

# Test Plan for Assessment 3



## Test Considerations

### Introduction

Testing is a critical part of any software release. This Test Plan document details any testing we have done for this assessment, whether functional or nonfunctional. Generally we have used unit tests on new classes, class extensions and their methods, and black-box testing on the game as a whole and other specifically designed tests where relevant. As we have used a form of test-driven *AGILE* Development, we have tested all relevant places throughout the implementation, and we have ensured that each stage of the project is fully working as intended at the end of each of our sprints.

### Test Coverage

Clearly the original parts of the project have been well tested in the previous assessment by Team HEC as detailed by their [testing document](#). Any requirements that have passed their tests will be assumed as working at the start of this assessment. To utilise time as efficiently as possible we will therefore not **re-test** any of these requirements, unless we are modifying the classes or methods that are specifically fulfilling these requirements.

All Classes will be **thoroughly** tested using both Unit Tests and black box testing to ensure that they are functioning correctly and robustly.

Accessors and mutators will be given a **low precedence** in testing are unlikely to harbour any serious bugs that will affect critical system functionality.

Third-party libraries and in-built Java functionality will assumed effectively stable and therefore will **not** be tested. We have deliberately chosen third-party libraries that are very widely used in the hope that they will have very few bugs, and none that are critical to system functionality.

### Testing Environments

Java is a flexible programming language and can be run on a very wide range of systems. One of our project requirements is that the project runs on the computers in the Department Labs, therefore all whole system tests- such as black-box testing on the whole project will be run on these to ensure correct/expected functionality when run on these computers

Other testing, such as unit testing, that examines source code may be run on other systems, as the core reason for these tests is to ensure the correct code at low-level rather than functionality of the system as a whole.

## **Testing Methods**

We have used a wide variety of tests and testing methods within our software testing to ensure our game is free from errors. These testing methods have included:

### Unit Testing

We isolated the whole program into separate testable units to see if they were fit for use and whether they corresponded to the design specifications. We built test classes using JUnit which instantiated the objects with specific input data and called all relevant methods belonging to the unit class and tested their outputs against expected outputs using JUnit assertions. We have decided to make use of JUnit due to it's wide use in testing applications, it allows us to make many assertions about the functionality of a class and check to ensure that the assertions remain true. By verifying that the output of each method corresponded to the expected value (through the use of assertions), we were able to show that our class units worked as intended. These tests were run near the end of this development cycle to ensure that the code was all working in its finished state.

To run the JUnit tests, the author of each section of code wrote tests and ran them using eclipse's built in JUnit functionality as it gave useful feedback about not only which tests failed but also why they failed

### Black Box Testing

Brief example of how black box testing has been used- forming a range of test cases, testing them and the results.

### System Testing

We will use system testing to test the system works as a whole. This performed by the developers playing the game, they will press all buttons and ensure that

everything works as intended and the system is robust enough to handle unexpected button clicks and events.

### Usability Testing

We put the game in front of a completely new user with the game manual to test if they could easily understand it. We set some initial test instructions for the user and some test conditions to decide how we would measure if the test was passed.

### Acceptance Testing

Due to time constraints in this very short assessment run, we haven't been able to run any acceptance testing for the project as yet. This is not a major issue as we do not yet have a final product anyway.

## Test Design & Results

### Unit Tests for CargoGoal Class

A brief test class is written using JUnit that instantiates a Special Cargo Goal and calls several of its methods and the expected outputs were matched against real time outputs. All relevant methods and instance data members associated with CargoGoal Class have been tested via JUnit so coverage is extensive.

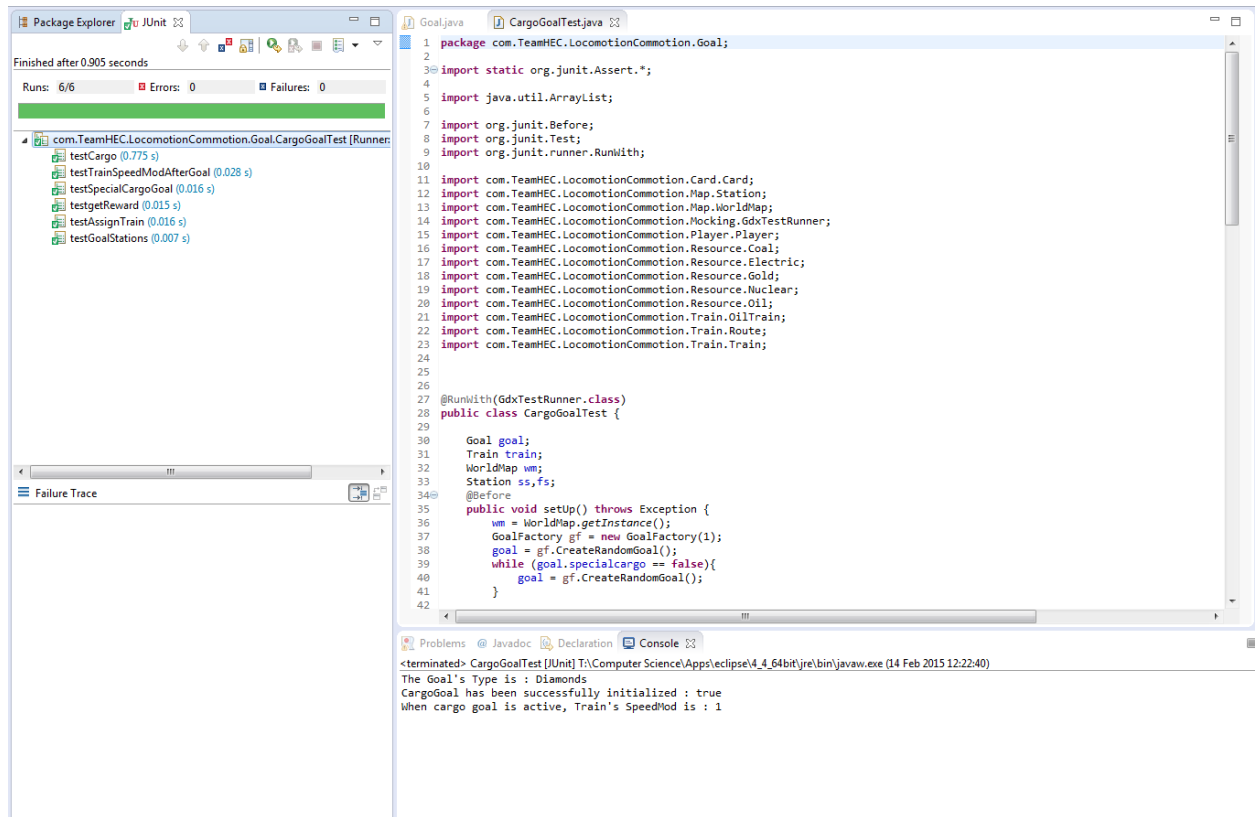
The Methods & Data members of the CargoGoal class that we tested are –

- boolean **specialcargo**
- String **getCargo()**
- String **getSStation()**
- String **getFStation()**
- void **assignTrain()** & Train **getTrain()**
- int **getReward()**
- void **goalComplete()**

**Note:** We found an error/bug where the train's speedmod won't reset to the original value after Special Cargo Goal Completion and fixed it.

