1 Race.java

```java
package cycling;

import java.util.HashMap;
import java.util.HashSet;

public class Race {

   /**
    * All the races in the system.
    *
    *
    * @author Kechen Liu
    * @version 1.0
    */
   public static Integer raceCounter = 1;
   private int raceID;
   private String raceName;
   private String raceDesc;
   // store all the stages in one race
   private HashMap<Integer, Stage> raceStages = new HashMap<Integer, Stage>();

   // constructor
   Race(String raceName) {
      this.setRaceName(raceName);
      this.setRaceDesc(null);
      this.setRaceID(raceCounter);
      raceCounter++;
   }

   Race(String raceName, String raceDesc) {
      this.setRaceName(raceName);
      this.setRaceDesc(raceDesc);
      this.setRaceID(raceCounter);
      raceCounter++;
   }

   // methods:getter& setter
   public int getRaceID() {
      return this.raceID;
   }

   public void setRaceID(int raceID) {
      this.raceID = raceID;
   }

   public String getRaceName() {
      return this.raceName;
   }

   public void setRaceName(String raceName) {
      this.raceName = raceName;
   }
```

```
53
54      public String getRaceDesc() {
55          return this.raceDesc;
56      }
57
58      public void setRaceDesc(String raceDesc) {
59          this.raceDesc = raceDesc;
60      }
61
62      public HashMap<Integer, Stage> getRaceStages() {
63          return this.raceStages;
64      }
65
66      public void setRaceStages(HashMap<Integer, Stage> rs) {this.raceStages = rs;}
67
68      public double showRaceLength() {
69          double totallength = 0.0;
70          for (Stage stage : this.raceStages.values()) {
71              totallength += stage.getStageLength();
72          }
73          return totallength;
74      }
75
76      public boolean stageNameCheck(String name) {
77          HashSet<String> names = new HashSet<String>();
78          for (Integer i :raceStages.keySet()) {
79              names.add(raceStages.get(i).getStageName());
80          }
81          return names.contains(name) == true;
82      }
83
84
85
86  }
```

2 Stage.java

```
1   package cycling;
2
3   import java.time.LocalDateTime;
4   import java.time.LocalTime;
5   import java.util.ArrayList;
6   import java.util.HashMap;
7   import java.util.Map;
8
9   public class Stage {
10      /**
11       * All the stages in the system.
12       *
13       * @author Kechen Liu
14       * @version 1.0
15       */
16      public static int stageCounter = 1;
17      private int stageID;
```

```
18      private String stageName;
19      private String stageDesc;
20      private double stageLength;
21      private StageType type;
22      private LocalDateTime startTime;
23      private String stageState;
24      private int raceID;
25      private HashMap<Integer, Segment> stageSegments = new HashMap<Integer, Segment>();
26      // store all the riders in one stage
27      private HashMap<Integer, Rider> stageRiders = new HashMap<Integer, Rider>();
28
29      // constructor
30      Stage(int raceId, String stageName, String stageDesc, double stageLength, LocalDateTime startTime,
            StageType type) {
31          this.setRaceID(raceId);
32          this.setStageName(stageName);
33          this.setStageDesc(stageDesc);
34          this.setStageID(stageCounter);
35          this.setStageLength(stageLength);
36          this.setStageType(type);
37          this.setStartTime(startTime);
38          this.setStageState(null);
39          stageCounter++;
40      }
41
42      // getters and setters
43      public int getraceID() {
44          return this.raceID;
45      }
46
47      public void setRaceID(int raceID) {
48          this.raceID = raceID;
49      }
50
51      public int getStageID() {
52          return this.stageID;
53      }
54
55      public void setStageID(int stageID) {
56          this.stageID = stageID;
57      }
58
59      public String getStageName() {
60          return this.stageName;
61      }
62
63      public void setStageName(String stageName) {
64          this.stageName = stageName;
65      }
66
67      public String getStageDesc() {
68          return this.stageDesc;
69      }
70
71      public void setStageDesc(String stageDesc) {
```

```
72          this.stageDesc = stageDesc;
73      }
74
75      public double getStageLength() {
76          return this.stageLength;
77      }
78
79      public void setStageLength(double stageLength) {
80          this.stageLength = stageLength;
81      }
82
83      public StageType getStageType() {
84          return this.type;
85      }
86
87      public void setStageType(StageType type) {
88          this.type = type;
89      }
90
91      public LocalDateTime getStartTime() {
92          return this.startTime;
93      }
94
95      public void setStartTime(LocalDateTime startTime) {
96          this.startTime = startTime;
97      }
98
99      public String getStageState() {
100         return this.stageState;
101     }
102
103     public HashMap<Integer, Segment> getStageSegments() {
104         return this.stageSegments;
105     }
106
107     public void setStageState(String stageState) {
108         this.stageState = stageState;
109     }
110
111     public HashMap<Integer, Rider> getStageRiders() {
112         return this.stageRiders;
113     }
114
115     /**
116      * Get the rider's result in the corresponding stage
117      * @param st the stage user wants to find result.
