1 CyclingPortals.java

```java
package cycling;

public class CyclingPortals {
   /**
    * store all the portal instances
    *
    * @param args not used
    */
   public static MiniCyclingPortal portal = new MiniCyclingPortal();

}
```

2 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 races in the system
   public static HashMap<Integer, Race> races = new HashMap<Integer, Race>();
   // store all the stages in one race
   public 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++;
   }

```

```
39     // methods:getter& setter
40     public int getRaceID() {
41        return this.raceID;
42     }
43
44     public void setRaceID(int raceID) {
45        this.raceID = raceID;
46     }
47
48     public String getRaceName() {
49        return this.raceName;
50     }
51
52     public void setRaceName(String raceName) {
53        this.raceName = raceName;
54     }
55
56     public String getRaceDesc() {
57        return this.raceDesc;
58     }
59
60     public void setRaceDesc(String raceDesc) {
61        this.raceDesc = raceDesc;
62     }
63
64     public double showRaceLength() {
65        double totallength = 0.0;
66        for (Stage stage : this.raceStages.values()) {
67           totallength += stage.getStageLength();
68        }
69        return totallength;
70     }
71
72     public static boolean checkName(String newname) {
73        HashSet<String> names = new HashSet<String>();
74        for (Integer i : races.keySet()) {
75           names.add(races.get(i).getRaceName());
76        }
77        if (names.contains(newname) == true) {
78           return true;
79        } else {
80           return false;
81        }
82     }
83
84  }
```

3 Stage.java

```
1  package cycling;
2
3  import java.time.LocalDateTime;
4  import java.time.LocalTime;
5  import java.util.Collections;
```

```java
import java.util.Comparator;
import java.util.HashMap;
import java.util.HashSet;
import java.util.LinkedHashMap;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;

public class Stage {
   /**
    * All the stages in the system.
    *
    *
    * @author Kechen Liu
    * @version 1.0
    */
   public static int stageCounter = 1;
   private int stageID;
   private String stageName;
   private String stageDesc;
   private double stageLength;
   private StageType type;
   private LocalDateTime startTime;
   private String stageState;
   private int raceID;
   // store all the stages in the system
   public static HashMap<Integer, Stage> stages = new HashMap<Integer, Stage>();
   // store all the segment in one stage
   public HashMap<Integer, Segment> stageSegments = new HashMap<Integer, Segment>();
   // store all the riders in one stage
   public HashMap<Integer, Rider> stageRiders = new HashMap<Integer, Rider>();
   // store all the riders's totalElapsedTime in one stage
   // <riderid,totalElapsedTime>
   public HashMap<Integer, LocalTime> stageRidersTotalElapsedTime = new HashMap<Integer, LocalTime>();

   // constructor
   Stage(int raceId, String stageName, String stageDesc, double stageLength, LocalDateTime startTime,
       StageType type) {
      this.setRaceID(raceId);
      this.setStageName(stageName);
      this.setStageDesc(stageDesc);
      this.setStageID(stageCounter);
      this.setStageLength(stageLength);
      this.setStageType(type);
      this.setStartTime(startTime);
      this.setStageState(null);
      stageCounter++;
   }

   // getters and setters
   public int getraceID() {
      return this.raceID;
   }

   public void setRaceID(int raceID) {
```

```
60          this.raceID = raceID;
61      }
62
63      public int getStageID() {
64          return this.stageID;
65      }
66
67      public void setStageID(int stageID) {
68          this.stageID = stageID;
69      }
70
71      public String getStageName() {
72          return this.stageName;
73      }
74
75      public void setStageName(String stageName) {
76          this.stageName = stageName;
77      }
78
79      public String getStageDesc() {
80          return this.stageDesc;
81      }
82
83      public void setStageDesc(String stageDesc) {
84          this.stageDesc = stageDesc;
85      }
86
87      public double getStageLength() {
88          return this.stageLength;
89      }
90
91      public void setStageLength(double stageLength) {
92          this.stageLength = stageLength;
93      }
94
95      public StageType getStageType() {
96          return this.type;
97      }
98
99      public void setStageType(StageType type) {
100         this.type = type;
101     }
102
103     public LocalDateTime getStartTime() {
104         return this.startTime;
105     }
106
107     public void setStartTime(LocalDateTime startTime) {
108         this.startTime = startTime;
109     }
110
111     public String getStageState() {
112         return this.stageState;
113     }
114
```