Test Description	Expected Result	Result	Proof of Result	Status
Instantiate Special Cargo Goal and test the value of ' <i>specialcargo</i> '	Special Cargo Goal is successfully instantiated and ' <i>specialcargo</i> ' is set to true to confirm it	Special Cargo Goal is successfully instantiated and ' <i>specialcargo</i> ' is set to true to confirm it	The absence of errors on building and the assertion ' <i>specialcargo</i> ' being true via JUnit test [testSpecialCargoGoal()]	Pass
Get the cargo type for Special Cargo Goal	A string 'Diamonds' is returned	A string 'Diamonds' is returned	Assertion of the string 'Diamonds' via JUnit test [testCargo()]	Pass
Check if the Special Cargo Goal has a start station	True	True	Assertion via JUnit test [testGoalStations()]	Pass

Check if the Special Cargo Goal has a finish station	True	True	Assertion via JUnit test [testGoalStations()]	Pass
Get the reward for the Special Cargo Goal	An integer greater than zero	An integer greater than zero	Assertion of a reward being returned which greater than zero (via Junit test )	Pass
Check if the CargoGoal has successfully been assigned to a player's train	The selected train should be assigned to the goal and there is a decrease in the SpeedMod of the assigned train	The selected train should be assigned to the goal and there is a decrease in the SpeedMod of the assigned train	Assertion via Junit test [testAssignTrain()] , the ability to call getTrain() method of the goal and the decreased speedmod value, as shown in Fig 1	Pass
Check if a special cargo goal can successfully be completed	Execution of method goalComplete() should be successful	Execution of method goalComplete() should be successful	The absence of errors on calling the method and assertion via JUnit test [testTrainSpeedModAfterGoal()]	Pass
Check if the speedMod of the player's train is restored back after cargo goal completion	SpeedMod is back to original value	SpeedMod is back to original value	Assertion via JUnit test [testTrainSpeedModAfterGoal()]	Pass



(Fig 1 - Running of JUnit Tests for CargoGoalTest class and some outputs)

## Unit Test for GoalFactory Class

To ensure that the extended methods in Goal Factory class worked as intended, the JUnit test class for Goal Factory Class was extended and the extended methods were called with test data, and the outputs were verified via JUnit.

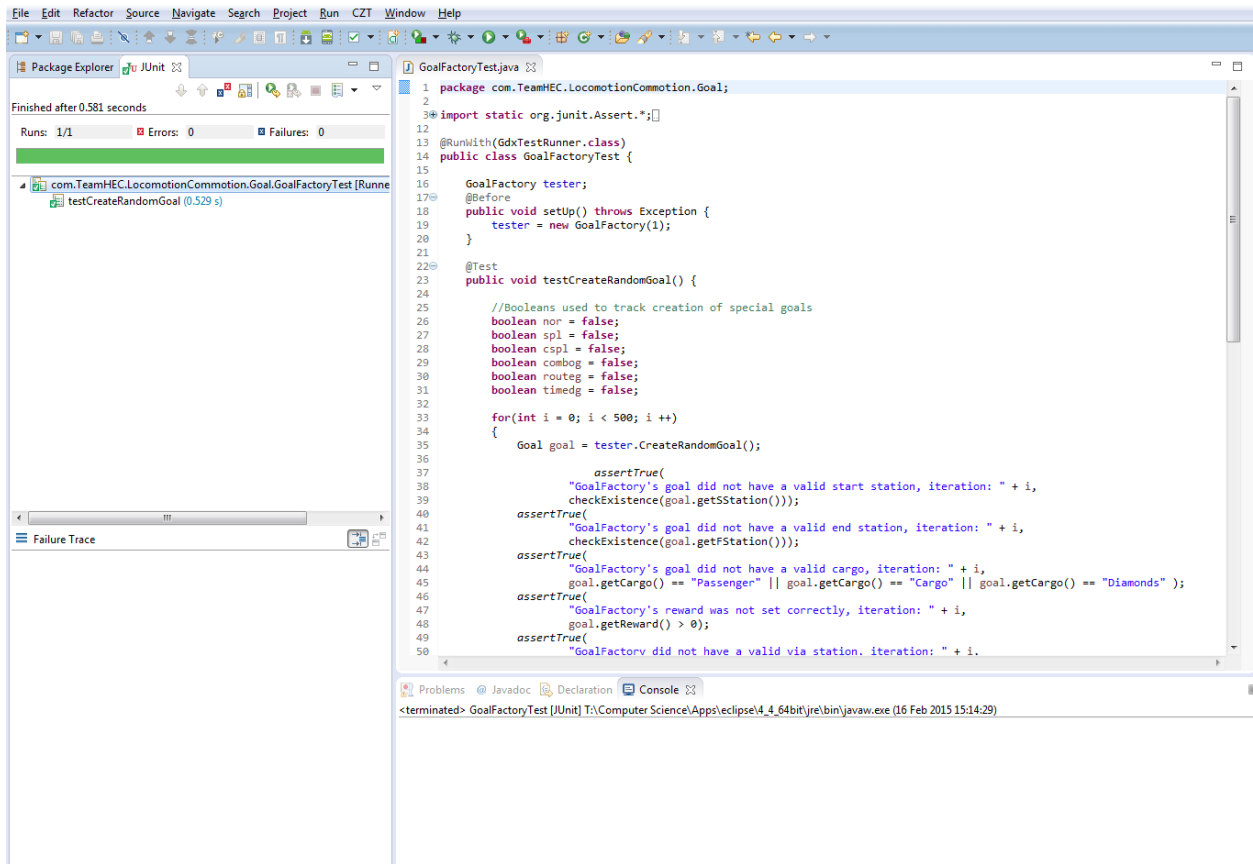
A series of 500 random goals were created via iteration in the JUnit test class and the following extended functionalities were tested.

Test Description	Expected Result	Result	Proof of Result	Status
Ability to create one or more non-special goals <b>(Func.Sys.2.2)</b> <b>(Func.Sys.2.3)</b>	One or more non-special goals are created by the GoalFactory	One or more non-special goals are created by the GoalFactory	Assertion via JUnit test [testCreateRandomGoal()]	Pass
Ability to create one or more Special goals <b>(Func.Sys.2.2)</b> <b>(Func.Sys.2.3)</b>	One or more non-special goals are created by the GoalFactory	One or more Special goals are created by the GoalFactory	Assertion via JUnit test [testCreateRandomGoal()]	Pass
Ability to create one or more Special CargoGoals <b>(Func.Sys.2.2)</b> <b>(Func.Sys.2.3)</b>	One or more non-special goals are created by the GoalFactory	One or more Special Cargo Goals are created by the GoalFactory	Assertion via JUnit test [testCreateRandomGoal()]	Pass
Ability to create one or more Route based goals <b>(Func.Sys.2.2)</b> <b>(Func.Sys.2.3)</b>	One or more Route based goals are created by the GoalFactory	One or more Route based goals are created by the GoalFactory	Assertion via JUnit test [testCreateRandomGoal()]	Pass
Ability to create one or more Timed based goals <b>(Func.Sys.2.2)</b> <b>(Func.Sys.2.3)</b>	One or more Timed based goals are created by the GoalFactory	One or more Timed based goals are created by the GoalFactory	Assertion via JUnit test [testCreateRandomGoal()]	Pass
Ability to create one or more	One or more Special Combo goals are created	One or more Special Combo goals are created	Assertion via JUnit test	Pass

Special Combo goals <b>(Func.Sys.2.2)</b>	by the GoalFactory	by the GoalFactory	[testCreateRandomGoal()]	
Test each goal to ensure a station isn't included that is currently unfixable (making the goal impossible to complete) <b>(User.GP.2.5)</b> <b>(Func.Sys.2.1)</b> <b>(Func.Sys.2.2)</b> <b>(Func.Sys.2.3)</b>	No goals are created with an unfixable station as either the start, via, or final stations.	No goals were created with an unfixable station as either the start, via, or final stations.	Assertion via JUnit test [testCreateRandomGoal()]	Pass

**Note** : All the above tests have been performed before 'score' class was added to the game design





(Fig 2 - Running of JUnit Tests for GoalFactoryTest class)

## Unit Tests for TimedGoal Class

A small test class was written for the new TimedGoal class, that generates an instance of TimedGoal and tests each of its methods to ensure that they are operating as intended. All new methods relevant to TimedGoal were tested and their outputs were asserted using expected outputs through JUnit. All of the methods tested in the original GoalTest class were also implemented to extensive testing has been upheld for this class.

