# TEST CASE REPORT

## Test Coverage/Completeness

A series of interface tests, unit tests and integration tests have been evidenced in this document. These tests were all undertaken upon the Assessment 4 TaxE project by team LYS.

All of the system and user requirements for this Assessment, available in 'Extra Requirements' at http://keemyb.github.io/SEPR-LYS-A4/, were tested thoroughly. The tests are available to view below.

The unit tests are primarily focused upon the Track Modification feature as opposed to the Replay Mode feature. This was due to the team deciding that running an interface test and actually watching the replay would be a more efficient manner of testing this feature. Buying a connection was one aspect of Track Modification that was difficult to undertake via a unit test and as such was interface tested. However, a lot of the Track Modification modules which were unit tested were also interface tested. This was done to ensure that the buttons on the interface worked correctly.

Most of the unit and interface tests focus on whether the code and the interface meet the requirements themselves. The interface tests particularly, with their test cases, make it easy to test solely for the purpose of whether the test meets the software and user requirements; sometimes even checking if the constraints have been met.

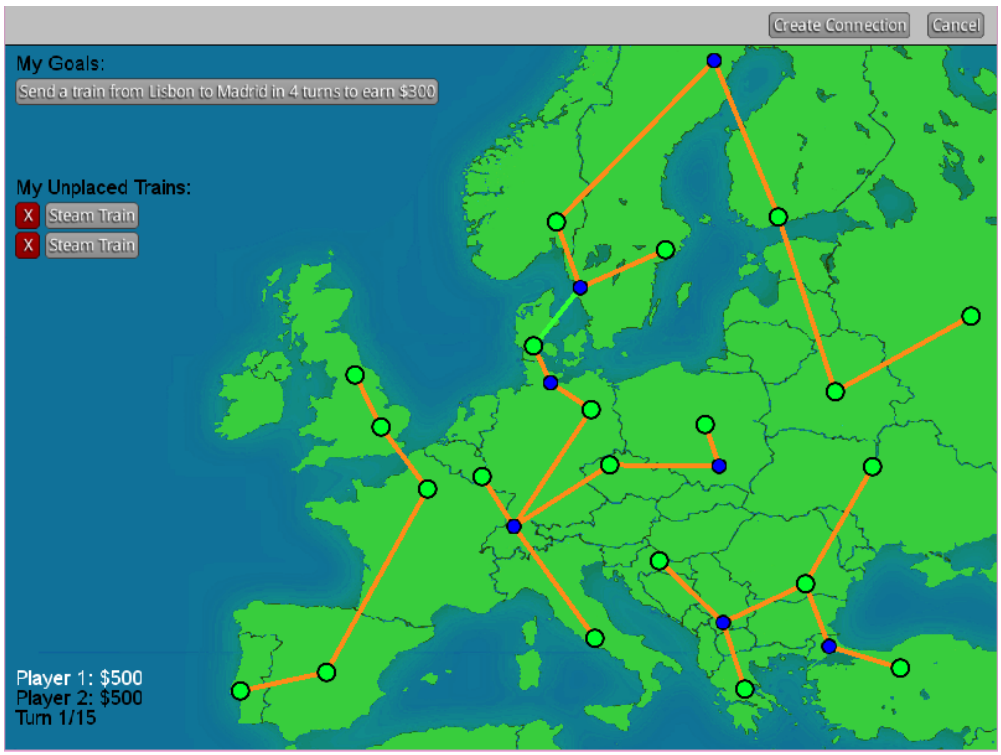
The unit tests are entirely made up of assertions which act almost like test cases of their own, as the programmer determines what the contents of the data structure should be after running a module. If possible for the module, a test was written immediately after the module in order to ensure that it produced the expected results. If that wasn't the case, then the code would be refactored in order to meet the expected result, and then the test would be rerun (regression testing). Due to having a certain amount of practice with writing code for this purpose now, the unit tests generally all pass upon the first try.