118      * @return A HashMap of riderId and corresponding result
119      */
120     public static HashMap<Integer, Result> getRiderResultInStage(Stage st) {
121         int stageId = st.stageID;
122         // riderId <stageId, result>
123         HashMap<Integer, HashMap<Integer, Result>> rsr = new HashMap<>();
124         HashMap<Integer, Rider> riders = st.getStageRiders();
125
126         for (Map.Entry<Integer, Rider> mapElement : riders.entrySet()) {
```

```
127             rsr.put(mapElement.getKey(), mapElement.getValue().getResults());
128         }
129 
130         // riderId, result
131         HashMap<Integer, Result> rr = new HashMap<>();
132         for (Map.Entry<Integer, HashMap<Integer, Result>> m : rsr.entrySet()) {
133             if (m.getValue().containsKey(stageId)) {
134                 rr.put(m.getKey(), m.getValue().get(stageId));
135             }
136         }
137         return rr;
138     }
139 
140     /**
141      *  Get Ranked riderIds
142      * @param rt HashMap of riderId and LocalTime(can be eclipsed time or adjusted time
143      * @return a ranked int array of riders' ids.
144      */
145     public static int[] getRankedIds(HashMap<Integer, LocalTime> rt) {
146         ArrayList<Integer> listRids = new ArrayList<>();
147         // iterate over and get keys and values
148         for (Map.Entry<Integer, LocalTime> m : rt.entrySet()) {
149             listRids.add(m.getKey());
150         }
151         int[] ids = new int[listRids.size()];
152         for (int i = 0; i < listRids.size(); i++) {
153             int e = listRids.get(i);
154             ids[i] = e;
155         }
156         return ids;
157     }
158 
159     /**
160      * Get ranked LocalTimes
161      * @param rt HashMap of riderId and LocalTime(can be eclipsed time or adjusted time
162      * @return a ranked LocalTime array of riders' time
163      */
164     public static LocalTime[] getRankedTimes(HashMap<Integer, LocalTime> rt) {
165         ArrayList<LocalTime> listTimes = new ArrayList<>();
166         // iterate over and get keys and values
167         for (Map.Entry<Integer, LocalTime> m : rt.entrySet()) {
168             listTimes.add(m.getValue());
169         }
170         LocalTime[] times = new LocalTime[listTimes.size()];
171         for (int i = 0; i < listTimes.size(); i++) {
172             LocalTime e = listTimes.get(i);
173             times[i] = e;
174         }
175         return times;
176     }
177 
178 }
```

3 Segment.java

```java
package cycling;

public class Segment {
   /**
    * All the segments in the system.