```
115     public void setStageState(String stageState) {
116        this.stageState = stageState;
117     }
118
119     // method: check satge name already exits
120     public static boolean checkName(String newname) {
121        HashSet<String> names = new HashSet<String>();
122        for (Integer i : stages.keySet()) {
123           names.add(stages.get(i).getStageName());
124        }
125        if (names.contains(newname) == true) {
126           return true;
127        } else {
128           return false;
129        }
130     }
131
132     // sort Stage.satges.get(staid).stageRiders by EclapsedTime
133     // server for getRidersRankInStage getRidersPointsInStage
134     public static HashMap<Rider, LocalTime> sortRiderByEclapsedTime(int stageId) {
135        // 1. create current Stage's hashmap
136        HashMap<Rider, LocalTime> stageRiderEclapedTime = new HashMap<>();
137        // 2.put value
138        // obtain all riders belong to the stage
139        Integer staId = stageId;
140        // Stage.stages.get(staId).stageRiders
141        // traverse
142        for (Map.Entry<Integer, Rider> mapElement : Stage.stages.get(staId).stageRiders.entrySet()) {
143           stageRiderEclapedTime.put(mapElement.getValue(), mapElement.getValue().getTotalElapsedTime());
144        }
145        // Create a list from elements of HashMap
146        List<Map.Entry<Rider, LocalTime>> list = new LinkedList<Map.Entry<Rider, LocalTime>>(
147              stageRiderEclapedTime.entrySet());
148
149        // Sort the list
150        Collections.sort(list, new Comparator<Map.Entry<Rider, LocalTime>>() {
151           public int compare(Map.Entry<Rider, LocalTime> o1, Map.Entry<Rider, LocalTime> o2) {
152              return (o1.getValue()).compareTo(o2.getValue());
153           }
154        });
155
156        // put data from sorted list to hashmap
157        HashMap<Rider, LocalTime> sorted = new LinkedHashMap<Rider, LocalTime>();
158        for (Map.Entry<Rider, LocalTime> aa : list) {
159           sorted.put(aa.getKey(), aa.getValue());
160        }
161        return sorted;
162     }
163
164     // sort Stage.satges.get(staid).stageRiders by FinishTime
165     // server for getRankedAdjustedElapsedTimesInStage
166     // getRidersMountainPointsInStage
167     public static HashMap<Rider, LocalTime> sortRiderByFinishTime(int stageId) {
168        HashMap<Rider, LocalTime> stageRiderFinishTime = new HashMap<>();
169        Integer staId = stageId;
```

```
170         // Stage.stages.get(staId).stageRiders
171         // traverse
172         // finishtime = checkpoints[checkpoints.length - 1]
173         for (Map.Entry<Integer, Rider> mapElement : Stage.stages.get(staId).stageRiders.entrySet()) {
174            LocalTime[] checkpoints = mapElement.getValue().getCheckPoints();
175            LocalTime finishTime = checkpoints[checkpoints.length - 1];
176            stageRiderFinishTime.put(mapElement.getValue(), finishTime);
177         }
178         List<Map.Entry<Rider, LocalTime>> list = new LinkedList<Map.Entry<Rider, LocalTime>>(
179               stageRiderFinishTime.entrySet());
180
181         // Sort the list
182         Collections.sort(list, new Comparator<Map.Entry<Rider, LocalTime>>() {
183            public int compare(Map.Entry<Rider, LocalTime> o1, Map.Entry<Rider, LocalTime> o2) {
184               return (o1.getValue()).compareTo(o2.getValue());
185            }
186         });
187         // put data from sorted list to hashmap
188         HashMap<Rider, LocalTime> sorted = new LinkedHashMap<Rider, LocalTime>();
189         for (Map.Entry<Rider, LocalTime> aa : list) {
190            sorted.put(aa.getKey(), aa.getValue());
191         }
192         return sorted;
193      }
194
195      // assign stageRider's points to all riders in a stage
196      // serve for getRidersPointsInStage
197      // call sortRiderByEclapsedTime
198      public static void assignRiderPointsInStage(int stageId) {
199         // get sorted hashmap<Rider,LocalTime> --------> EclapsedTime
200         HashMap<Rider, LocalTime> sortedET = Stage.sortRiderByEclapsedTime(stageId);
201         // determine stage's type
202         Integer staid = stageId;
203         assert (Stage.stages.get(staid).getStageType() == StageType.FLAT
204               || Stage.stages.get(staid).getStageType() == StageType.TT);
205         // calculate the number of riders in a stage
206         int numOfRider = Stage.stages.get(staid).stageRiders.size();
207         assert (numOfRider > 0);
208         if (Stage.stages.get(staid).getStageType() == StageType.FLAT) {
209            // traverse hashmap
210            int n = 1;
211            for (Map.Entry<Rider, LocalTime> mapElement : sortedET.entrySet()) {
212               int riderid = mapElement.getKey().getRiderId();
213               Integer rid = riderid;
214               Rider riderInStage = Stage.stages.get(staid).stageRiders.get(rid);
215               switch (n) {
216                  case 1:
217                     riderInStage.setRiderPoints(50);
218                     break;
219                  case 2:
220                     riderInStage.setRiderPoints(30);
221                     break;
222                  case 3:
223                     riderInStage.setRiderPoints(20);
224                     break;
```

