```java
import java.util.Random;

public abstract class CollegeBasketballTeamComponentSecondary
        implements CollegeBasketballTeam {

    @Override
    public void runSimulation(CollegeBasketballTeam team2) {

        SimpleWriter out = new SimpleWriter1L();

        // Retrieve statistics for both teams

        Sequence1L<StatCategory> categories1 =
this.getAllCategories();
        Sequence1L<StatCategory> categories2 =
team2.getAllCategories();

        // Calculate score for team 1
        double score1 = 0.0;
        for (StatCategory category : categories1) {
            Map1L<Integer, Double> statistics = this
                    .getStatisticsByCategory(category);
            for (Map1L.Pair<Integer, Double> entry :
statistics) {
                score1 += entry.value();
            }
        }

        // Calculate score for team 2
        double score2 = 0.0;
        for (StatCategory category : categories2) {
            Map1L<Integer, Double> statistics = team2
                    .getStatisticsByCategory(category);
            for (Map1L.Pair<Integer, Double> entry :
statistics) {
                score2 += entry.value(); // Accumulate scores
for all statistics
            }
        }

        // Introduce some randomness
        Random random = new Random();
        score1 *= random.nextDouble(); // Scale the score by a
random factor between 0 and 1
```

```
44          score2 *= random.nextDouble();
45
46          // Determine the winner
47          String winner;
48          if (score1 > score2) {
49              winner = "Team 1";
50          } else if (score1 < score2) {
51              winner = "Team 2";
52          } else {
53              winner = "Tie"; // Handle tie scenario
54          }
55
56          // Print simulation results
57          out.println("Simulation Results:");
58          out.println("Team 1 Score: " + score1);
59          out.println("Team 2 Score: " + score2);
60          out.println("Winner: " + winner);
61          out.close();
62      }
63
64      @Override
65      public String toString() {
66          StringBuilder sb = new StringBuilder();
67          sb.append("Team Statistics:\n");
68          Sequence1L<StatCategory> categories =
    this.getAllCategories();
69          for (StatCategory category : categories) {
70              sb.append(category).append(":\n");
71              Map1L<Integer, Double> statistics = this
72                      .getStatisticsByCategory(category);
73              for (Map1L.Pair<Integer, Double> statistic :
    statistics) {
74                  sb.append("Rank
    ").append(statistic.key()).append(": ")
75                          .append(statistic.value()).append("\n")
    ;
76              }
77          }
78          return sb.toString();
79      }
80
81      @Override
82      public boolean equals(Object team2) {
83          // Filter code for special cases
```

```
 84         if (team2 == this) {
 85             return true;
 86         }
 87         if (team2 == null) {
 88             return false;
 89         }
 90         if (!(team2 instanceof CollegeBasketballTeam)) {
 91             return false;
 92         }
 93 
 94         CollegeBasketballTeam thatTeam =
    (CollegeBasketballTeam) team2;
 95 
 96         // Check if the teams have the same statistics data
 97 
 98         // Check if the categories are the same
 99         Sequence1L<StatCategory> team1Categories =
    this.getAllCategories();
100         Sequence1L<StatCategory> team2Categories =
    thatTeam.getAllCategories();
101 
102         if (!team1Categories.equals(team2Categories)) {
103             return false;
104         }
105 
106         // Check if the statistics for each category are the
    same
107         for (StatCategory category : team1Categories) {
108             Map1L<Integer, Double> thisStats = this
109                     .getStatisticsByCategory(category);
110             Map1L<Integer, Double> thoseStats = thatTeam
111                     .getStatisticsByCategory(category);
112 
113             if (!thisStats.equals(thoseStats)) {
114                 return false;
115             }
116         }
117 
118         return true;
119 
120     }
121 
122     @Override
123     public int hashCode() {
```

```
124         int hashCode = 0;
125 
126         // Iterate through each category of statistics data
127         // then add the rank of the team to the hashCode value.
128         Sequence1L<StatCategory> categories =
    this.getAllCategories();
129         for (StatCategory category : categories) {
130             Map1L<Integer, Double> stats = this
131                     .getStatisticsByCategory(category);
132             for (Map1L.Pair<Integer, Double> pair : stats) {
133                 int stat = pair.key();
134                 hashCode += stat;
135             }
136         }
137 
138         return hashCode;
139     }
140 
141 }
142 
```