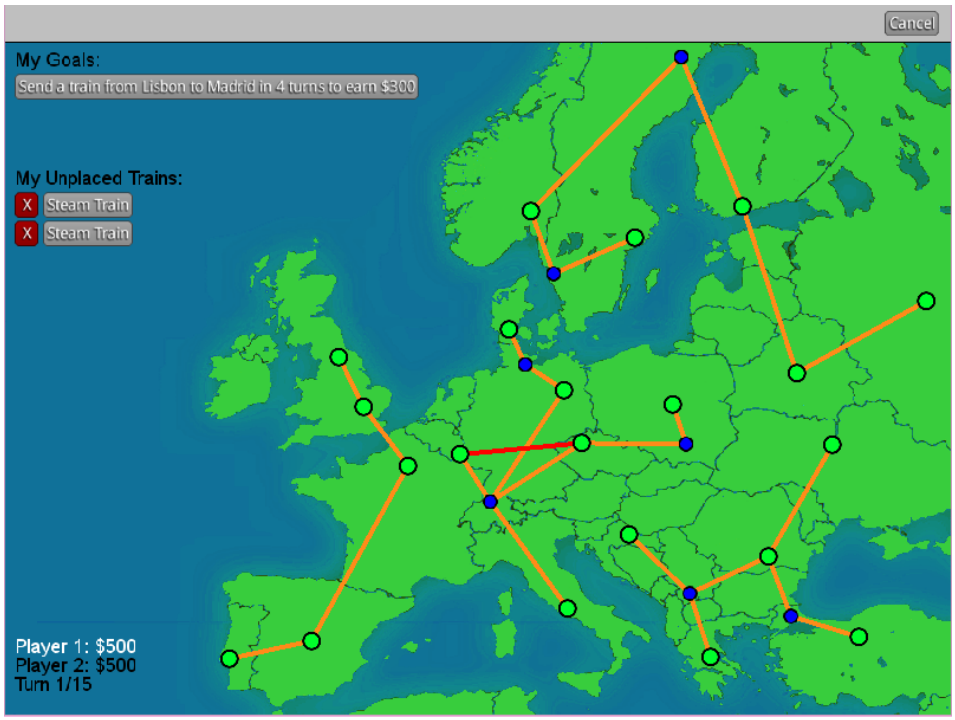
The unit tests combined with the interface tests cover all aspects of the required software features. Any other modifications made, which were tested by interface tests, weren't recorded as it was deemed more important to record the tests that actually met the necessary requirements.

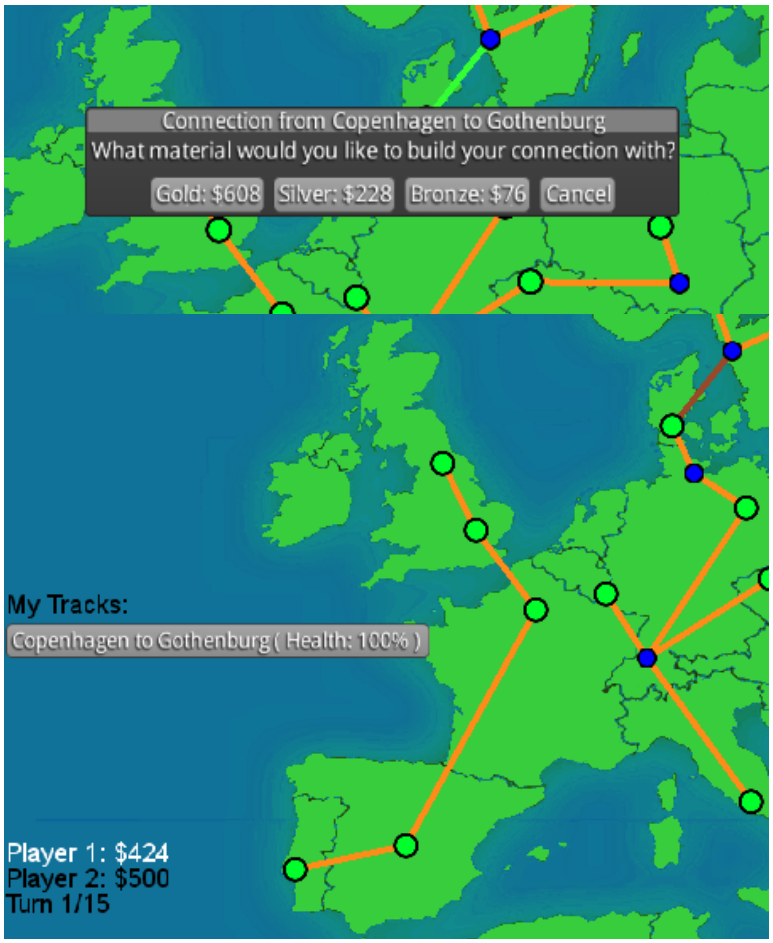
Performance tests were run to ensure that the problem encountered by team FVS, of too much CPU usage, hadn't returned in XYG's project. The game was also run in Windows and Linux to ensure that it continued to meet the OS requirements.