```
            case 4:
               riderInStage.setRiderPoints(18);
               break;
            case 5:
               riderInStage.setRiderPoints(16);
               break;
            case 6:
               riderInStage.setRiderPoints(14);
               break;
            case 7:
               riderInStage.setRiderPoints(12);
               break;
            case 8:
               riderInStage.setRiderPoints(10);
               break;
            case 9:
               riderInStage.setRiderPoints(8);
               break;
            case 10:
               riderInStage.setRiderPoints(7);
               break;
            case 11:
               riderInStage.setRiderPoints(6);
               break;
            case 12:
               riderInStage.setRiderPoints(5);
               break;
            case 13:
               riderInStage.setRiderPoints(4);
               break;
            case 14:
               riderInStage.setRiderPoints(3);
               break;
            case 15:
               riderInStage.setRiderPoints(2);
               break;
         }
         n++;
      }
   }
   if (Stage.stages.get(staid).getStageType() == StageType.TT
         || Stage.stages.get(staid).getStageType() == StageType.HIGH_MOUNTAIN) {
      int n = 1;
      for (Map.Entry<Rider, LocalTime> mapElement : sortedET.entrySet()) {
         int riderid = mapElement.getKey().getRiderId();
         Integer rid = riderid;
         Rider riderInStage = Stage.stages.get(staid).stageRiders.get(rid);
         switch (n) {
            case 1:
               riderInStage.setRiderPoints(20);
               break;
            case 2:
               riderInStage.setRiderPoints(17);
               break;
            case 3:
```

```
                riderInStage.setRiderPoints(15);
                break;
            case 4:
                riderInStage.setRiderPoints(13);
                break;
            case 5:
                riderInStage.setRiderPoints(11);
                break;
            case 6:
                riderInStage.setRiderPoints(10);
                break;
            case 7:
                riderInStage.setRiderPoints(9);
                break;
            case 8:
                riderInStage.setRiderPoints(8);
                break;
            case 9:
                riderInStage.setRiderPoints(7);
                break;
            case 10:
                riderInStage.setRiderPoints(6);
                break;
            case 11:
                riderInStage.setRiderPoints(5);
                break;
            case 12:
                riderInStage.setRiderPoints(4);
                break;
            case 13:
                riderInStage.setRiderPoints(3);
                break;
            case 14:
                riderInStage.setRiderPoints(2);
                break;
            case 15:
                riderInStage.setRiderPoints(1);
                break;
        }
        n++;
    }
}

if (Stage.stages.get(staid).getStageType() == StageType.MEDIUM_MOUNTAIN) {
    // traverse hashtable
    int n = 1;
    for (Map.Entry<Rider, LocalTime> mapElement : sortedET.entrySet()) {
        int riderid = mapElement.getKey().getRiderId();
        Integer rid = riderid;
        Rider riderInStage = Stage.stages.get(staid).stageRiders.get(rid);
        switch (n) {
            case 1:
                riderInStage.setRiderPoints(30);
                break;
            case 2:
```

