1 CyclingPortal.java

```
1  package cycling;
2
3  import java.util.Arrays;
4
5  import java.io.IOException;
6  import java.time.LocalDateTime;
7  import java.time.LocalTime;
8  import java.util.ArrayList;
9  import java.io.ObjectOutputStream;
10 import java.io.FileOutputStream;
11 import java.io.ObjectInputStream;
12 import java.io.FileInputStream;
13
14
15
16 /**
17  * CyclingPortal implements CyclingPortalInterface; contains
        methods for
18  * handling the following classes: Race, Stage, Segment,
        RiderManager (and in
19  * turn Rider and Team), and Result.
20  * These classes are used manage races and their subdivisions,
        teams and their
21  * riders, and to calculate and assign points.
22  * Also contains methods for saving and loading
        MiniCyclingPortalInterface to
23  * and from a file.
24  *
25  * @author Ethan Ray & Thomas Newbold
26  * @version 1.0
27  *
28  */
29 public class CyclingPortal implements CyclingPortalInterface {
30     public RiderManager riderManager = new RiderManager();
31
32     @Override
33     public int[] getRaceIds() {
34         return Race.getAllRaceIds();
35     }
36
37     @Override
38     public int createRace(String name, String description)
           throws IllegalNameException, InvalidNameException {
39         Race r = new Race(name, description);
40         return r.getRaceId();
41     }
42
```

```
43      @Override
44      public String viewRaceDetails(int raceId) throws
            IDNotRecognisedException {
45          double sum = 0.0;
46          for(int id : Race.getStages(raceId)) {
47              sum += Stage.getStageLength(id);
48          }
49          return Race.toString(raceId)+Double.toString(sum)+";";
50      }
51
52      @Override
53      public void removeRaceById(int raceId) throws
            IDNotRecognisedException {
54          Race.removeRace(raceId);
55      }
56
57      @Override
58      public int getNumberOfStages(int raceId) throws
            IDNotRecognisedException {
59          int[] stageIds = Race.getStages(raceId);
60          return stageIds.length;
61      }
62
63      @Override
64      public int addStageToRace(int raceId, String stageName,
            String description, double length, LocalDateTime
            startTime,
65              StageType type)
66              throws IDNotRecognisedException,
                    IllegalNameException, InvalidNameException,
                    InvalidLengthException {
67          return Race.addStageToRace(raceId, stageName,
                description, length, startTime, type);
68      }
69
70      @Override
71      public int[] getRaceStages(int raceId) throws
            IDNotRecognisedException {
72          return Race.getStages(raceId);
73      }
74
75      @Override
76      public double getStageLength(int stageId) throws
            IDNotRecognisedException {
77          return Stage.getStageLength(stageId);
78      }
79
80      @Override
81      public void removeStageById(int stageId) throws
            IDNotRecognisedException {
```

```
82          Race.removeStage(stageId);
83      }
84
85      @Override
86      public int addCategorizedClimbToStage(int stageId, Double
            location, SegmentType type, Double averageGradient,
87              Double length) throws IDNotRecognisedException,
                    InvalidLocationException,
                    InvalidStageStateException,
88              InvalidStageTypeException {
89          return Stage.addSegmentToStage(stageId, location, type,
                 averageGradient, length);
90      }
91
92      @Override
93      public int addIntermediateSprintToStage(int stageId, double
             location) throws IDNotRecognisedException,
94              InvalidLocationException,
                    InvalidStageStateException,
                    InvalidStageTypeException {
95          // TODO Check inputs?
