```java
import java.util.Comparator;

import components.map.Map;
import components.map.Map.Pair;
import components.sortingmachine.SortingMachine;
import components.sortingmachine.SortingMachine1L;

/**
 * Abstract class implementing the ToDoList interface. Subclasses can override
 * specific methods as needed.
 */
public abstract class ToDoListSecondary implements ToDoList {

    @Override
    public final void addEvent(String event) {
        if (ALL_EVENTS_IMP.hasKey(event) && this.getImportance(event) == -1) {
            this.addImportance(event, 0);
            this.addDate(event, 0);
        } else {
            ALL_EVENTS_IMP.add(event, 0);
            ALL_EVENTS_DATE.add(event, 0);
        }
    }

    @Override
    public final void addEvent(String event, int importance, int date) {
        if (ALL_EVENTS_IMP.hasKey(event) && ALL_EVENTS_IMP.value(event) == -1) {
            this.addImportance(event, importance);
            this.addDate(event, date);
        } else {
            ALL_EVENTS_IMP.add(event, importance);
            ALL_EVENTS_DATE.add(event, date);
        }
    }

    @Override
    public final void removeEvent(String event, int dateRem) {
        this.addImportance(event, -1);
        this.addDate(event, dateRem);
    }

```

```
42      /**
43       * Comparator for comparing pairs of strings and integers based
    on the
44       * integer values. Comparator for sorting importance in
    orderByImportance.
45       */
46      private static final class CompareImportance
47              implements Comparator<Map.Pair<String, Integer>> {
48
49          @Override
50          public int compare(Map.Pair<String, Integer> o1,
51                  Map.Pair<String, Integer> o2) {
52              return o2.value().compareTo(o1.value());
53          }
54
55      }
56
57      @Override
58      public final void orderByImportance() {
59          Comparator<Map.Pair<String, Integer>> order = new
    CompareImportance();
60          SortingMachine<Map.Pair<String, Integer>> sortByImp = new
    SortingMachine1L<>(
61                  order);
62
63          if (SORTED_BY_DATE.length() > 0) {
64              while (SORTED_BY_DATE.length() != 0) {
65                  Map.Pair<String, Integer> rem =
    SORTED_BY_DATE.dequeue();
66                  ALL_EVENTS_DATE.add(rem.key(), rem.value());
67              }
68          }
69
70          int size = ALL_EVENTS_IMP.size();
71          for (int i = 0; i < size; i++) {
72              Pair<String, Integer> temp =
    ALL_EVENTS_IMP.removeAny();
73              sortByImp.add(temp);
74          }
75
76          sortByImp.changeToExtractionMode();
77
78          for (int i = 0; i < size; i++) {
79              SORTED_BY_IMP.enqueue(sortByImp.removeFirst());
80          }
```

```
 81
 82     }
 83
 84     /**
 85      * Comparator for comparing pairs of strings and integers based
    on the
 86      * integer values. Comparator for sorting date in orderByDate.
 87      */
 88     private static final class CompareDate
 89             implements Comparator<Map.Pair<String, Integer>> {
 90
 91         @Override
 92         public int compare(Map.Pair<String, Integer> o1,
 93                 Map.Pair<String, Integer> o2) {
 94             return o1.value().compareTo(o2.value());
 95         }
 96
 97     }
 98
 99     @Override
100     public final void orderByDate() {
101         Comparator<Map.Pair<String, Integer>> order = new
    CompareDate();
102         SortingMachine<Map.Pair<String, Integer>> sortByDate = new
    SortingMachine1L<>(
103                 order);
104
105         if (SORTED_BY_IMP.length() > 0) {
106             while (SORTED_BY_IMP.length() != 0) {
107                 Map.Pair<String, Integer> rem =
    SORTED_BY_IMP.dequeue();
108                 ALL_EVENTS_IMP.add(rem.key(), rem.value());
109             }
110         }
111
112         int size = ALL_EVENTS_DATE.size();
113         for (int i = 0; i < size; i++) {
114             Pair<String, Integer> temp =
    ALL_EVENTS_DATE.removeAny();
115             sortByDate.add(temp);
116         }
117
118         sortByDate.changeToExtractionMode();
119
120         for (int i = 0; i < size; i++) {
```

```
121             SORTED_BY_DATE.enqueue(sortByDate.removeFirst());
122         }
123     }
124 
125     @Override
126     public final String displayEventsNeedToDo() {
127         String ans = "";
128         for (Map.Pair<String, Integer> i : ALL_EVENTS_IMP) {
129             if (!(i.value() == -1)) {
130                 ans = ans + i.key() + "\n";
131             }
132         }
133         for (Map.Pair<String, Integer> i : SORTED_BY_IMP) {
134             if (!(i.value() == -1)) {
135                 ans = ans + i.key() + "\n";
136             }
137         }
138         return ans;
139     }
140 
141     @Override
142     public final String displayEventsCompleted() {
143         String ans = "";
144         for (Map.Pair<String, Integer> i : ALL_EVENTS_IMP) {
145             if ((i.value() == -1)) {
146                 ans = ans + i.key() + "\n";
147             }
148         }
149         for (Map.Pair<String, Integer> i : SORTED_BY_IMP) {
150             if ((i.value() == -1)) {
151                 ans = ans + i.key() + "\n";
152             }
153         }
154         return ans;
155     }
156 
157     @Override
158     public final String toString() {
159 
160         String ans = "To-Do-List \nEvent, Importance or Completed,
    "
161                 + "Date Completed or Date to be Completed\n";
162         String imp = "";
163         String date = "";
164         if (SORTED_BY_DATE.length() == 0 && SORTED_BY_IMP.length()
```

```
    == 0) {
165             for (Map.Pair<String, Integer> i : ALL_EVENTS_IMP) {
166                 imp =
    this.getStringImportance(this.getImportance(i.key()));
167                 date = this.getStringDate(this.getDate(i.key()));
168                 ans = ans + i.key() + ", " + imp + ", " + date +
    "\n";
169             }
170
171         }
172
173         if (SORTED_BY_IMP.length() > 0 && SORTED_BY_DATE.length()
    == 0) {
174             this.orderByImportance();
175             for (Map.Pair<String, Integer> i : SORTED_BY_IMP) {
176                 imp = this.getStringImportance(i.value());
177                 date = this.getStringDate(this.getDate(i.key()));
178                 ans = ans + i.key() + ", " + imp + ", " + date +
    "\n";
179             }
180         }
181
182         if (SORTED_BY_DATE.length() > 0 && SORTED_BY_IMP.length()
    == 0) {
183             this.orderByDate();
184             for (Map.Pair<String, Integer> i : SORTED_BY_DATE) {
185                 imp =
    this.getStringImportance(this.getImportance(i.key()));
186                 date = this.getStringDate(i.value());
187                 ans = ans + i.key() + ", " + imp + ", " + date +
    "\n";
188             }
189
190         }
191         return ans;
192
193     }
194
195 }
196
```