```
                riderInStage.setRiderPoints(25);
                break;
            case 3:
                riderInStage.setRiderPoints(22);
                break;
            case 4:
                riderInStage.setRiderPoints(19);
                break;
            case 5:
                riderInStage.setRiderPoints(17);
                break;
            case 6:
                riderInStage.setRiderPoints(15);
                break;
            case 7:
                riderInStage.setRiderPoints(13);
                break;
            case 8:
                riderInStage.setRiderPoints(11);
                break;
            case 9:
                riderInStage.setRiderPoints(9);
                break;
            case 10:
                riderInStage.setRiderPoints(7);
                break;
            case 11:
                riderInStage.setRiderPoints(6);
                break;
            case 12:
                riderInStage.setRiderPoints(5);
                break;
            case 13:
                riderInStage.setRiderPoints(4);
                break;
            case 14:
                riderInStage.setRiderPoints(3);
                break;
            case 15:
                riderInStage.setRiderPoints(2);
                break;
          }
          n++;
        }
    }
}

// assign stageRider's mountainpoints to all riders in a stage
// serve for getRidersMountainPointsInStage
// call sortRiderByTotalTime
public static void assignRiderMountainPointsInStage(int stageId) {
    // get sorted hashmap<Rider,LocalTime> --------> EclapsedTime
    HashMap<Rider, LocalTime> sortedFT = Stage.sortRiderByFinishTime(stageId);
    // determine stage's type
    Integer staid = stageId;
```

```
390         assert (Stage.stages.get(staid).getStageType() == StageType.HIGH_MOUNTAIN
391               || Stage.stages.get(staid).getStageType() == StageType.MEDIUM_MOUNTAIN);
392         // calculate the number of riders in a stage
393         int numOfRider = Stage.stages.get(staid).stageRiders.size();
394         assert (numOfRider > 0);
395         if (Stage.stages.get(staid).getStageType() == StageType.MEDIUM_MOUNTAIN) {
396            // traverse hashtable
397            int n = 1;
398            for (Map.Entry<Rider, LocalTime> mapElement : sortedFT.entrySet()) {
399               int riderid = mapElement.getKey().getRiderId();
400               Integer rid = riderid;
401               Rider riderInStage = Stage.stages.get(staid).stageRiders.get(rid);
402               switch (n) {
403                  case 1:
404                     riderInStage.setRiderMountainPoint(30);
405                     break;
406                  case 2:
407                     riderInStage.setRiderMountainPoint(25);
408                     break;
409                  case 3:
410                     riderInStage.setRiderMountainPoint(22);
411                     break;
412                  case 4:
413                     riderInStage.setRiderMountainPoint(19);
414                     break;
415                  case 5:
416                     riderInStage.setRiderMountainPoint(17);
417                     break;
418                  case 6:
419                     riderInStage.setRiderMountainPoint(15);
420                     break;
421                  case 7:
422                     riderInStage.setRiderMountainPoint(13);
423                     break;
424                  case 8:
425                     riderInStage.setRiderMountainPoint(11);
426                     break;
427                  case 9:
428                     riderInStage.setRiderMountainPoint(9);
429                     break;
430                  case 10:
431                     riderInStage.setRiderMountainPoint(7);
432                     break;
433                  case 11:
434                     riderInStage.setRiderMountainPoint(6);
435                     break;
436                  case 12:
437                     riderInStage.setRiderMountainPoint(5);
438                     break;
439                  case 13:
440                     riderInStage.setRiderMountainPoint(4);
441                     break;
442                  case 14:
443                     riderInStage.setRiderMountainPoint(3);
444                     break;
```

```
            case 15:
               riderInStage.setRiderMountainPoint(2);
               break;
         }
         n++;
      }
   }
   if (Stage.stages.get(staid).getStageType() == StageType.HIGH_MOUNTAIN) {
      int n = 1;
      for (Map.Entry<Rider, LocalTime> mapElement : sortedFT.entrySet()) {
         int riderid = mapElement.getKey().getRiderId();
         Integer rid = riderid;
         Rider riderInStage = Stage.stages.get(staid).stageRiders.get(rid);
         switch (n) {
            case 1:
               riderInStage.setRiderMountainPoint(20);
               break;
            case 2:
               riderInStage.setRiderMountainPoint(17);
               break;
            case 3:
               riderInStage.setRiderMountainPoint(15);
               break;
            case 4:
               riderInStage.setRiderMountainPoint(13);
               break;
            case 5:
               riderInStage.setRiderMountainPoint(11);
               break;
            case 6:
               riderInStage.setRiderMountainPoint(10);
               break;
            case 7:
               riderInStage.setRiderMountainPoint(9);
               break;
            case 8:
               riderInStage.setRiderMountainPoint(8);
               break;
            case 9:
               riderInStage.setRiderMountainPoint(7);
               break;
            case 10:
               riderInStage.setRiderMountainPoint(6);
               break;
            case 11:
               riderInStage.setRiderMountainPoint(5);
               break;
            case 12:
               riderInStage.setRiderMountainPoint(4);
               break;
            case 13:
               riderInStage.setRiderMountainPoint(3);
               break;
            case 14:
               riderInStage.setRiderMountainPoint(2);
```