    *
    *
    * @author Kechen Liu
    * @version 1.0
    */
   private Double location;

   private int segmentId;

   private int stageId;

   private SegmentType type;

   public static int segmentCounter = 1;

   // constructor
   public Segment() {
   }

   // get&set
   public Double getLocation() {
      return this.location;
   }

   public void setLocation(Double location) {
      this.location = location;
   }

   public SegmentType gettype() {
      return this.type;
   }

   public void setType(SegmentType type) {
      this.type = type;
   }

   public int getStageId() {
      return this.stageId;
   }

   public void setStageId(int stageId) {
      this.stageId = stageId;
   }

   public int getSegmentId() {
      return this.segmentId;
   }
```

```
53
54      public void setSegmentId(int segmentId) {
55         this.segmentId = segmentId;
56      }
57
58
59      /**
60       * assign rider's point in stage based on their rank in stage and stage type
61       * @param riderRank rider's rank in stage, call portal's getRiderRank method to get
62       * @param type stage's type
63       * @return int array of rider's points, corresponding to rider's id
64       */
65      public static int[] assignRiderPointsInStage(int[] riderRank, StageType type) {
66         int[] points = new int[riderRank.length];
67         if (type == StageType.FLAT) {
68            for (int i = 0; i < riderRank.length; i++) {
69               int n = riderRank[i];
70               switch (n) {
71                  case 1 -> points[i] = 50;
72                  case 2 -> points[i] = 30;
73                  case 3 -> points[i] = 20;
74                  case 4 -> points[i] = 18;
75                  case 5 -> points[i] = 16;
76                  case 6 -> points[i] = 14;
77                  case 7 -> points[i] = 12;
78                  case 8 -> points[i] = 10;
79                  case 9 -> points[i] = 8;
80                  case 10 -> points[i] = 7;
81                  case 11 -> points[i] = 6;
82                  case 12 -> points[i] = 5;
83                  case 13 -> points[i] = 4;
84                  case 14 -> points[i] = 3;
85                  case 15 -> points[i] = 2;
86               }
87            }
88         }
89         if (type == StageType.TT
90               || type == StageType.HIGH_MOUNTAIN) {
91            for (int i = 0; i < riderRank.length; i++) {
92               int n = riderRank[i];
93               switch (n) {
94                  case 1 -> points[i] = 20;
95                  case 2 -> points[i] = 17;
96                  case 3 -> points[i] = 15;
97                  case 4 -> points[i] = 13;
98                  case 5 -> points[i] = 11;
99                  case 6 -> points[i] = 10;
100                 case 7 -> points[i] = 9;
101                 case 8 -> points[i] = 8;
102                 case 9 -> points[i] = 7;
103                 case 10 -> points[i] = 6;
104                 case 11 -> points[i] = 5;
105                 case 12 -> points[i] = 4;
106                 case 13 -> points[i] = 3;
107                 case 14 -> points[i] = 2;
```

```
                case 15 -> points[i] = 1;
            }
        }

    }

    if (type == StageType.MEDIUM_MOUNTAIN) {
        for (int i = 0; i < riderRank.length; i++) {
            int n = riderRank[i];
            switch (n) {
                case 1 -> points[i] = 30;
                case 2 -> points[i] = 25;
                case 3 -> points[i] = 22;
                case 4 -> points[i] = 19;
                case 5 -> points[i] = 17;
                case 6 -> points[i] = 15;
                case 7 -> points[i] = 13;
                case 8 -> points[i] = 11;
                case 9 -> points[i] = 9;
                case 10 -> points[i] = 7;
                case 11 -> points[i] = 6;
                case 12 -> points[i] = 5;
                case 13 -> points[i] = 4;
                case 14 -> points[i] = 3;
                case 15 -> points[i] = 2;
            }
        }
    }
    return points;
  }

}
```

4 Sprint.java

```
package cycling;

public class Sprint extends Segment{

    public Sprint(int stageId, double location) {
        this.setStageId(stageId);
        this.setLocation(location);
        this.setType(SegmentType.SPRINT);
        this.setSegmentId(segmentCounter);
        segmentCounter++;
    }
}
```

5 Mountains.java

```
package cycling;

/* Mountain segment extend segment. */
public class Mountains extends Segment {
```

```
5
6       private Double averageGradient;
7
8       private Double length;
9
10      public Double getLength() {
11          return this.length;
12      }
13
14      public void setLength(Double length) {
15          this.length = length;
16      }
17
18      public Double getAverageGradient() {
19          return this.averageGradient;
20      }
21
22      public void setAverageGradient(Double averageGradient) {
23          this.averageGradient = averageGradient;
24      }
25
26      public void setAverageGrandient(Double averageGrandient) {
27          this.averageGradient = averageGrandient;
28      }
29
30      public Mountains(int stageId, Double location, SegmentType type, Double averageGragient, Double length)
            {
31          this.setStageId(stageId);
32          this.setLocation(location);
33          this.setType(type);
34          this.setAverageGradient(averageGragient);
35          this.setLength(length);
36          this.setSegmentId(segmentCounter);
37          segmentCounter++;
38      }
39
40      public Mountains(int stageId, Double location) {
41      }
42  }
```

6 Team.java

```
1   package cycling;
2
3   import java.util.HashMap;
4
5   public class Team {
6      /**
7       * All the teams in the system.