96          return Stage.addSegmentToStage(stageId, location,
                SegmentType.SPRINT, 0.0, location);
97      }
98
99      @Override
100     public void removeSegment(int segmentId) throws
            IDNotRecognisedException, InvalidStageStateException {
101         Stage.removeSegment(segmentId);
102     }
103
104     @Override
105     public void concludeStagePreparation(int stageId) throws
            IDNotRecognisedException, InvalidStageStateException {
106         Stage.updateStageState(stageId);
107     }
108
109     @Override
110     public int[] getStageSegments(int stageId) throws
            IDNotRecognisedException {
111         return Stage.getSegments(stageId);
112     }
113
114     @Override
115     public int createTeam(String name, String description)
            throws IllegalNameException, InvalidNameException {
116         return riderManager.createTeam(name, description);
117     }
118
119     @Override
```

```
120     public void removeTeam(int teamId) throws
            IDNotRecognisedException {
121         riderManager.removeTeam(teamId);
122
123     }
124
125     @Override
126     public int[] getTeams() {
127         return riderManager.getTeams();
128     }
129
130     @Override
131     public int[] getTeamRiders(int teamId) throws
            IDNotRecognisedException {
132         return riderManager.getTeamRiders(teamId);
133     }
134
135     @Override
136     public int createRider(int teamID, String name, int
            yearOfBirth) throws IDNotRecognisedException,
            IllegalArgumentException {
137         return riderManager.createRider(teamID, name,
                yearOfBirth);
138
139     }
140
141     @Override
142     public void removeRider(int riderId) throws
            IDNotRecognisedException {
143         riderManager.removeRider(riderId);
144
145     }
146
147     @Override
148     public void registerRiderResultsInStage(int stageId, int
            riderId, LocalTime... checkpoints)
149             throws IDNotRecognisedException,
                    DuplicatedResultException,
                    InvalidCheckpointsException,
150                 InvalidStageStateException {
151         if(Stage.getStageState(stageId).equals(StageState.
                BUILDING)) {
152             throw new InvalidStageStateException("stage is not
                    waiting for results");
153         } else if(Stage.getSegments(stageId).length+2 !=
                checkpoints.length) {
154             throw new InvalidCheckpointsException("checkpoint
                    count mismatch");
155         }
156         try {
```

```
157             Result.getResult(stageId, riderId);
158             throw new DuplicatedResultException();
159         } catch(IDNotRecognisedException ex) {
160             Stage.getStage(stageId);
161             riderManager.getRider(riderId);
162             // above should throw exceptions if IDs are not in
                    system
163             new Result(stageId, riderId, checkpoints);
164         }
165     }
166 
167     @Override
168     public LocalTime[] getRiderResultsInStage(int stageId, int
            riderId) throws IDNotRecognisedException {
169         Stage.getStage(stageId);
170         riderManager.getRider(riderId);
171         // above should throw exceptions if IDs are not in
                system
172         Result result = Result.getResult(stageId, riderId);
173         LocalTime[] checkpointTimes = result.getCheckpoints();
174         LocalTime[] out = new LocalTime[checkpointTimes.length
                +1];
175         for(int i=0; i<checkpointTimes.length; i++) {
176             out[i] = checkpointTimes[i];
177         }
178         out[-1] = result.getTotalElasped();
179         return out;
180     }
181 
182     @Override
183     public LocalTime getRiderAdjustedElapsedTimeInStage(int
            stageId, int riderId) throws IDNotRecognisedException {
184         Stage.getStage(stageId);
185         riderManager.getRider(riderId);
186         // above should throw exceptions if IDs are not in
                system
187         LocalTime[] adjustedTimes = Result.getResult(stageId,
                riderId).adjustedCheckpoints();
188         LocalTime elapsedTime = adjustedTimes[0];
189         for(int i=1; i<adjustedTimes.length; i++) {
190             LocalTime t = adjustedTimes[i];
191             elapsedTime.plusHours(t.getHour()).plusMinutes(t.
                    getMinute()).plusSeconds(t.getSecond()).