```
                break;
            case 15:
                riderInStage.setRiderMountainPoint(1);
                break;
          }
          n++;
        }
      }
   }

}
```

4 Segment.java

```
package cycling;

import java.util.HashMap;

public class Segment {
   /**
    * All the segments in the system.
    *
    *
    * @author Kechen Liu
    * @version 1.0
    */
   private Double location;
   private Double length;
   private Double averageGrandient;
   private SegmentType type;
   public static int segmentCounter = 1;
   private int segmentId;
   private int stageId;
   // store all the segments in the system
   public static HashMap<Integer, Segment> segments = new HashMap<Integer, Segment>();

   // constructor
   // this constructor for addCateClimlStage
   public Segment(int stageId, Double location, SegmentType type, Double averageGragient, Double length) {
      this.setStageId(stageId);
      this.setLocation(location);
      this.setType(type);
      this.setAverageGrandient(averageGragient);
      this.setLength(length);
      this.setSegmentId(segmentCounter);
      segmentCounter++;
   }

   public Segment(int stageId, Double location) {
      this.setStageId(stageId);
      this.setLocation(location);
      this.setType(SegmentType.SPRINT);
      this.setAverageGrandient(null);
      this.setLength(null);
```

```
41          this.setSegmentId(segmentCounter);
42          segmentCounter++;
43      }
44
45      // get&set
46      public Double getLocation() {
47          return this.location;
48      }
49
50      public void setLocation(Double location) {
51          this.location = location;
52      }
53
54      public Double getLength() {
55          return this.length;
56      }
57
58      public void setLength(Double length) {
59          this.length = length;
60      }
61
62      public Double getAverageGrandient() {
63          return this.averageGrandient;
64      }
65
66      public void setAverageGrandient(Double averageGrandient) {
67          this.averageGrandient = averageGrandient;
68      }
69
70      public SegmentType gettype() {
71          return this.type;
72      }
73
74      public void setType(SegmentType type) {
75          this.type = type;
76      }
77
78      public int getStageId() {
79          return this.stageId;
80      }
81
82      public void setStageId(int stageId) {
83          this.stageId = stageId;
84      }
85
86      public int getSegmentId() {
87          return this.segmentId;
88      }
89
90      public void setSegmentId(int segmentId) {
91          this.segmentId = segmentId;
92      }
93
94  }
```

5 Team.java

```java
package cycling;

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

public class Team {
   /**
    * All the teams in the system.
    *
    *
    * @author Kechen Liu
    * @version 1.0
    */
   static int teamCounter = 1;
   private int teamID;
   private String teamName;
   private String teamDesc;
   // store all the teams in the system
   public static HashMap<Integer, Team> teams = new HashMap<Integer, Team>();
   // store all the riders in one team
   public HashMap<Integer, Rider> teamRiders = new HashMap<Integer, Rider>();

   // constructor
   Team(String teamName) {
      this.setTeamName(teamName);
      this.setTeamDesc(null);
      this.setTeamID(teamCounter);
      teamCounter++;
   }

   Team(String teamName, String teamDesc) {
      this.setTeamName(teamName);
      this.setTeamDesc(teamDesc);
      this.setTeamID(teamCounter);
      teamCounter++;
   }

   // getter and setter methods
   public int getTeamID() {
      return this.teamID;
   }