The Methods and Attributes of the Timed class that we tested are –

- Station **sStation**
- Station **fStation**
- Train **train**
- int **reward**
- boolean **isSpecial**
- int **turnLimit**
- int **startTurn**
- Station **getSStation()**
- Station **getFStation()**
- Train **getTrain()**
- int **getReward()**
- boolean **isSpecial()**
- int **getTurnLimit()**
- int **getStartTurn()**

Test Description	Expected Result	Result	Proof of Result	Statu s
Generates an instance of TimedGoal and test the value of isSpecial	isSpecial returns true	isSpecial returned true	The absence of errors upon instantiating TimedGoal and the assertion via JUnit	Pass

			testIsSpecial()	
Test the value of sStation	compareStations returns true	compareStations returned true	Assertion via JUnit [testGoalStations()]	Pass
Test the value of fStation	compareStations returns true	compareStations returned true	Assertion via JUnit [testGoalStations()]	Pass
Test the value of train	Train is correctly assigned to TimedGoal	Train was correctly assigned to TimedGoal	Assertion via JUnit [testAssignTrain()]	Pass
Test the value of reward	reward should be greater than 0	reward was greater than 0	Assertion via JUnit [testGetReward()]	Pass
Test the value of turnLimit	turnLimit should be greater than 0	turnLimit was greater than 0	Assertion via JUnit [testTurnLimit()]	Pass
Test the value of startTurn	startTurn should be greater than or equal to 0	startTurn was greater than or equal to 0	Assertion via JUnit [testStartTurn()]	Pass

**Note:** After this test startTurn was changed to be 0 at the goal's creation and later changed to reflect the current turn counter once a train was assigned to it, at all points the startTurn is greater than or equal to 0.

```

76  assertEquals(compareStations(goal.getStation()), true); //At the LocomotionCom goal's start station
77  assertTrue(compareStations(goal.getStation())); //if the TimedGoal has a finish station
78  }
79
80  @Test
81  public void testAssignTrain() {
82      assertTrue("goal.getTrain() == train"); //TimedGoal has been successfully been assigned to a train
83  }
84
85
86  @Test
87  public void testGetReward() {
88      assertTrue(goal.getReward() > 0); //TimedGoal's reward is successfully generated and is greater than zero
89  }
90
91  @Test
92  public void testIsSpecial() {
93      assertTrue(goal.isSpecial() == true); //On initialisation of TimedGoal the isSpecial bool of Goal class should be set to true
94  }
95
96  @Test
97  public void testTurnLimit() {
98      assertTrue(goal.getTurnLimit() > 0); //On initialisation the goal would be allocated a turn limit that is greater than 0
99  }
100
101  @Test
102  public void testStartTurn() {
103      assertTrue(goal.getStartTurn() >= 0); //If the TimedGoal has been allocated a start turn then it would be greater than or equal to 0
104  }
105
106  public boolean compareStations(String Sname){
107      for (int i = 0; i < wm.stationsList.size(); i++){
108          if (Sname == wm.stationsList.get(i).getName()){
109              return true;
110          }
111      }
112      return false;
113  }
114
115

```

JUnit 22  
 finished after 0.298 seconds  
 Runs: 4/4 Errors: 0 Failures: 0  
 com.TeamHEC.LocomotionCommotion.Goal.TimedGoalTest [Runner: JUnit 4] (0.270 s)  
 Failure Trace

## Unit Tests for RouteGoalClass

A small test class was written for the new RouteGoal class, that generates an instance of RouteGoal and tests each of its methods to ensure that they are operating as intended. All new methods relevant to RouteGoal were tested and their outputs were asserted using expected outputs through JUnit. All of the methods tested in the original GoalTest class were also implemented to extensive testing has been upheld for this class.

The Methods and Attributes of the Timed class that we tested are –

- Station **sStation**
- Station **viaStation**
- Station **fStation**
- Train **train**
- int **reward**
- boolean **isSpecial**
- Station **getSStation()**
- Station **getViaStation()**
- Station **getFStation()**
- Train **getTrain()**
- int **getReward()**
- boolean **isSpecial()**

Test Description	Expected Result	Result	Proof of Result	Status
Generates an instance of RouteGoal and test the value of isSpecial	isSpecial returns true	isSpecial returned true	The absence of errors upon instantiating TimedGoal and the assertion via JUnit [testIsSpecial()]	Pass
Test the value of sStation	compareStations returns true	compareStations returned true	Assertion via JUnit [testGoalStations()]	Pass
Test the value of viaStation	compareStations returns true	compareStations returned true	Assertion via JUnit [testGoalStations()]	Pass
Test the value of fStation	compareStations returns true	compareStations returned true	Assertion via JUnit [testGoalStations()]	Pass
Test the value of train	Train is correctly assigned to TimedGoal	Train was correctly assigned to TimedGoal	Assertion via JUnit [testAssignTrain()]	Pass

Test the value of reward	reward should be greater than 0	reward was greater than 0	Assertion via JUnit [testgetReward()]	Pass
--------------------------	---------------------------------	---------------------------	---------------------------------------	------

```
47 Coal coal = new Coal(200);
48 Oil oil = new Oil(200);
49 Electric electric = new Electric(200);
50 Nuclear nuclear = new Nuclear(200);
51 ArrayList<Card> cards = new ArrayList<Card>();
52 ArrayList<Goal> goals = new ArrayList<Goal>();
53 ArrayList<Train> trains = new ArrayList<Train>();
54
55 Player player = new Player(
56     name,
57     score,
58     gold,
59     coal,
60     electric,
61     nuclear,
62     oil,
63     cards,
64     goals,
65     trains);
66
67
68 train = new OilTrain(0, true, new Route(WorldMap.getInstance().AMSTERDAM), player);
69
70 goal.assignTrain(train);
71
72 }
73
74 @Test
75 public void testGoalStations() {
76     assertTrue(compareStations(goal.getStartStation())); //if the RouteGoal has a start station
77     assertTrue(compareStations(goal.getViaStation())); //if the RouteGoal has a via station
78     assertTrue(compareStations(goal.getFinishStation())); //if the RouteGoal has a finish station
79 }
80
81 @Test
82 public void testAssignTrain() {
83     assertTrue("goal.getTrain() == train"); //RouteGoal has been successfully been assigned to a train
84 }
85
86
JUnit 22
ished after 0.32 seconds
runs: 4/4 Errors: 0 Failures: 0
com.TeamHEC.LocomotionCommotion.Goal.RouteGoalTest [Runner: JUnit 4] [0.282 s]
Failure Trace
```

## Unit Tests for ComboGoal Class

A small test class was written for the new ComboGoal class, that generates an instance of ComboGoal and tests each of its methods to ensure that they are operating as intended. All new methods relevant to ComboGoal were tested and their outputs were asserted using expected outputs through JUnit. All of the methods tested in the original GoalTest class were also implemented to extensive testing has been upheld for this class.

The Methods and Attributes of the Timed class that we tested are –

- Station **sStation**
- Station **viaStation**
- Station **fStation**
- Train **train**
- int **reward**
- boolean **isSpecial**
- int **turnLimit**
- int **startTurn**
- Station **getSStation()**
- Station **getViaStation()**
- Station **getFStation()**
- Train **getTrain()**
- int **getReward()**
- boolean **isSpecial()**
- int **getTurnLimit()**
- int **getStartTurn()**

Test Description	Expected Result	Result	Proof of Result	Status
Generates an instance of RouteGoal and test the value of isSpecial	isSpecial returns true	isSpecial returned true	The absence of errors upon instantiating TimedGoal and the assertion via JUnit [testIsSpecial()]	Pass
Test the value of sStation	compareStations returns true	compareStations returned true	Assertion via JUnit [testGoalStations()]	Pass
Test the value of viaStation	compareStations returns true	compareStations returned true	Assertion via JUnit [testGoalStations()]	Pass

Test the value of fStation	compareStations returns true	compareStations returned true	Assertion via JUnit [testGoalStations()]	Pass
Test the value of train	Train is correctly assigned to TimedGoal	Train was correctly assigned to TimedGoal	Assertion via JUnit [testAssignTrain()]	Pass
Test the value of reward	reward should be greater than 0	reward was greater than 0	Assertion via JUnit [testgetReward()]	Pass
Test the value of turnLimit	turnLimit should be greater than 0	turnLimit was greater than 0	Assertion via JUnit [testTurnLimit()]	Pass
Test the value of startTurn	startTurn should be greater than or equal to 0	startTurn was greater than or equal to 0	Assertion via JUnit [testStartTurn()]	Pass

