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;
   }
```

```java

    public String getRaceDesc() {
       return this.raceDesc;
    }

    public void setRaceDesc(String raceDesc) {
       this.raceDesc = raceDesc;
    }

    public HashMap<Integer, Stage> getRaceStages() {
       return this.raceStages;
    }

    public void setRaceStages(HashMap<Integer, Stage> rs) {this.raceStages = rs;}

    public double showRaceLength() {
       double totallength = 0.0;
       for (Stage stage : this.raceStages.values()) {
          totallength += stage.getStageLength();
       }
       return totallength;
    }

    public boolean stageNameCheck(String name) {
       HashSet<String> names = new HashSet<String>();
       for (Integer i :raceStages.keySet()) {
          names.add(raceStages.get(i).getStageName());
       }
       return names.contains(name) == true;
    }



}
```

2 Stage.java

```java
package cycling;

import java.time.LocalDateTime;
import java.time.LocalTime;
import java.util.ArrayList;
import java.util.HashMap;
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;
    private HashMap<Integer, Segment> stageSegments = new HashMap<Integer, Segment>();
    // store all the riders in one stage
    private HashMap<Integer, Rider> stageRiders = new HashMap<Integer, Rider>();

    // 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) {
        this.raceID = raceID;
    }

    public int getStageID() {
        return this.stageID;
    }

    public void setStageID(int stageID) {
        this.stageID = stageID;
    }

    public String getStageName() {
        return this.stageName;
    }

    public void setStageName(String stageName) {
        this.stageName = stageName;
    }

    public String getStageDesc() {
        return this.stageDesc;
    }

    public void setStageDesc(String stageDesc) {
        this.stageDesc = stageDesc;
    }

    public double getStageLength() {
        return this.stageLength;
    }

    public void setStageLength(double stageLength) {
        this.stageLength = stageLength;
    }

    public StageType getStageType() {
        return this.type;
    }

    public void setStageType(StageType type) {
        this.type = type;
    }

    public LocalDateTime getStartTime() {
        return this.startTime;
    }

    public void setStartTime(LocalDateTime startTime) {
        this.startTime = startTime;
    }

    public String getStageState() {
        return this.stageState;
    }

    public HashMap<Integer, Segment> getStageSegments() {
        return this.stageSegments;
    }

    public void setStageState(String stageState) {
        this.stageState = stageState;
    }

    public HashMap<Integer, Rider> getStageRiders() {
        return this.stageRiders;
    }

    /**
     * Get the rider's result in the corresponding stage
     * @param st the stage user wants to find result.
     * @return A HashMap of riderId and corresponding result
     */
    public static HashMap<Integer, Result> getRiderResultInStage(Stage st) {
        int stageId = st.stageID;
        // riderId <stageId, result>
        HashMap<Integer, HashMap<Integer, Result>> rsr = new HashMap<>();
        HashMap<Integer, Rider> riders = st.getStageRiders();

        for (Map.Entry<Integer, Rider> mapElement : riders.entrySet()) {
            rsr.put(mapElement.getKey(), mapElement.getValue().getResults());
        }

        // riderId, result
        HashMap<Integer, Result> rr = new HashMap<>();
        for (Map.Entry<Integer, HashMap<Integer, Result>> m : rsr.entrySet()) {
            if (m.getValue().containsKey(stageId)) {
                rr.put(m.getKey(), m.getValue().get(stageId));
            }
        }
        return rr;
    }

    /**
     *  Get Ranked riderIds
     * @param rt HashMap of riderId and LocalTime(can be eclipsed time or adjusted time
     * @return a ranked int array of riders' ids.
     */
    public static int[] getRankedIds(HashMap<Integer, LocalTime> rt) {
        ArrayList<Integer> listRids = new ArrayList<>();
        // iterate over and get keys and values
        for (Map.Entry<Integer, LocalTime> m : rt.entrySet()) {
            listRids.add(m.getKey());
        }
        int[] ids = new int[listRids.size()];
        for (int i = 0; i < listRids.size(); i++) {
            int e = listRids.get(i);
            ids[i] = e;
        }
        return ids;
    }

