```java
import components.map.Map1L;
import components.sequence.Sequence;
import components.sequence.Sequence1L;
import components.simplewriter.SimpleWriter;
import components.simplewriter.SimpleWriter1L;

public class CollegeBasketballTeamComponent {

    private Map1L<String, Map1L<Integer, String>> teamData;

    private void createNewRep() {
        this.teamData = new Map1L<String, Map1L<Integer, String>>();
    }

    /**
     * No-argument constructor.
     */
    public CollegeBasketballTeamComponent() {
        this.createNewRep();
    }

    private int turnValueToInt(String value) {
//          since some statistics can't directly be converted into a integer
//          value, this method will convert them for us.

//          for example: Statistic Category: Win/Loss, Statistic Rank: 10, Statistic Value: "10-2"
//          this method will convert the win to loss ratio into a percentage: 10/12 = 0.833
//          that way all stats can be used for some sort of analysis
        return 5;
    }

    // Kernel Methods

    public void getAllStatistics(String teamName) {
//          use webscraping tools to get desired statistics
//          will use https://www.teamrankings.com/ncb/stats/
//          since it has a bunch of stats already formatted in nice tables
//          that can be scraped. the parameter teamName will have to match with
//          one of the names in the table from the site in order
//          for this to be able to work

//          Use a web scraping tool/library like Jsoup to fetch the HTML content of the provided URL. Example: https://www.teamrankings.com/ncaa-basketball/stat/points-per-game.

//          Parse the HTML content to extract the desired statistics for the specified teamName.

//          Identify the HTML elements that contain the statistical data for the specified team. This may involve inspecting the HTML structure of the webpage.

//          Extract the relevant statistical data (e.g., points per game, field goal percentage, rebounds per game) for the specified teamName from the HTML elements.

//          Store the extracted statistics in the teamData map using appropriate keys and values, such as statistical categories as keys and the corresponding values for each category.
    }

    public void addCustomStatistic(String category, int rank, String value) {

```

```
55          if (this.teamData.hasKey(category)) {
56              Map1L<Integer, String> newStatistic = new Map1L<>();
57              newStatistic.add(rank, value);
58              this.teamData.replaceValue(category, newStatistic);
59          } else {
60              Map1L<Integer, String> newCategoryData = new Map1L<>();
61              newCategoryData.add(rank, value);
62              this.teamData.add(category, newCategoryData);
63          }
64      }
65
66      public void removeStatistic(String category) {
67          if (this.teamData.hasKey(category)) {
68              this.teamData.remove(category);
69          } else {
70  //              error not found or use an assert statement at the beginning
71          }
72      }
73
74      public Map1L<Integer, String> getStatisticsByCategory(String category) {
75  //          returns a map with the key being the statistics rank, and the value of the
    map being the value of the value of the statistic
76          return this.teamData.value(category);
77
78      }
79
80      public Sequence<String> getAllCategories() {
81  //          return a list of the all the keys in this
82
83          Sequence<String> categories = new Sequence1L<>();
84
85          for (Map1L.Pair<String, Map1L<Integer, String>> pair : this.teamData) {
86              categories.add(0, pair.key());
87          }
88
89          return categories;
90      }
91
92      public void bestAndWorstStatistics() {
93          SimpleWriter out = new SimpleWriter1L();
94          Map1L<String, Integer> bestStatistics = new Map1L<>();
95          Map1L<String, Integer> worstStatistics = new Map1L<>();
96
97          // Iterate over the teamData map to find the best and worst statistics
98          for (Map1L.Pair<String, Map1L<Integer, String>> team : this.teamData) {
99              String category = team.key();
100             Map1L<Integer, String> teamValues = team.value();
101             for (Map1L.Pair<Integer, String> teamPair : teamValues) {
102                 int rank = teamPair.key();
103
104                 // Logic for best statistics
105                 if (bestStatistics.size() < 5) {
106                     bestStatistics.add(category, rank);
107                 } else {
108                     int worstRank = Integer.MAX_VALUE;
109                     String worstCategory = null;
110                     for (Map1L.Pair<String, Integer> bestStats : bestStatistics) {
111                         int currentRank = bestStats.value();
112                         if (currentRank < worstRank) {
113                             worstRank = currentRank;
114                             worstCategory = bestStats.key();
115                         }
116                     }
```

```
117                     if (rank > worstRank) {
118                         bestStatistics.remove(worstCategory);
119                         bestStatistics.add(category, rank);
120                     }
121                 }
122 
123                 // Logic for worst statistics
124                 if (worstStatistics.size() < 5) {
125                     worstStatistics.add(category, rank);
126                 } else {
127                     int bestRank = -1; // Initialize to minimum possible value
128                     String bestCategory = null;
129                     for (Map1L.Pair<String, Integer> worstStats : worstStatistics) {
130                         int currentRank = worstStats.value();
131                         if (currentRank > bestRank) {
132                             bestRank = currentRank;
133                             bestCategory = worstStats.key();
134                         }
135                     }
136                     if (rank < bestRank) {
137                         worstStatistics.remove(bestCategory);
138                         worstStatistics.add(category, rank);
139                     }
140                 }
141             }
142         }
143 
144         // Print out the best statistics names
145         out.println("Top 5 statistics:");
146         for (Map1L.Pair<String, Integer> bestStat : bestStatistics) {
147             out.println(bestStat.key());
148         }
149 
150         // Print out the worst statistics names
151         out.println("Bottom 5 statistics:");
152         for (Map1L.Pair<String, Integer> worstStat : worstStatistics) {
153             out.println(worstStat.key());
154         }
155 
156         out.close();
157     }
158 
159     // Secondary Methods
160 
161     public static void runSimulation(String team1, String team2) {
162 // using the stats from each team and all the methods above, come up with a method to try and predict the
163 // outcome of the game, and also use some sort of randomness so each simulation
164 // isn't the same
165     }
166 
167     public static void main(String[] args) {
168         CollegeBasketballTeamComponent ohioState = new CollegeBasketballTeamComponent();
169         CollegeBasketballTeamComponent nebraska = new CollegeBasketballTeamComponent();
170 
171         ohioState.getAllStatistics("Ohio State");
172         ohioState.addCustomStatistic("3PT Percentage", 25, "37%");
173         ohioState.removeStatistic("Rebounds Per Game");
174 
175         ohioState.bestAndWorstStatistics(); // can be used to compare the strengths and weakness of each team
```

```
176         nebraska.bestAndWorstStatistics();
177
178         runSimulation("Ohio State", "Nebraska");
179
180     }
181 }
182
```