**Note:** After this test startTurn was changed to be 0 at the goal's creation and later changed to reflect the current turn counter once a train was assigned to it, at all points the startTurn is greater than or equal to 0.

```

69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```


## **Black Box Testing (Goal Extensions)**

<b>Test Description</b>	<b>Expected Result</b>	<b>Result</b>	<b>Proof of Result</b>	<b>Status</b>
For Special Cargo Goals, Cargo should be set to Diamonds and a player should be able to choose such a goal	'Diamonds' is displayed on the card for Special Cargo Goal and a player is able to add that goal to his list of goals	'Diamonds' is displayed on the card for Special Cargo Goal and a player is able to add that goal to his list of goals	As shown in Fig. 1, 'Diamonds' is displayed on the Special Cargo Goal and it has been added to player's list of chosen goals	Pass
For Special route based goals, a via/route station should be specified and a player should be able to choose such a goal	A route station is specified for Special route based goal and a player is able to add that goal to his list of goals	A route station is specified for Special route based goal and a player is able to add that goal to his list of goals	As shown in Fig. 1, a route/via station 'Madrid' is displayed on the Route Based Goal and it has been added to player's list of chosen goals	Pass
For Special Timed based goals, a turn limit should be specified and a player should be able to choose such a goal	A turn limit is specified for Special Time based goal and a player is able to add that goal to his list of goals	A turn limit is specified for Special Time based goal and a player is able to add that goal to his list of goals	As shown in Fig. 2, a Turn Limit of '3' is displayed on a Special Route Goal and it has been added to player's list of chosen goals	Pass
For Special Combo Goals, a turn limit and a route station should be specified and a player should be able to choose such a goal	Both turn limit and route station are specified on the the goal card and a player is able to add that goal to his list of goals	Both turn limit and route station are specified on the the goal card and a player is able to add that goal to his list of goals	As shown in Fig. 2, a Turn Limit of '12' and a route station 'Moscow' are displayed on the Combo Goal and it has been added to player's list of chosen goals	Pass
Attempt to complete a standard (non-special) goal <b>(Func.SYS.2.4)</b>	The non-special goal is completed and the player is rewarded for the completion of the goal	The non-special goal is completed and the player is rewarded for the completion of the goal	As shown in Fig 4, the non-special goal is completed and the player is rewarded and the Goal Completion message is displayed.	Pass
Attempt to complete a Special Cargo Goal <b>(Func.SYS.2.4)</b>	The Special Cargo Goal is completed and the player is	The Special Cargo Goal is completed and the player is	There is no screenshot of this test due to it	Pass



	rewarded for the completion of the goal	rewarded for the completion of the goal	displaying the same message and using the same method as a standard goal.	
Attempt to complete a Special Route based goal <b>(Func.SYS.2.4)</b>	The Special Route Goal is completed and the player is rewarded for the completion of the goal	The Special Route Goal is completed and the player is rewarded for the completion of the goal	There is no screenshot of this test due to it displaying the same message and using the same method as a standard goal.	Pass
Attempt to complete a Special timed goal <b>(Func.SYS.2.4)</b>	The Special Timed Goal is completed and the player is rewarded for the completion of the goal	The Special Timed Goal is completed and the player is rewarded for the completion of the goal	There is no screenshot of this test due to it displaying the same message and using the same method as a standard goal.	Pass
Attempt to complete a Combo Goal <b>(Func.SYS.2.4)</b>	The Special Combo Goal is completed and the player is rewarded for the completion of the goal	The Special Combo Goal is completed and the player is rewarded for the completion of the goal	There is no screenshot of this test due to it displaying the same message and using the same method as a standard goal.	Pass
When a player fails to complete a goal in certain time limit after assigning the goal to his train, Goal Failed Message should be displayed. <b>(Func.SYS.2.4)</b>	Goal Failed Message is displayed whenever the turn limit is exceeded after assigning the goal to a train	- On exceeding the turn limit, Goal Failed message is displayed when a train has been to the start station, otherwise it is not. (Bug)	As shown in Fig. 3, a message was displayed to the player about him not being able to complete the goal in the required time limit	Pass (Fail if train hasn't been to start station )

**Note :** All the above tests have been performed before 'score' class was added to the game design



# GOAL SCREEN

TICKET NO: 1234-5678-90A

Diamonds	REWARD	826
Vilnius	START DATE	0