   public void setTeamID(int teamID) {
      this.teamID = teamID;
   }

   public String getTeamName() {
      return this.teamName;
   }

   public void setTeamName(String teamName) {
      this.teamName = teamName;
```

```
53     }
54
55     public String getTeamDesc() {
56        return this.teamDesc;
57     }
58
59     public void setTeamDesc(String teamDesc) {
60        this.teamDesc = teamDesc;
61     }
62
63     public static boolean checkName(String newname) {
64        HashSet<String> names = new HashSet<String>();
65        for (Integer i : Team.teams.keySet()) {
66           names.add(Team.teams.get(i).getTeamName());
67        }
68        if (names.contains(newname) == true) {
69           return true;
70        } else {
71           return false;
72        }
73     }
74
75  }
```

6 Rider.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.HashMap;
11 import java.util.List;
12 import java.util.Map;
13
14 public class Rider {
15    /**
16     * All the rider in the system.
17     *
18     *
19     * @author Kechen Liu
20     * @version 1.0
21     */
22    private String name;
23    private int yearOfBirth;
24    public static int riderIDCounter = 1;
25    private int riderID;
26    private int teamId;
27    // these fields specifically used for riders in stage
28    private ArrayList<Integer> stageIds = new ArrayList<Integer>();
```

```
29      private LocalTime[] checkpoints;
30      private LocalTime totalElapsedTime;
31      private boolean RegisteredInStage = false;
32      private LocalTime adjustedElapsedTime;
33      private int riderPoints;
34      private int riderMountainPoints;
35      // store all the riders in the system
36      public static HashMap<Integer, Rider> riders = new HashMap<Integer, Rider>();
37
38      // constructor
39      public Rider(int teamId, String name, int yearOfBirth) {
40         this.teamId = teamId;
41         this.setRiderName(name);
42         this.setYearOfBirth(yearOfBirth);
43         this.riderID = riderIDCounter;
44         riderIDCounter++;
45      }
46
47      // method: getter& setter
48      public String getRiderName() {
49         return this.name;
50      }
51
52      public void setRiderName(String name) {
53         this.name = name;
54      }
55
56      public int getYearOfBirth() {
57         return this.yearOfBirth;
58      }
59
60      public void setYearOfBirth(int yearOfBirth) {
61         this.yearOfBirth = yearOfBirth;
62      }
63
64      public int getRiderId() {
65         return this.riderID;
66      }
67
68      public void setRiderId(String name) {
69         this.name = name;
70      }
71
72      public int getTeamId() {
73         return this.teamId;
74      }
75
76      public void setTeamId(int teamId) {
77         this.teamId = teamId;
78      }
79
80      public ArrayList<Integer> getStageIds() {
81         return this.stageIds;
82      }
83
```

```
84      public void setStageId(ArrayList<Integer> stageIds) {
85          this.stageIds = stageIds;
86      }
87
88      public LocalTime[] getCheckPoints() {
89          return this.checkpoints;
90      }
91
92      public void setCheckPoints(LocalTime[] checkpoints) {
93          this.checkpoints = checkpoints;
94      }
95
96      public LocalTime getAdjustedElapsedTime() {
97          return this.adjustedElapsedTime;
98      }
99
100     public void setAdjustedElapsedTime(LocalTime adjustedElapsedTime) {
101         this.adjustedElapsedTime = adjustedElapsedTime;
102     }
103
104     public void setTotalElapsedTime(LocalTime totalElapsedTime) {
105         this.totalElapsedTime = totalElapsedTime;
106     }
107
108     public LocalTime getTotalElapsedTime() {
109         return this.totalElapsedTime;
110     }
111
112     public void setRegisteredInStage(boolean RegisteredInStage) {
113         this.RegisteredInStage = RegisteredInStage;
114     }
115
116     // RegisteredTotalET
117     public boolean getRegisteredInStage() {
118         return this.RegisteredInStage;
119     }
120
121     public int getRiderPoints() {
122         return this.riderPoints;
123     }
124
125     public void setRiderPoints(int riderPoints) {
126         this.riderPoints = riderPoints;
127     }
128
129     public int getRiderMountainPoints() {
130         return this.riderMountainPoints;
131     }
132
133     public void setRiderMountainPoint(int riderMountainPoints) {
134         this.riderMountainPoints = riderMountainPoints;
135     }
136
137     // method: guarantee checkpoints's length correct
138     // serve for portal: registerRiderResultsInStage's InvalidCheckpointsException
```

```
139     public static boolean guranteeCheckpoints(int stageId, LocalTime... checkpoints) {
140        // get the stage
141        Integer staId = stageId;
142        Stage stage = Stage.stages.get(staId);
143        // get stage's segment number
