem is found;
        //               otherwise, returns -1.
    public int seqSearch(DataElement searchItem)
    {
          int loc;
          boolean found = false;

          for(loc = 0; loc < length; loc++)
              if(list[loc].equals(searchItem))
              {
                    found = true;
                    break;
              }

          if(found)
              return loc;
          else
              return -1;
    } //end seqSearch

        //Method to insert insertItem at the end
        //of the list. However, first the list is searched to
        //see whether the item to be inserted is already in the list.
        //Postcondition: list[length] = insertItem and length++
        //           If insertItem is already in the list or the list
        //           is full, an appropriate message is output.
   public void insert(DataElement insertItem)
    {
        int loc;

        if(length == 0)          //list is empty
           list[length++] = insertItem.getCopy();  //insert acopy the item
                                                   // andincrement the length
        else
              if(length == maxSize)
                System.err.println("Cannot insert in a full list.");
              else
              {
                  loc = seqSearch(insertItem);

                  if(loc == -1)   //the item to be inserted
                                  //does not exist in the list
                        list[length++] = insertItem.getCopy();
                  else
                        System.err.println("The item to be inserted is already in "
                                         + "the list. No duplicates are allowed.");
              }
    } //end insert

```

```
71          //Method to remove an item from the list.
72          //The parameter removeItem specifies the item to
73          //be removed.
74          //Postcondition: If removeItem is found in the list, it is
75          //               removed from the list and length is
76          //               decremented by one.
77      public void remove(DataElement removeItem)
78      {
79          int loc;
80
81            if(length == 0)
82                System.err.println("Cannot delete from an empty list.");
83            else
84            {
85                loc = seqSearch(removeItem);
86
87                if(loc != -1)
88                      removeAt(loc);
89                else
90                      System.out.println("The item to be deleted is "
91                                  + "not in the list.");
92            }
93      } //end remove
94
95      //*********************************************************************
96      //Override the method in ArrayListClass here in UnorderedArrayList
97      //in order to tune the base method only in the derived class f/ HW
98      //*********************************************************************
99
100         //Method to remove the item from the list at the position
101         //specified by location.
102         //Postcondition: The list element at list[location] is removed
103         //    and length is decremented by 1.
104         //    If location is out of range, an appropriate message
105         //    is printed.
106     public void removeAt(int location)
107     {
108        if(location < 0 || location >= length)
109           System.err.println("The location of the item to be removed "
110                 + "is out of range.");
111        else
112        {
113           // for(int i = location; i < length - 1; i++)    // removed
114           //    list[i] = list[i+1];
115           list[location] = list[length - 1];               // added
116           list[length - 1] = null;
117           length--;
118        }
119     } //end removeAt
120
121     //*********************************************************************
122     //Don't bother putting the abstract method in the base class as we
123     //only want to make changes in the derived class - therefore, just
124     //add the implementation for removeAll here...
125     //*********************************************************************
126
127     public void removeAll(DataElement removeItem)
128     {
129        int loc;
130
131        if(length == 0)
132           System.err.println("Cannot delete from an empty list.");
133        else
134        {
135           loc = 0;
136
137           while(loc < length)
138              if(list[loc].equals(removeItem))
139                 removeAt(loc);
140           else
```

```
141                loc++;
142             }
143          }
144
145          //*********************************************************************
146          //Add the min() and max() methods to the derived class only - this
147          //completes the last to parts of the HW - in book changed the base
148          //*********************************************************************
149
150          public DataElement min()
151          {
152             if(length == 0)
153             {
154                System.out.println("The list is empty. "
155                      + "Cannot return the smallest element.");
156                System.exit(0);
157             }
158
159             DataElement smallest = list[0];
160
161             for(int i = 1; i < length; i++)
162                if(smallest.compareTo(list[i]) > 0)
163                   smallest = list[i];
164             return smallest;
165          }
166
167          public DataElement max()
168          {
169             if(length == 0)
170             {
171                System.out.println("The list is empty. "
172                      + "Cannot return the smallest element.");
173                System.exit(0);
174             }
175
176             DataElement largest = list[0];
177
178             for(int i = 1; i < length; i++)
179                if(largest.compareTo(list[i]) < 0)
180                   largest = list[i];
181             return largest;
182          }
183
184    }
```