Vienna	ROUTE	Any

Special Cargo Goal

≡

TICKET NO: 1234-5678-90A

Cargo	REWARD	1820
Helsinki	START DATE	0
Reykjavik	ROUTE	Madrid

Route Based Goal


≡

TICKET NO: 1234-5678-90A

TYPE	REWARD	
FROM	START DATE	
TO	ROUTE	

≡

(Figure 1)



# GOAL SCREEN

TICKET NO: 1234-5678-90A

Passenger	Turn Limit: 3	REWARD	520
Bern	START DATE	0	
Berlin	ROUTE	Any	

Turn Based Goal

≡

TICKET NO: 1234-5678-90A

Passenger	Turn Limit: 12	REWARD	3604
Monaco	START DATE	0	
Prague	ROUTE	Moscow	

Combo Goal

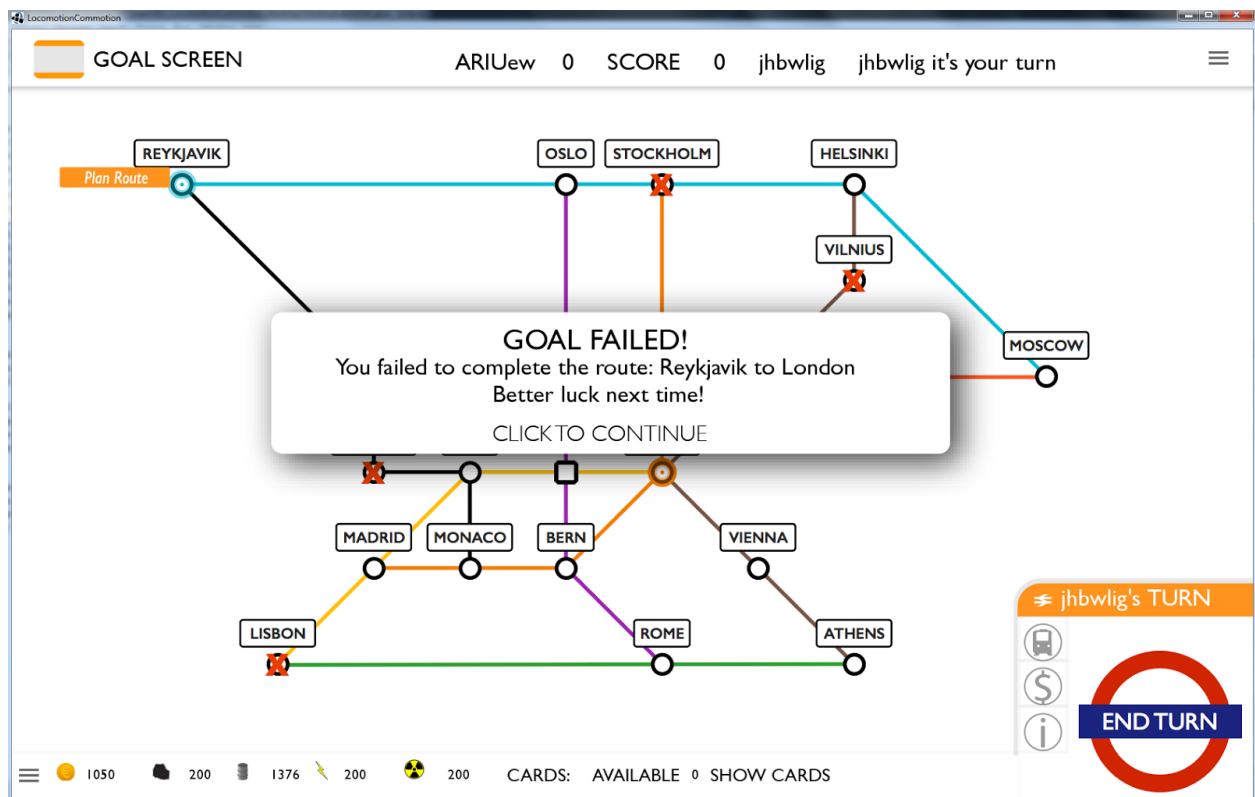
≡

TICKET NO: 1234-5678-90A

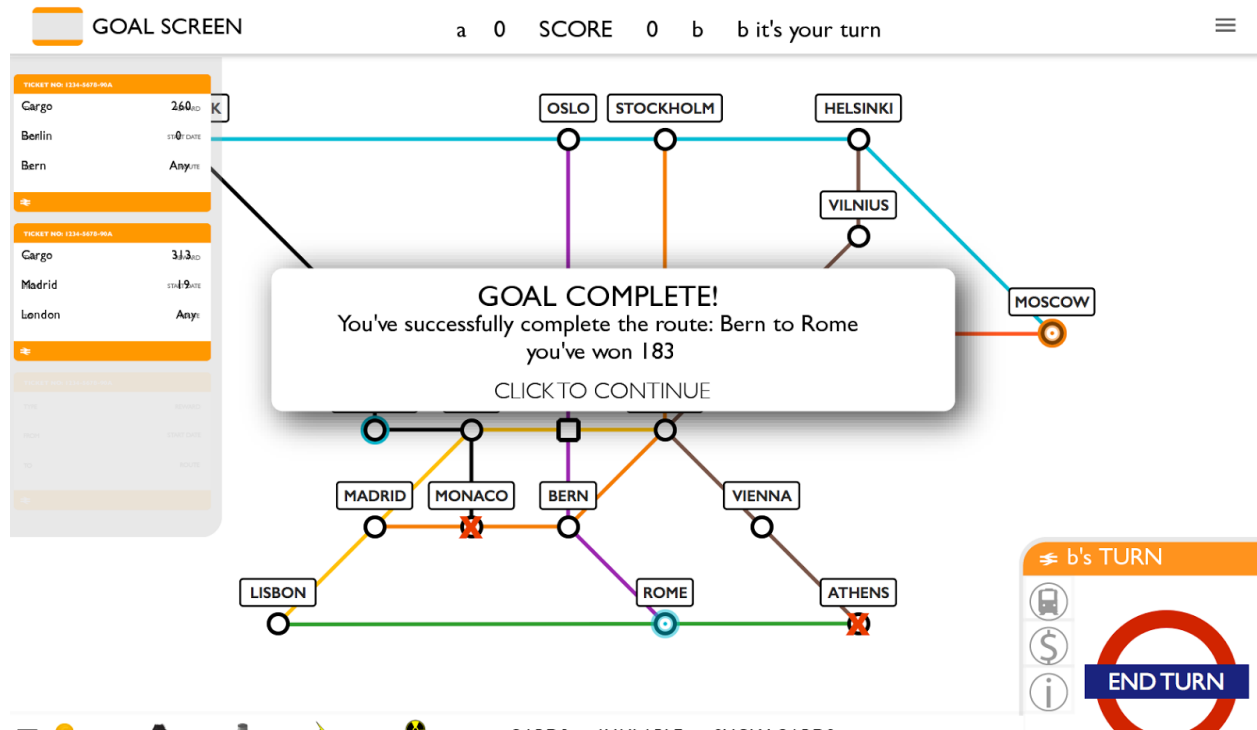
Cargo	REWARD	443
Vienna	START DATE	2
Berlin	ROUTE	Any

≡

(Figure 2)



(Figure 3 - Message displayed on failing a goal)



(Figure 4 - Message displayed on completing a goal successfully)

### Unit testing for Shop

In order to ensure that the methods added to the Shop class, the shopTest was extended to test the newly added buyTrain function. All acceptable inputs for the buyTrain method were tested and their outcomes were asserted through expected outcomes in JUnit. The methods originally implemented in the Shop were also tested to ensure extensive testing has been upheld in this class.

Test Description	Expected Result	Result	Proof of Result	Status
The buyTrain function is called with "Coal" as the parameter [ Func.SYS.4.9 ]	The player's gold is reduced by the price of the train and the train is added to their list of trains	The player's gold was reduced by the cost of the train and the train was added to their list of trains	The JUnit test for the size of the players list of trains increasing by one and their gold reducing by the cost of the train	Pass
The buyTrain function is called with "Oil" as the parameter [ Func.SYS.4.9 ]	The player's gold is reduced by the price of the train and the train is added to their list of trains	The player's gold was reduced by the cost of the train and the train was added to their list of trains	The JUnit test for the size of the players list of trains increasing by one and their gold	Pass

			reducing by the cost of the train	
The buyTrain function is called with "Electric" as the parameter [ <b>Func.SYS.4.9</b> ]	The player's gold is reduced by the price of the train and the train is added to their list of trains	The player's gold was reduced by the cost of the train and the train was added to their list of trains	The JUnit test for the size of the players list of trains increasing by one and their gold reducing by the cost of the train	Pass
The buyTrain function is called with "Nuclear" as the parameter [ <b>Func.SYS.4.9</b> ]	The player's gold is reduced by the price of the train and the train is added to their list of trains	The player's gold was reduced by the cost of the train and the train was added to their list of trains	The JUnit test for the size of the players list of trains increasing by one and their gold reducing by the cost of the train	Pass
The repairStation function was called on a test station	The players gold is decremented by 300 and the station is repaired	the players gold was decremented by 300 and the station was repaired	The JUnit test for the station isFaulty becoming true and the players gold reducing by 300	Pass
the repair station function was called on a test station when the player had no gold	the station remains faulty and the players gold remains the same	the station remained faulty and the players gold did not change	the JUnit test for the station isFaulty remaining true and the players gold not changing	Pass
the upgrade station function was called on a test station	The station level increasing by one and the players gold reducing by 400	the station level increased by 1 and the players gold was reduced by 400	the JUnit test for the station level increasing by one the the player's gold reducing by 400	Pass
The upgrade station function was called on a test station when the player had no gold	the station level and the player's gold should both remain unchanged	the stations level and the players gold both remained unchanged	The JUnit test for the players gold remaining the same and the station's level not changing	Pass

## Unit test for TeleportCard

Originally the Teleportation WildCard did not function as intended, upon activation it would teleport the player's first train to a single pre-set station (London). This has now been altered to teleport a random train (owned by the player) to a random station on the map. A small extension was made to the original JUnit test class in order to ensure this new functionality was implemented correctly and worked as intended. The test table below only includes the extensions to the JUnit test as all original functionality and unit testing will have been performed by the previous team.

Test Description	Expected Result	Result	Proof of Result	Status
Activating the card moves a random train (owned by the player) to a random location. <b>(Func.SYS.4.8)</b>	A random train is moved to a random station.	A random train has moved to a random station.	Assertion via JUnit {testImplementCard()}	Pass