    /**
     * Get ranked LocalTimes
     * @param rt HashMap of riderId and LocalTime(can be eclipsed time or adjusted time
     * @return a ranked LocalTime array of riders' time
     */
    public static LocalTime[] getRankedTimes(HashMap<Integer, LocalTime> rt) {
        ArrayList<LocalTime> listTimes = new ArrayList<>();
        // iterate over and get keys and values
        for (Map.Entry<Integer, LocalTime> m : rt.entrySet()) {
            listTimes.add(m.getValue());
        }
        LocalTime[] times = new LocalTime[listTimes.size()];
        for (int i = 0; i < listTimes.size(); i++) {
            LocalTime e = listTimes.get(i);
            times[i] = e;
        }
        return times;
    }

}
```

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;
   }
```

```java

    public void setSegmentId(int segmentId) {
        this.segmentId = segmentId;
    }


    /**
     * assign rider's point in stage based on their rank in stage and stage type
     * @param riderRank rider's rank in stage, call portal's getRiderRank method to get
     * @param type stage's type
     * @return int array of rider's points, corresponding to rider's id
     */
    public static int[] assignRiderPointsInStage(int[] riderRank, StageType type) {
        int[] points = new int[riderRank.length];
        if (type == StageType.FLAT) {
            for (int i = 0; i < riderRank.length; i++) {
                int n = riderRank[i];
                switch (n) {
                    case 1 -> points[i] = 50;
                    case 2 -> points[i] = 30;
                    case 3 -> points[i] = 20;
                    case 4 -> points[i] = 18;
                    case 5 -> points[i] = 16;
                    case 6 -> points[i] = 14;
                    case 7 -> points[i] = 12;
                    case 8 -> points[i] = 10;
                    case 9 -> points[i] = 8;
                    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;
                }
            }
        }
        if (type == StageType.TT
                || type == StageType.HIGH_MOUNTAIN) {
            for (int i = 0; i < riderRank.length; i++) {
                int n = riderRank[i];
                switch (n) {
                    case 1 -> points[i] = 20;
                    case 2 -> points[i] = 17;
                    case 3 -> points[i] = 15;
                    case 4 -> points[i] = 13;
                    case 5 -> points[i] = 11;
                    case 6 -> points[i] = 10;
                    case 7 -> points[i] = 9;
                    case 8 -> points[i] = 8;
                    case 9 -> points[i] = 7;
                    case 10 -> points[i] = 6;
                    case 11 -> points[i] = 5;
                    case 12 -> points[i] = 4;
                    case 13 -> points[i] = 3;
                    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

```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

```java
package cycling;

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

    private Double averageGradient;

    private Double length;

    public Double getLength() {
        return this.length;
    }

    public void setLength(Double length) {
        this.length = length;
    }

    public Double getAverageGradient() {
        return this.averageGradient;
    }

    public void setAverageGradient(Double averageGradient) {
        this.averageGradient = averageGradient;
    }

    public void setAverageGrandient(Double averageGrandient) {
        this.averageGradient = averageGrandient;
    }

    public Mountains(int stageId, Double location, SegmentType type, Double averageGragient, Double length)
         {
        this.setStageId(stageId);
        this.setLocation(location);
        this.setType(type);
        this.setAverageGradient(averageGragient);
        this.setLength(length);
        this.setSegmentId(segmentCounter);
        segmentCounter++;
    }

    public Mountains(int stageId, Double location) {
    }
}
```

6 Team.java

```java
package cycling;

import java.util.HashMap;

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 riders in one team
    private HashMap<Integer, Rider> teamRiders = new HashMap<Integer, Rider>();

    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;
    }

    public String getTeamDesc() {
        return this.teamDesc;
    }

    public void setTeamDesc(String teamDesc) {
        this.teamDesc = teamDesc;
    }

    public HashMap<Integer, Rider> getTeamRiders() {
        return this.teamRiders;
    }

}
```