144        int n = stage.stageSegments.size();
145        if (checkpoints.length != (n + 2)) {
146           return true;
147        } else {
148           return false;
149        }
150     }
151
152     // method: calculate rider's totalElapseTime
153     public static LocalTime calculateTotalElapsedTime(int riderID, int stageID) {
154        Integer rid = riderID;
155        Integer staid = stageID;
156        // 1. obtain rider's checkpoints
157        Rider riderInStage = Stage.stages.get(staid).stageRiders.get(rid);
158        LocalTime[] checkpoints = riderInStage.getCheckPoints();
159        // obtain Duration(calculate the difference)
160        Duration totalETd = Duration.between(checkpoints[0], checkpoints[checkpoints.length - 1]);
161        // 2.convert durantion to localtim
162        long longd = totalETd.toMillis();
163        LocalTime totalElapsedTime = LocalDateTime.ofInstant(Instant.ofEpochMilli(longd),
               ZoneId.systemDefault())
164              .toLocalTime();
165        return totalElapsedTime;
166     }
167
168     // method: assign TotalElapsedTime to rider when registered in a specific stage
169     public static void confiTotalElapsedTime(Integer staid, Integer rid, LocalTime totalElapsedTime) {
170        Rider riderInStage = Stage.stages.get(staid).stageRiders.get(rid);
171        // assign value to rider's totalElapsedTime
172        riderInStage.setTotalElapsedTime(totalElapsedTime);
173        // add rider's totalElapsedTime to stageRiderTotalElapsedTime
174        Stage.stages.get(staid).stageRidersTotalElapsedTime.put(rid, totalElapsedTime);
175     }
176
177     // method: calculate rider's AdjustedElapsedTime
178     public static LocalTime calculateAdjustedElapsedTime(int riderID, int stageID) {
179        // 1. obtain all riders' finish time, find the smallest
180        // 1.1 obtain all the riders in the stage
181        Integer rid = riderID;
182        Integer staid = stageID;
183        // There is no adjustments on elapsed time on time-trials.
184        if (Stage.stages.get(staid).getStageType() == StageType.TT) {
185           return null;
186        }
187        HashMap<Integer, Rider> allRidersInStage = Stage.stages.get(staid).stageRiders;
188        // 1.2 find the smallest finishtime
189        // 1.2.1 create a list to store all the finishtime
190        List<LocalTime> ftList = new ArrayList<>();
191        // 1.2.2 traverse the hashmap
192
```

```
193         for (Map.Entry<Integer, Rider> mapElement : allRidersInStage.entrySet()) {
194             LocalTime[] checkpoints = mapElement.getValue().getCheckPoints();
195             LocalTime finishTime = checkpoints[checkpoints.length - 1];
196             // System.out.print(finishTime + " ");
197             ftList.add(LocalTime.parse(finishTime.toString()));
198         }
199         // 1.2.3 sortftList
200         Collections.sort(ftList);
201         // 1.2.4 find the smallest finishTime
202         LocalTime smallFt = ftList.get(0);
203         // 2.calculate rider's aet
204         // 2.1 get the rider
205         Rider riderInStage = Stage.stages.get(staid).stageRiders.get(rid);
206         // 2.2 calculate the difference
207         LocalTime startTime = riderInStage.getCheckPoints()[0];
208         // System.out.println("debug");
209         // System.out.println("startTime" + startTime); ok
210         Duration aetd = Duration.between(startTime, smallFt);
211         // 2.convert durantion to localtim
212         long longd = aetd.toMillis();
213         LocalTime adjustedElapsedTime = LocalDateTime.ofInstant(Instant.ofEpochMilli(longd),
                ZoneId.systemDefault())
214               .toLocalTime();
215         return adjustedElapsedTime;
216     }
217
218     // // method: assign AdjustedElapsedTime to rider when registered in a specific
219     // stage
220     public static void confiAdjustedElapsedTime(Integer staid, Integer rid, LocalTime adjustedElapsedTime) {
221         Rider riderInStage = Stage.stages.get(staid).stageRiders.get(rid);
222         // assign value to rider's adjustedElapsedTime
223         riderInStage.setAdjustedElapsedTime(adjustedElapsedTime);
224     }
225
226     // these method call the four methods up it
227     // assign attributes to rider in the specific stage
228     // serve for registerRiderResultsInStage
229     public static void generateRecord(int staid, int rid) {
230         Integer riderid = rid;
231         Integer sid = staid;
232         LocalTime totalElapsedTime = Rider.calculateTotalElapsedTime(riderid, sid);
233         Rider.confiTotalElapsedTime(sid, riderid, totalElapsedTime);
234         LocalTime adjustedElapsedTime = Rider.calculateAdjustedElapsedTime(riderid, sid);
235         Rider.confiAdjustedElapsedTime(sid, riderid, adjustedElapsedTime);
236     }
237
238 }
```