```

66
67
68 @Test
69 public void testImplementCard() {
70     int index[] = new int[5];
71     boolean moved = false;
72
73     for(int i=0; i<5; i++) {
74         testImplementCard();
75         assertEquals("Train made index was not set correctly", player.getTrains().get(0).getRoute().getRouteIndex() == 0);
76         assertEquals("Train connection travelled was not set correctly", player.getTrains().get(0).getRoute().getConnectionTravelled() == 0);
77
78         for(int j=0; j<map.getStations().stationList.size(); j++) {
79             if(player.getTrains().get(0).getRoute().getStation() == nullMap.getStations().stationList.get(j)) {
80                 index[i] = j;
81             }
82         }
83     }
84
85     for(int i=0; i<5; i++) {
86         if(index[i] != index[i+1]) {
87             moved = true;
88         }
89     }
90
91     assertEquals("Train currentMapObj was not set correctly", moved);
92 }
93
94 @Test
95 public void testTeleportCard() {
96     assertEquals("TeleportCard's owner was not set correctly", test(0).getOwner() == player);
97     assertEquals("TeleportCard's reactive was not set correctly", test(0).getReactive() == cardReactive);
98     assertEquals("TeleportCard's name was not set correctly", test(0).getName() == cardName);
99 }
100 }
101

```

Finished after 0.383 seconds

Run: 2/2    0 Errors    0 Failures

com.team01.LocomotionCommunication.Card.TeleportCardTest (Runner: JUnit4 (0.29.0))

testImplementCard (0.251 s)

testTeleportCard (0.044 s)

Failure Trace

### **Black box testing for TeleportCard**

<b>Test Description</b>	<b>Expected Result</b>	<b>Result</b>	<b>Proof of Result</b>	<b>Status</b>
Activating the card moves a random train (owned by the player) to a random location. <b>(Func.SYS.4.8)</b>	A random train is moved to a random station.	A random train has moved to a random station.	See before activation and after activation screenshots below. The Orange train has moved from London to Madrid.	Pass





## Unit Testing for Faults

A short test class was written for testing faults within the game. Although the 'Faults' implementation spans several several classes, we have tested it as one system.

The Methods and Attributes of we have tested for Faults are-

- WorldMap Class
  - generateFaults()
- Station Class
  - isFaulty()
  - isRepairable()
  - makeFaulty()
  - fixFault()
  - getStationLevel()
  - upgradeStation()
  - getFaultRate()

The faults section of the project was intended to fulfil the following Requirements:

**User.GP.6.3** : There **MUST** be at least two obstacles in the game.

The requirement **User.UI.10** also refers to faults, however this is about the GUI which is to be tested using a black box test, later in this document.

Test Description	Expected Result	Result	Proof of Result	Status
Station start level is 0	Station initialises at level 0	Station initialises at level 0	Assertion via JUnit passes	Pass
Station start fault rate is 0.1.	Station fault rate returns 0.1 when at level 0.	Station fault rate returns 0.1 when at level 0.	Assertion via JUnit passes	Pass
Station can be upgraded	Station level increases when upgraded	Station level increases when upgraded	Assertion via JUnit passes	Pass
Station fault rate is lower at higher levels	Station fault rate decreases when upgraded	Station fault rate decreases when upgraded	Assertion via JUnit passes	Pass
Station is initially not faulty.	Station isFaulty() method returns false.	Station isFaulty() method returns false.	Assertion via JUnit passes	Pass
Station is initially repairable.	Station isRepairable() method returns true.	Station isRepairable() method returns true.	Assertion via JUnit passes	Pass
Station can be made faulty	Station isFaulty() returns true,	Station isFaulty() returns true,	Assertion via JUnit passes	Pass


	when station is broken.	when station is broken.		
Station can be fixed if not permanently damaged	Station isFaulty() returns false again after fixFault() is called	Station isFaulty() returns false again after fixFault() is called	Assertion via JUnit passes	Pass