                    plusNanos(t.getNano());
192         }
193         return elapsedTime;
194     }
195 
196     @Override
197     public void deleteRiderResultsInStage(int stageId, int
```

```
            riderId) throws IDNotRecognisedException {
198         Stage.getStage(stageId);
199         riderManager.getRider(riderId);
200         // above should throw exceptions if IDs are not in
                system
201         Result.removeResult(stageId, riderId);
202     }
203
204     @Override
205     public int[] getRidersRankInStage(int stageId) throws
            IDNotRecognisedException {
206         Result[] results = Result.getResultsInStage(stageId);
207         int[] riderRanks = new int[results.length];
208         Arrays.fill(riderRanks, -1);
209         for(Result r : results) {
210             for(int i=0; i<riderRanks.length; i++) {
211                 if(riderRanks[i] == -1) {
212                     riderRanks[i] = r.getRiderId();
213                 } else {
214                     LocalTime[] rTimes = r.getCheckpoints();
215                     LocalTime[] compTimes = Result.getResult(
                            stageId, riderRanks[i]).getCheckpoints()
                            ;
216                     if(rTimes[rTimes.length-1].isBefore(
                            compTimes[compTimes.length-1])) {
217                         int temp;
218                         int prev = r.getRiderId();
219                         for(int j=i; j<riderRanks.length; j++)
                                {
220                             temp = riderRanks[j];
221                             riderRanks[j] = prev;
222                             prev = temp;
223                             if(prev == -1) {
224                                 break;
225                             }
226                         }
227                     }
228                     break;
229                 }
230             }
231         }
232         return riderRanks;
233     }
234
235     @Override
236     public LocalTime[] getRankedAdjustedElapsedTimesInStage(int
            stageId) throws IDNotRecognisedException {
237         // TODO Auto-generated method stub
238         // TODO Thomas do this after mountain points
239         return null;
```

```
240     }
241
242     @Override
243     public int[] getRidersPointsInStage(int stageId) throws
            IDNotRecognisedException {
244         StageType type = Stage.getStageType(stageId);
245         int[] points = new int[Result.getResultsInStage(stageId
                ).length];
246         int[] distribution = new int[15];
247         // distributions from https://en.wikipedia.org/wiki/
                Points_classification_in_the_Tour_de_France
248         switch(type) {
249             case FLAT:
250                 distribution = new int
                        []{50,30,20,18,16,14,12,10,8,7,6,5,4,3,2};
251                 break;
252             case MEDIUM_MOUNTAIN:
253                 distribution = new int
                        []{30,25,22,19,17,15,13,11,9,7,6,5,4,3,2};
254                 break;
255             case HIGH_MOUNTAIN:
256                 distribution = new int
                        []{20,17,15,13,11,10,9,8,7,6,5,4,3,2,1};
257                 break;
258             case TT:
259                 distribution = new int
                        []{20,17,15,13,11,10,9,8,7,6,5,4,3,2,1};
260                 break;
261         }
262         for(int i=0; i<Math.min(points.length, distribution.
                length); i++) {
263             points[i] = distribution[i];
264         }
265         return points;
266     }
267
268     @Override
269     public int[] getRidersMountainPointsInStage(int stageId)
            throws IDNotRecognisedException {
270         Result[] results = Result.getResultsInStage(stageId);
271         // All results refering to the stage with id *stageId*
272         int[] riders = getRidersRankInStage(stageId);
273         // An int array of rider ids, from first to last
274         int[] segments = Stage.getSegments(stageId);
275         // An int array of the segment ids in the stage
276         int[] points = new int[riders.length];
277         // The int in position i is the number of points to be
                awarded to the rider with id riders[i]
278         for(int s=0; s<segments.length; s++) {
279             SegmentType type = Segment.getSegmentType(segments[
```

```
                s]);
280             int[] distribution = new int[1];
281             // The points to be awarded in order for the
                    segment
282             switch(type) {
283                 case C4:
284                     distribution = new int[]{1};
285                     break;
286                 case C3:
287                     distribution = new int[]{2,1};
288                     break;
289                 case C2:
290                     distribution = new int[]{5,3,2,1};
291                     break;
292                 case C1:
293                     distribution = new int[]{10,8,6,4,2,1};
294                     break;
295                 case HC:
296                     distribution = new int
                            []{20,15,12,10,8,6,4,2};
297                     break;
298                 case SPRINT:
299             }
300             // get ranks for segment
301             int[] riderRanks = new int[results.length];
302             Arrays.fill(riderRanks, -1);
303             for(Result r : results) {
304                 for(int i=0; i<riderRanks.length; i++) {
305                     if(riderRanks[i] == -1) {
306                         riderRanks[i] = r.getRiderId();
307                     } else {
308                         Result compare = Result.getResult(
                                stageId, riderRanks[i]);
309                         if(r.getCheckpoints()[s].isBefore(
                                compare.getCheckpoints()[s])) {
310                             int temp;
311                             int prev = r.getRiderId();
312                             for(int j=i; j<riderRanks.length; j
                                    ++) {
313                                 temp = riderRanks[j];
314                                 riderRanks[j] = prev;
315                                 prev = temp;
316                                 if(prev == -1) {
317                                     break;
318                                 }
319                             }
320                         }
321                         break;
322                     }
323                 }
```

```
324             }
325             //return riderRanks;
326             ArrayList<Integer> ridersArray = new ArrayList<
                    Integer>();
327             for(int r : riders) { ridersArray.add(r); }
328             for(int i=0; i<Math.min(points.length, distribution
                    .length); i++) {
329                 int overallPos = ridersArray.indexOf(riderRanks
                        [i]);
330                 if(overallPos<points.length) {
331                     points[overallPos] += distribution[i];
332                 }
333             }
334         }
335         return points;
336     }
337 
338     @Override
339     public void eraseCyclingPortal() {
340 
341         Team.teamNames.clear();
342         Team.teamTopId = 0;
343         Rider.ridersTopId = 0;
344 
345         RiderManager.allRiders.clear();
346         RiderManager.allTeams.clear();
347 
348 
349         Race.allRaces.clear();
350         Race.removedIds.clear();
351         Race.loadId();
352 
353         Segment.allSegments.clear();
354         Segment.removedIds.clear();
355         Segment.loadId();
356 
357         Stage.allStages.clear();
358         Stage.removedIds.clear();
359         Stage.loadId();
360 
361         Result.allResults.clear();
362 
363 
364     }
365 
366     @Override
367     public void saveCyclingPortal(String filename) throws
            IOException {
368         try {
369             FileOutputStream fos = new FileOutputStream(
```

```
                filename);
370             ObjectOutputStream oos = new ObjectOutputStream(fos
                    );
371             ArrayList<ArrayList> allObj = new ArrayList<>();
372             allObj.add(RiderManager.allTeams);
373             allObj.add(RiderManager.allRiders);
374             allObj.add(Stage.allStages);
375             allObj.add(Stage.removedIds);
376             allObj.add(Race.allRaces);
377             allObj.add(Race.removedIds);
378             allObj.add(Result.allResults);
379             allObj.add(Segment.allSegments);
380             allObj.add(Segment.removedIds);
381
382             oos.writeObject(allObj);
383
384             oos.flush();
385             oos.close();
386
387         } catch (IOException ex) {
388             ex.printStackTrace();
389         }
390
391     }
392
393     @Override
394     public void loadCyclingPortal(String filename) throws
            IOException, ClassNotFoundException {
395         try {
396
397             FileInputStream fis = new FileInputStream(filename)
                    ;
398             ObjectInputStream ois = new ObjectInputStream(fis);
399             ArrayList<Object> allObjects = new ArrayList<>();
400             ArrayList<Team> allTeams = new ArrayList<>();
401             ArrayList<Rider> allRiders = new ArrayList<>();
402             ArrayList<Result> allResults = new ArrayList<Result
                    >();
403             ArrayList<Race> allRaces = new ArrayList<Race>();
404             ArrayList<Stage> allStages = new ArrayList<Stage>()
                    ;
405             ArrayList<Segment> allSegments = new ArrayList<
                    Segment>();
406             ArrayList<Integer> removedIds = new ArrayList<>();
407
408             Class<?> classFlag = null;
409
410             allObjects = (ArrayList) ois.readObject();
411             for (Object tempObj : allObjects){
412                 ArrayList Objects = (ArrayList) tempObj;
```

```
413                 for (Object obj : Objects){
414                     if (classFlag != null){
415                         if (obj.getClass() != classFlag && obj.