7 Rider.java

```java
package cycling;

import java.util.ArrayList;
import java.util.HashMap;

public class Rider {
    /**
```

```java
     * 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() {
        return this.teamId;
    }

    public void setTeamId(int teamId) {
        this.teamId = teamId;
    }

    public ArrayList<Integer> getStageIds() {
        return this.stageIds;
    }

    public void setStageId(ArrayList<Integer> stageIds) {
        this.stageIds = stageIds;
    }

    public int getRiderPoints() {
        return this.riderPoints;
    }

    public void setRiderPoints(int riderPoints) {
        this.riderPoints = riderPoints;
    }

    public int getRiderMountainPoints() {
        return this.riderMountainPoints;
    }

    public void setRiderMountainPoint(int riderMountainPoints) {
        this.riderMountainPoints = riderMountainPoints;
    }

    public HashMap<Integer, Result> getResults() {
        return this.results;
    }

    public void setResult(HashMap<Integer, Result> results) {
        this.results = results;
    }

}
```

8 Result.java

```java
package cycling;

import java.time.Duration;
import java.time.Instant;
import java.time.LocalDateTime;
import java.time.LocalTime;
import java.time.ZoneId;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.HashMap;
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]);
```

```java
        return et;
    }

    public LocalTime calculateAdjustedElapsedTime(LocalTime smallest) {
        this.adjustedElapsedTime = calcTime(checkpoints[0], smallest);
        return adjustedElapsedTime;
    }

    /* method to calculate the difference between start and end. */
    public static LocalTime calcTime(LocalTime start, LocalTime end) {
        Duration time = Duration.between(start, end);
        long longd = time.toMillis();
        LocalTime newTime = LocalDateTime.ofInstant(Instant.ofEpochMilli(longd), ZoneId.systemDefault())
                .toLocalTime();
        return newTime;
    }

    /**
     * method to get rider's eclipsed finish time
     *
     * @param rr HashMap of RiderId and the corresponding result.
     * @return HashMap of RiderId and corresponding eclipsed finish time.
     */
    public static HashMap<Integer, LocalTime> getET(HashMap<Integer, Result> rr) {
        HashMap<Integer, LocalTime> et = new HashMap<>();
        for (Map.Entry<Integer, Result> m : rr.entrySet()) {
            et.put(m.getKey(), m.getValue().getTotalElapsedTime());
        }
        return et;
    }

    /**
     * method to get rider's adjusted finish time
     *
     * @param rr HashMap of RiderId and the corresponding result.
     * @return HashMap of RiderId and corresponding adjusted finish time.
     */
    public static HashMap<Integer, LocalTime> getAT(HashMap<Integer, Result> rr) {
        HashMap<Integer, LocalTime> et = new HashMap<>();
        for (Map.Entry<Integer, Result> m : rr.entrySet()) {
            et.put(m.getKey(), m.getValue().getAdjustedElapsedTime());
        }
        return et;
    }

    /**
     * method to get rider's adjusted finish time
     *
     * @param rr HashMap of RiderId and the corresponding result.
     * @return HashMap of RiderId and corresponding adjusted finish time.
     */
    public static HashMap<Integer, LocalTime> getFT(HashMap<Integer, Result> rr) {
        HashMap<Integer, LocalTime> ft = new HashMap<>();
        for (Map.Entry<Integer, Result> m : rr.entrySet()) {
            LocalTime finishTime = m.getValue().getCheckPoints()[m.getValue().getCheckPoints().length - 1];
            ft.put(m.getKey(), finishTime);
        }
        return ft;
    }

    /**
     *  method for sorting a HashMap of Integer and LocalTime.
     *
     * @param rt HashMap RiderId and LocalTime.
     * @return a sorted HashMap.
     */
    public static HashMap<Integer, LocalTime> sortRiderByTime(HashMap<Integer, LocalTime> rt) {
        List<Map.Entry<Integer, LocalTime>> list = new LinkedList<Map.Entry<Integer,
             LocalTime>>(rt.entrySet());

        // Sort the list
        list.sort(new Comparator<Map.Entry<Integer, LocalTime>>() {
            public int compare(Map.Entry<Integer, LocalTime> o1, Map.Entry<Integer, LocalTime> o2) {
                return (o1.getValue()).compareTo(o2.getValue());
            }
        });
        HashMap<Integer, LocalTime> sorted = new LinkedHashMap<Integer, LocalTime>();
        for (Map.Entry<Integer, LocalTime> m : list) {
            sorted.put(m.getKey(), m.getValue());
        }
        return sorted;
    }

}
```