```

1 package com.TeamHEC.LocomotionCommotion.Map;
2
3 import static org.junit.Assert.*;
4
24
25 @RunWith(GdxTestRunner.class)
26 public class FaultsTest {
27
28     Station testStation;
29
30     @Before
31     public void setup(){
32         testStation = WorldMap.getInstance().stationsList.get(0);
33     }
34
35     @Test //test upgradeStation()
36     public void upgradeStationTest(){
37         assertTrue("Station initialises as Level 0", testStation.getStationLevel() == 0);
38         assertTrue("Station fault rate is initially 0.1%", testStation.getFaultRate() == 0.1);
39         for(int i = 1; i < 5; i++){
40             testStation.upgradeStation();
41             assertTrue("Station can be upgraded", testStation.getStationLevel() == i);
42             System.out.println(testStation.getFaultRate());
43             assertTrue("Station fault rate decreases when upgraded", testStation.getFaultRate() <= 0.1);
44         }
45     }
46
47     @Test //test isFaulty(), makeFaulty(), fixFault()
48     public void makeFaultyTest(){
49         assertFalse("Station initialises as not-faulty", testStation.isFaulty());
50         assertTrue("Station initialises as repairable", testStation.isRepairable());
51         testStation.makeFaulty();
52         assertTrue("Station can be made faulty", testStation.isFaulty());
53         if(testStation.isRepairable()){
54             testStation.fixFault();
55             assertFalse("Station can be successfully fixed", testStation.isFaulty());
56         }
57     }
58
59
60     @Test //test generateFaults()
61     public void generateFaultsTest() {
62         for(int i = 0; i < 500; i++){
63             WorldMap.getInstance().generateFaults();
64         }
65
66         Boolean flag = false; //creates a flag to determine if any of the stations in newMap are faulty
67
68         for(int i = 0; i < WorldMap.getInstance().stationsList.size(); i++){
69             if(WorldMap.getInstance().stationsList.get(i).isFaulty()) {
70                 flag = true;
71                 WorldMap.getInstance().stationsList.get(i).fixFault();
72             }
73         }
74         assertTrue("Some faults are successfully generated at random.", flag);
75     }
76
77
78 }

```

Runs: 3/3    ❌ Errors: 0    ❌ Failures: 0

▼  com.TeamHEC.LocomotionCommotion.Map.FaultsTest [Runner: JUnit 4] (0.094 s)

-  generateFaultsTest (0.073 s)
-  makeFaultyTest (0.016 s)
-  upgradeStationTest (0.005 s)

### **Black box testing for Faults**

The faults section of our extension to the HEC project newly fulfils the requirements:

**Func.OD.4.2** Game **SHOULD** alert players when a random event occurs.

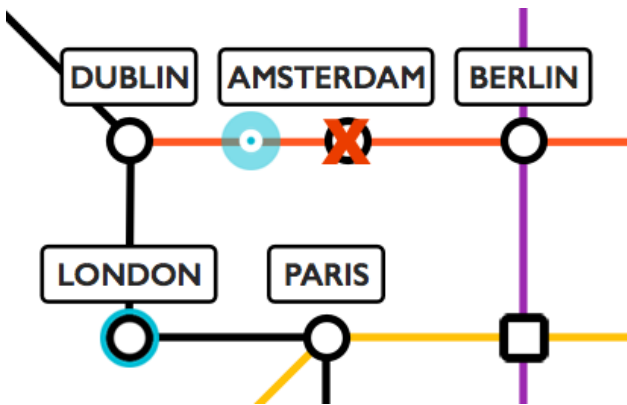
**USER.GP.6.3** There **MUST** be at least two obstacles in the game.

**USER.UI.10** **MUST** display hazards on screen.

<b>Test/Scenario</b>	<b>Expected Result</b>	<b>Result</b>	<b>Proof of Result</b>	<b>Status</b>
Run a game to see faults appear at random on the map.	Faults will randomly occur throughout the course of the game.	As expected, several faults appear on the map.	See the screenshots below this table.	Pass
Attempt to move to a faulty station	The train cannot move to that station and is returned to the previous station. A warning message is fired.	The train cannot move to that station and is returned to the previous station. A warning message is fired.	See the screenshots below this table.	Pass
Attempt to leave a faulty station	The train cannot leave the station. A warning message is fired.	The train cannot leave the station. A warning message is fired.	See the screenshots below this table.	Pass
Repair a station by clicking the "repair" button on the station info panel	The station is no longer faulty and the station icon goes to the standard one	The station is no longer faulty and the station icon goes to the standard one	As shown in the screenshot below, the station was faulty and is then repaired	Pass
Attempt to repair a non-repairable faulty station	Warning message fires to prompt user of the illegal move	Warning message fires to prompt user of the illegal move	See the screenshots below this table.	Pass

Test 1:

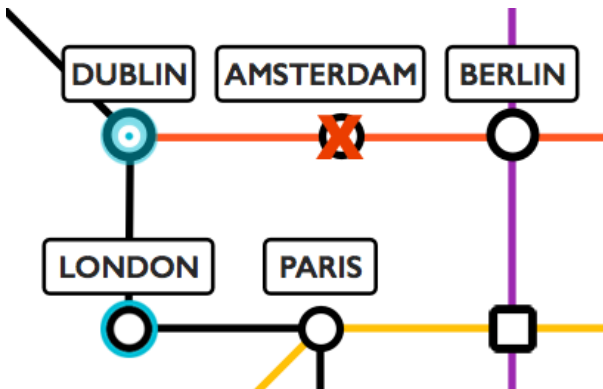
Test 2:



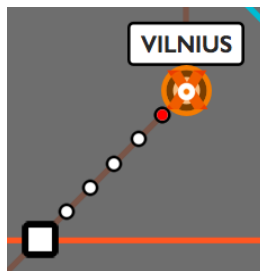
Sorry

The station is faulty. You must repair it to continue!

CLICK TO CONTINUE



Test 3:

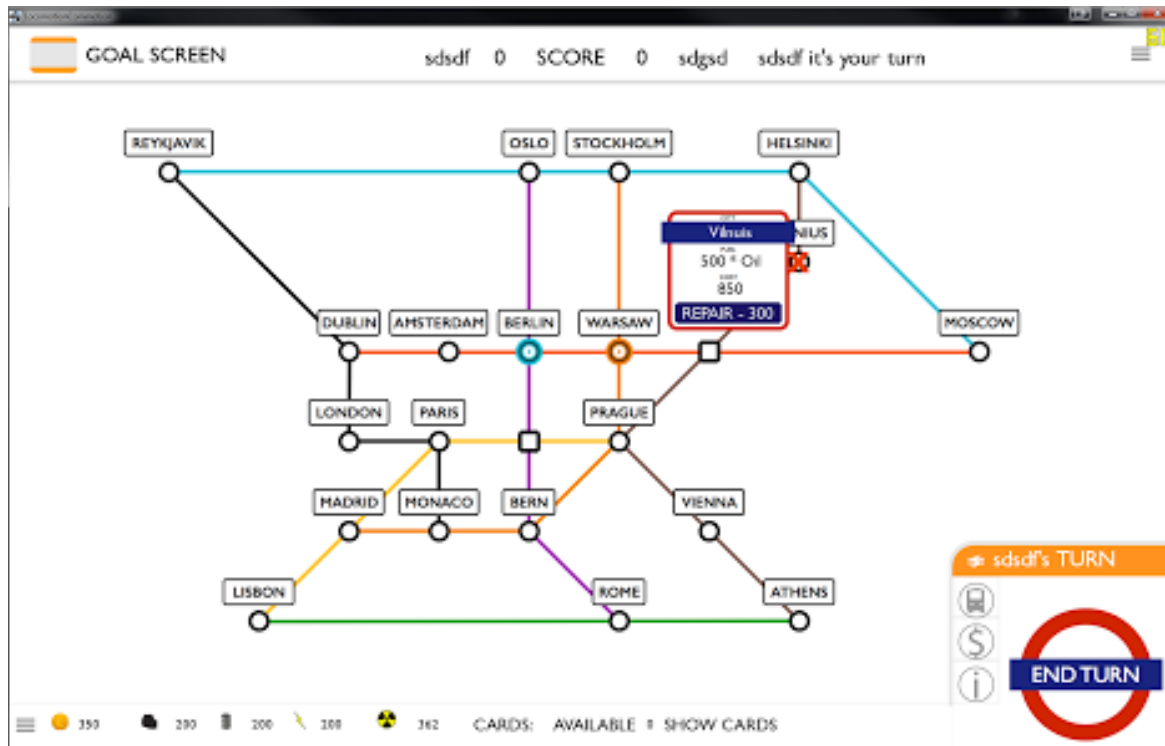


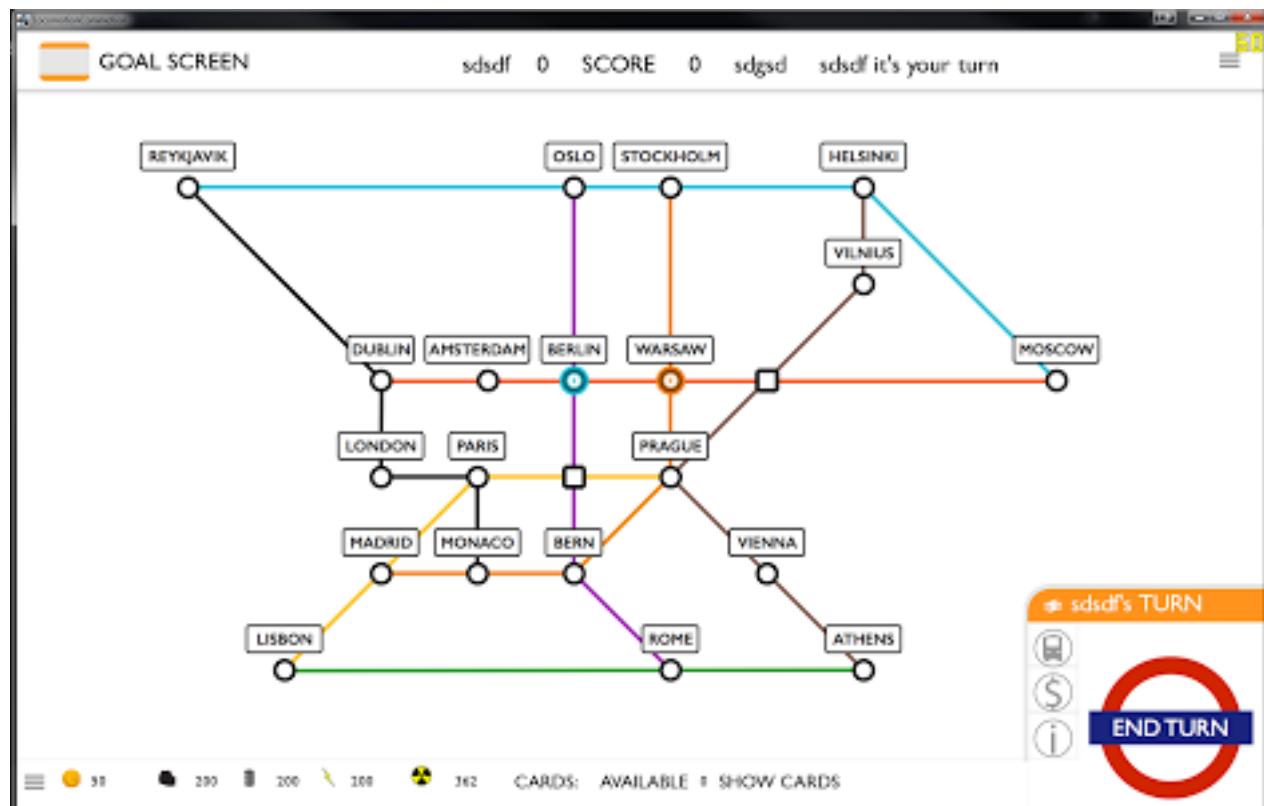
Sorry

The station is faulty. You must repair it to continue!

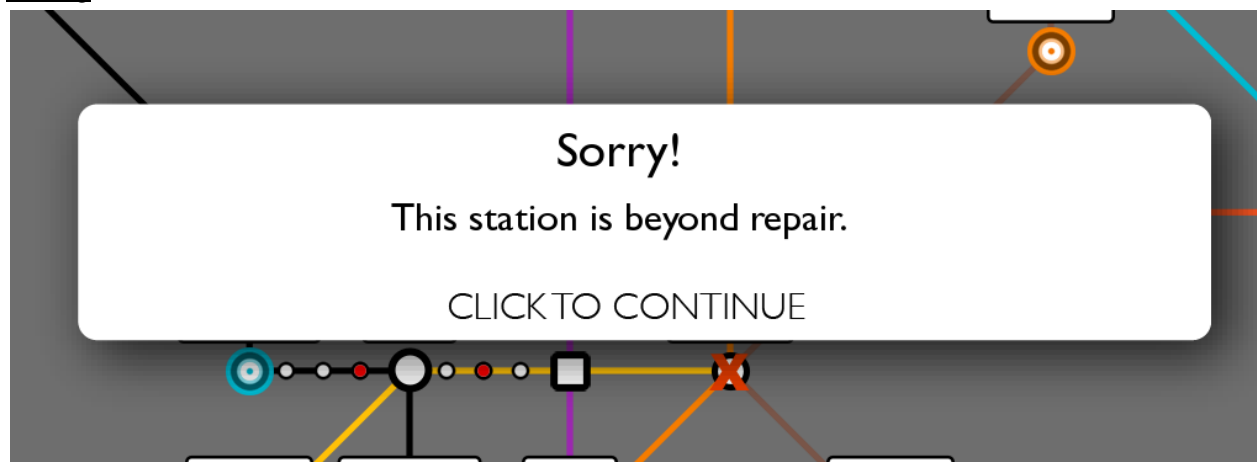
CLICK TO CONTINUE

Test 4:





Test 5



## Black box testing for Score

Most of the work in creating a score system was in pulling apart the "gold" and "points" systems that was implemented when we received the project from the previous group. The actual implementation of Score when finished is quite low level, with a mutator in Player, a Score class extending Resource and a method in goal that adds score.

**Func.SYS.1 :** System **must** keep both players' score

**Func.SYS.8.1 :** System **must** be able to add points to a players score

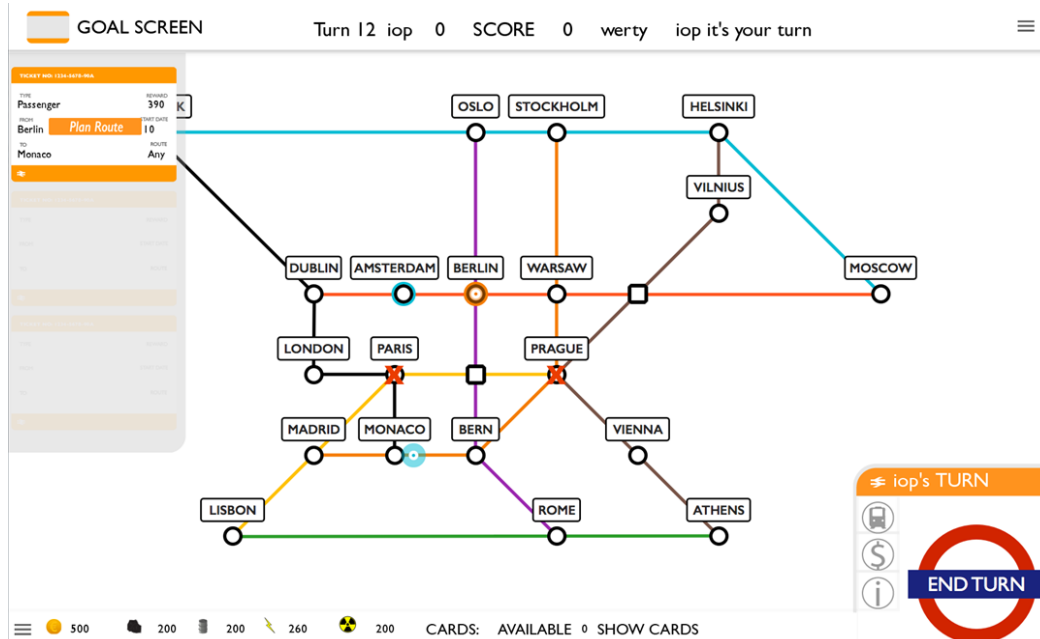