8       *
9       *
10      * @author Kechen Liu
11      * @version 1.0
12      */
```

```
13    static int teamCounter = 1;
14    private int teamID;
15    private String teamName;
16    private String teamDesc;
17    // store all the riders in one team
18    private HashMap<Integer, Rider> teamRiders = new HashMap<Integer, Rider>();
19
20    Team(String teamName, String teamDesc) {
21       this.setTeamName(teamName);
22       this.setTeamDesc(teamDesc);
23       this.setTeamID(teamCounter);
24       teamCounter++;
25    }
26
27    // getter and setter methods
28    public int getTeamID() {
29       return this.teamID;
30    }
31
32    public void setTeamID(int teamID) {
33       this.teamID = teamID;
34    }
35
36    public String getTeamName() {
37       return this.teamName;
38    }
39
40    public void setTeamName(String teamName) {
41       this.teamName = teamName;
42    }
43
44    public String getTeamDesc() {
45       return this.teamDesc;
46    }
47
48    public void setTeamDesc(String teamDesc) {
49       this.teamDesc = teamDesc;
50    }
51
52    public HashMap<Integer, Rider> getTeamRiders() {
53       return this.teamRiders;
54    }
55
56 }
```

7 Rider.java

```
1  package cycling;
2
3  import java.util.ArrayList;
4  import java.util.HashMap;
5
6  public class Rider {
7     /**
```

```
     * All the rider in the system.
     *
     *
     * @author Kechen Liu
     * @version 1.0
     */
    private String name;
    private int yearOfBirth;
    public static int riderIDCounter = 1;
    private int riderID;
    private int teamId;
    private ArrayList<Integer> stageIds = new ArrayList<Integer>();

    private int riderPoints;
    private int riderMountainPoints;

    private HashMap<Integer, Result> results;

    // constructor
    public Rider(int teamId, String name, int yearOfBirth) {
       this.teamId = teamId;
       this.setRiderName(name);
       this.setYearOfBirth(yearOfBirth);
       this.riderID = riderIDCounter;
       riderIDCounter++;
       // to do
       this.results = new HashMap<>();
    }

    // method: getter& setter
    public String getRiderName() {
       return this.name;
    }

    public void setRiderName(String name) {
       this.name = name;
    }

    public int getYearOfBirth() {
       return this.yearOfBirth;
    }

    public void setYearOfBirth(int yearOfBirth) {
       this.yearOfBirth = yearOfBirth;
    }

    public int getRiderId() {
       return this.riderID;
    }

    public void setRiderId(String name) {
       this.name = name;
    }

    public int getTeamId() {
```

```
63        return this.teamId;
64    }
65
66    public void setTeamId(int teamId) {
67        this.teamId = teamId;
68    }
69
70    public ArrayList<Integer> getStageIds() {
71        return this.stageIds;
72    }
73
74    public void setStageId(ArrayList<Integer> stageIds) {
75        this.stageIds = stageIds;
76    }
77
78    public int getRiderPoints() {
79        return this.riderPoints;
80    }
81
82    public void setRiderPoints(int riderPoints) {
83        this.riderPoints = riderPoints;
84    }
85
86    public int getRiderMountainPoints() {
87        return this.riderMountainPoints;
88    }
89
90    public void setRiderMountainPoint(int riderMountainPoints) {
91        this.riderMountainPoints = riderMountainPoints;
92    }
93
94    public HashMap<Integer, Result> getResults() {
95        return this.results;
96    }
97
98    public void setResult(HashMap<Integer, Result> results) {
99        this.results = results;
100   }
101
102 }
```

8 Result.java

```
1  package cycling;
2
3  import java.time.Duration;
4  import java.time.Instant;
5  import java.time.LocalDateTime;
6  import java.time.LocalTime;
7  import java.time.ZoneId;
8  import java.util.ArrayList;
9  import java.util.Collections;
10 import java.util.Comparator;
11 import java.util.HashMap;
```

```java
import java.util.LinkedHashMap;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;

public class Result {
    private LocalTime[] checkpoints;
    private LocalTime totalElapsedTime;
    private LocalTime adjustedElapsedTime;

    public LocalTime[] getCheckPoints() {
        return this.checkpoints;
    }

    public void setCheckPoints(LocalTime[] checkpoints) {
        this.checkpoints = checkpoints;
    }

    public LocalTime getAdjustedElapsedTime() {
        return this.adjustedElapsedTime;
    }

    public void setAdjustedElapsedTime(LocalTime adjustedElapsedTime) {
        this.adjustedElapsedTime = adjustedElapsedTime;
    }

    public LocalTime getTotalElapsedTime() {