                                getClass() != Integer.class){
416                             if (classFlag == Race.class){
417                                 Race.removedIds = removedIds;
418                             }
419                             if (classFlag == Segment.class){
420                                 Segment.removedIds = removedIds;
421                             }
422                             if (classFlag == Stage.class){
423                                 Stage.removedIds = removedIds;
424                             }
425                             classFlag = null;
426                             removedIds.clear();
427
428
429                         }
430                         else{
431                             Integer removedId = (Integer) obj;
432                             removedIds.add(removedId);
433
434                         }
435                     }
436                     String objClass = obj.getClass().getName();
437                     System.out.println(objClass);
438                     if (obj.getClass() == Rider.class){
439                         Rider newRider = (Rider) obj;
440                         allRiders.add(newRider);
441                         System.out.println("NEW RIDER");
442                     }
443                     if (obj.getClass() == Team.class){
444                         Team newTeam = (Team) obj;
445                         allTeams.add(newTeam);
446                         System.out.println("NEW TEAM");
447                     }
448                     if (obj.getClass() == Result.class){
449                         Result newResult = (Result) obj;
450                         allResults.add(newResult);
451                         System.out.println("NEW RESULT");
452                     }
453                     if (obj.getClass() == Stage.class){
454                         Stage newStage = (Stage) obj;
455                         allStages.add(newStage);
456                         System.out.println("NEW STAGE");
457                         classFlag = Stage.class;
458                     }
459                     if (obj.getClass() == Race.class){
460                         Race newRace = (Race) obj;
461                         allRaces.add(newRace);
```

```
462                     System.out.println("NEW Race");
463                     classFlag = Race.class;
464                 }
465                 if (obj.getClass() == Segment.class){
466                     Segment newSeg = (Segment) obj;
467                     allSegments.add(newSeg);
468                     System.out.println("NEW SEGMENT");
469                     classFlag = Segment.class;
470                 }
471
472
473                 System.out.println(obj.getClass());
474             }
475         }
476         if (classFlag == Race.class){
477             Race.removedIds = removedIds;
478         }
479         if (classFlag == Segment.class){
480             Segment.removedIds = removedIds;
481         }
482         if (classFlag == Stage.class){
483             Stage.removedIds = removedIds;
484         }
485
486             this.riderManager.setAllTeams(allTeams);
487             this.riderManager.setAllRiders(allRiders);
488             Race.allRaces = allRaces;
489             Race.loadId();
490             Stage.allStages = allStages;
491             Stage.loadId();
492             Segment.allSegments = allSegments;
493             Segment.loadId();
494             Result.allResults = allResults;
495             ois.close();
496
497         }
498         catch (Exception ex) {
499             ex.printStackTrace();
500         }
501
502     }
503
504     @Override
505     public void removeRaceByName(String name) throws
            NameNotRecognisedException {
506         boolean found = false;
507         for (int raceId : Race.getAllRaceIds()){ //Throwing
                this exception is impossible!
508             try {
509                 if (name == Race.getRaceName(raceId)){
```

```
510                     Race.removeRace(raceId);
511                 }
512             }
513             catch(Exception c){
514                 assert(false); //Assert false for this!
515             }
516 
517         }
518         if (!found){ throw new NameNotRecognisedException("Name
                not in System.");}
519 
520     }
521 
522     @Override
523     public LocalTime[] getGeneralClassificationTimesInRace(int
            raceId) throws IDNotRecognisedException {
524         // TODO Auto-generated method stub
525         return null;
526     }
527 
528     @Override
529     public int[] getRidersPointsInRace(int raceId) throws
            IDNotRecognisedException {
530         // TODO Auto-generated method stub
531         return null;
532     }
533 
534     @Override
535     public int[] getRidersMountainPointsInRace(int raceId)
            throws IDNotRecognisedException {
536         // TODO Auto-generated method stub
537         return null;
538     }
539 
540     @Override
541     public int[] getRidersGeneralClassificationRank(int raceId)
             throws IDNotRecognisedException {
542         // TODO Auto-generated method stub
543         return null;
544     }
545 
546     @Override
547     public int[] getRidersPointClassificationRank(int raceId)
            throws IDNotRecognisedException {
548         // TODO Auto-generated method stub
549         return null;
550     }
551 
552     @Override
553     public int[] getRidersMountainPointClassificationRank(int
```

```
            raceId) throws IDNotRecognisedException {
554         // TODO Auto-generated method stub
555         return null;
556     }
557
558 }
```