**Func.Sys.8.2 :** System **must** be able to assign points to a randomly generated goals.

**Func.Sys.4.1 :** System **must** track of players resources in real time

It was decided that none of these components needed testing using JUnit, as any tests of mutators etc are trivial. Black box testing however would show the successful awarding of points to a player and their presence on screen.

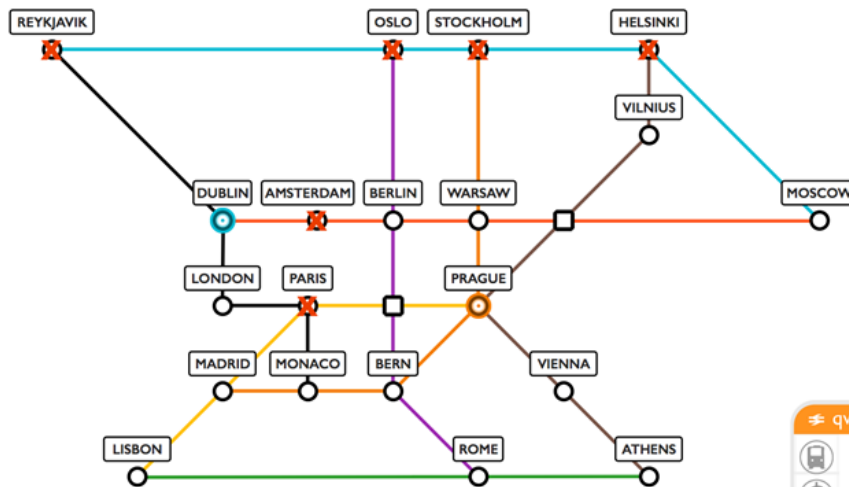
The only time a player may receive points is when they complete a goal:

A goal from Berlin to Monaco was completed by player 1 with a reward of 390 gold and 3 Score.









qwerty's TURN

END TURN

CARDS: AVAILABLE 0 SHOW CARDS



Thank you, you've completed the game!  
The Game was a draw!! Well done to both players  
CLICK TO CONTINUE

uiop's TURN

END TURN

CARDS: AVAILABLE 0 SHOW CARDS

## **Usability Testing**

### **Test Conditions**

The test will be run on the standard lab setup in the department labs as one of our requirements is for the system to work on these machines. Developers will not be present in order to ensure that the participants are not given hints either deliberately or inadvertently. All efforts will be made to ensure that the users are acting independently at all times.

The participants will be picked by being anyone passing by in the corridor outside the Software Laboratories in the Department of Computer Science, University of York. However, we will ensure that none of the participants have played any version of Locomotion Commotion before.

### **Method**

1. Give a pair of new users the game manual to read.
2. Open the game for the users- we are testing usability of the game, not the users' abilities to open an executable file.
3. Ask the users to complete each task specified in the table below.
4. After each task is complete, *immediately* ask the users to rate how easy it was to understand and, where relevant, how challenging it was to complete.
5. Ease of understanding will be marked on a 5 point scale, where Very Easy (5) is our optimum result
6. Level of Challenge will be marked on a 9 point scale where 1 is too easy, 5 is just right and 9 is too hard. 5 is our optimum result again. The actual score for this will be the difference between 5 and the result recorded.
7. If the marks for the tests average out as 3.5 out of 5 or higher, the test passes.

### **Participants**

The participants were all aged 18-20, which, although it is a limited range, fits the demographic of our expected audience for the game. There were 4 participants, of which 1 was female and 3 were male.

### **Results**

<b>Test/Scenario</b>	<b>Expected Result</b>	<b>(Proof of) Result</b>	<b>Status</b>
Start a game (50 turn limit)	Will be measured by Ease of Understanding only as this is not an aspect of gameplay. The score will be 3.5 or greater.	Scored an average of 5.0 for Ease of Understanding.	Pass.

Select a goal	Will be measured by Ease of Understanding only as this is not an aspect of gameplay. The score will be 3.5 or greater.	Scored an average of 4.5 for Ease of Understanding.	Pass.
Assign a goal to a train	Will be measured by Ease of Understanding only as this is not an aspect of gameplay. The score will be 3.5 or greater.	Scored an average of 4.0 for Ease of Understanding.	Pass.
Complete a goal	Will be measured on both scales. These scores will each be 3.5 or greater.	Scored an average of 4.5 for Ease of Understanding and 4.0 for Level of Challenge.	Pass.
Complete a further two goals	Will be measured on both scales. These scores will each be 3.5 or greater.	Scored an average of 4.5 for Ease of Understanding and 3.5 for Level of Challenge.	Pass.
Finish the game	Will be measured on both scales. These scores will each be 3.5 or greater.	Scored an average of 4.0 for Ease of Understanding and 4.0 for Level of Challenge.	Pass.

#### Additional Notes

The participants praised the comprehensiveness of the user manual as well as the on-screen prompts.

The Level of Challenge became an average of 3.5 for Level of Challenge for completing a total of three goals, this is a near miss. It was thought that the Level of Challenge was slightly too hard, particularly for turn-limited goals. Some attention should be paid to this in further development and further tests should be undertaken in the next phase.

### Trivial Requirements :

We felt that some of the system requirements, that had been traced by HEC already in the previous assessment didn't need re-testing as the code hasn't been modified. The following are those requirements **Func.Sys.2.3, Func.Sys.3.1, Func.Sys.3.2, Func.Sys.3.3, Func.Sys.4.3, Func.Sys.4.5, Func.Sys.4.6,, Func.Sys.4.8, Func.Sys.4.9, Func.Sys.4.10, Func.Sys.5.1, Func.Sys.6.1, Func.Sys.6.2, Func.Sys.6.3, Func.Sys.7.2, Func.Sys.10, Func.Sys.11.4, Func.Sys.12.2, Func.Sys.15.**

Some of the optional System Requirements were left out due to time constraints :

**Func.Sys.2.5** : System **should** have special goals which provide Wildcards as reward

**Func.Sys.4.5** : Stations **Could** randomly generate extra resources at the end of a turn.