        return this.totalElapsedTime;
    }

    public void setTotalElapsedTime(LocalTime totalElapsedTime) {
        this.totalElapsedTime = totalElapsedTime;
    }

    public Result(LocalTime[] checkp) {
        checkpoints = checkp;
        totalElapsedTime = calculateTotalElapsedTime();
    }

    /** method to find the smallest finish time in a stage. */
    public static LocalTime findSmallTime(Stage st) {
        int stageId = st.getStageID();
        HashMap<Integer, Rider> allRidersInStage = st.getStageRiders();
        List<LocalTime> ftList = new ArrayList<>();

        for (Map.Entry<Integer, Rider> mapElement : allRidersInStage.entrySet()) {
            LocalTime[] checkpoints = mapElement.getValue().getResults().get(stageId).getCheckPoints();
            LocalTime finishTime = checkpoints[checkpoints.length - 1];
            ftList.add(LocalTime.parse(finishTime.toString()));
        }
        return ftList.get(0);
    }

    public LocalTime calculateTotalElapsedTime() {
        LocalTime et = calcTime(checkpoints[0], checkpoints[checkpoints.length - 1]);
```

```
67          return et;
68      }
69
70      public LocalTime calculateAdjustedElapsedTime(LocalTime smallest) {
71          this.adjustedElapsedTime = calcTime(checkpoints[0], smallest);
72          return adjustedElapsedTime;
73      }
74
75      /* method to calculate the difference between start and end. */
76      public static LocalTime calcTime(LocalTime start, LocalTime end) {
77          Duration time = Duration.between(start, end);
78          long longd = time.toMillis();
79          LocalTime newTime = LocalDateTime.ofInstant(Instant.ofEpochMilli(longd), ZoneId.systemDefault())
80                  .toLocalTime();
81          return newTime;
82      }
83
84      /**
85       * method to get rider's eclipsed finish time
86       *
87       * @param rr HashMap of RiderId and the corresponding result.
88       * @return HashMap of RiderId and corresponding eclipsed finish time.
89       */
90      public static HashMap<Integer, LocalTime> getET(HashMap<Integer, Result> rr) {
91          HashMap<Integer, LocalTime> et = new HashMap<>();
92          for (Map.Entry<Integer, Result> m : rr.entrySet()) {
93              et.put(m.getKey(), m.getValue().getTotalElapsedTime());
94          }
95          return et;
96      }
97
98      /**
99       * method to get rider's adjusted finish time
100      *
101      * @param rr HashMap of RiderId and the corresponding result.
102      * @return HashMap of RiderId and corresponding adjusted finish time.
103      */
104     public static HashMap<Integer, LocalTime> getAT(HashMap<Integer, Result> rr) {
105         HashMap<Integer, LocalTime> et = new HashMap<>();
106         for (Map.Entry<Integer, Result> m : rr.entrySet()) {
107             et.put(m.getKey(), m.getValue().getAdjustedElapsedTime());
108         }
109         return et;
110     }
111
112     /**
113      * method to get rider's adjusted finish time
114      *
115      * @param rr HashMap of RiderId and the corresponding result.
116      * @return HashMap of RiderId and corresponding adjusted finish time.
117      */
118     public static HashMap<Integer, LocalTime> getFT(HashMap<Integer, Result> rr) {
119         HashMap<Integer, LocalTime> ft = new HashMap<>();
120         for (Map.Entry<Integer, Result> m : rr.entrySet()) {
121             LocalTime finishTime = m.getValue().getCheckPoints()[m.getValue().getCheckPoints().length - 1];
```

```
122             ft.put(m.getKey(), finishTime);
123         }
124         return ft;
125     }
126 
127     /**
128      *  method for sorting a HashMap of Integer and LocalTime.
129      *
130      * @param rt HashMap RiderId and LocalTime.
131      * @return a sorted HashMap.
132      */
133     public static HashMap<Integer, LocalTime> sortRiderByTime(HashMap<Integer, LocalTime> rt) {
134         List<Map.Entry<Integer, LocalTime>> list = new LinkedList<Map.Entry<Integer,
                LocalTime>>(rt.entrySet());
135 
136         // Sort the list
137         list.sort(new Comparator<Map.Entry<Integer, LocalTime>>() {
138             public int compare(Map.Entry<Integer, LocalTime> o1, Map.Entry<Integer, LocalTime> o2) {
139                 return (o1.getValue()).compareTo(o2.getValue());
140             }
141         });
142         HashMap<Integer, LocalTime> sorted = new LinkedHashMap<Integer, LocalTime>();
143         for (Map.Entry<Integer, LocalTime> m : list) {
144             sorted.put(m.getKey(), m.getValue());
145         }
146         return sorted;
147     }
148 
149 }
```