Usability testing was used in order to test that the user manual met its own requirements of accurately explaining how to play the TaxE game.

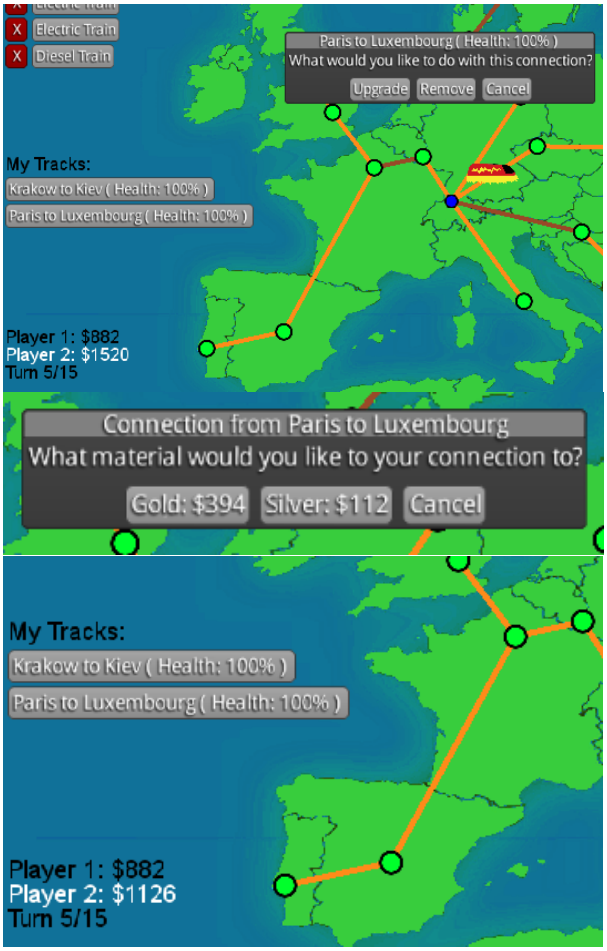
## Interface/System Tests

| | | | |
|---|---|---|---|
| **Id** | I1 | **Test Name** | InterfaceTest::testPlotValidConnection |
| **Related Code** | ConnnectionController:: createConnection<br>Map::prospectiveConnectionIsValid<br>Map::doesProspectiveConnectionIntersectExisting | | |
| **Related Requirements** | UR2, SR4, SR5 | **Category** | Interface Test |
| **Description** | "Add connection" button is selected.<br>Two Stations that do not have a connection between them are selected, such that the straight line between them does not cross any existing connection.<br><br>Assert that the newly drawn line is coloured green.<br>Assert that the "Add connection" button is still enabled/visible. | | |
| **Evidence** |  <br>The connection is from Copenhagen to Gothenburg | | |
| **Status** | Pass | | |

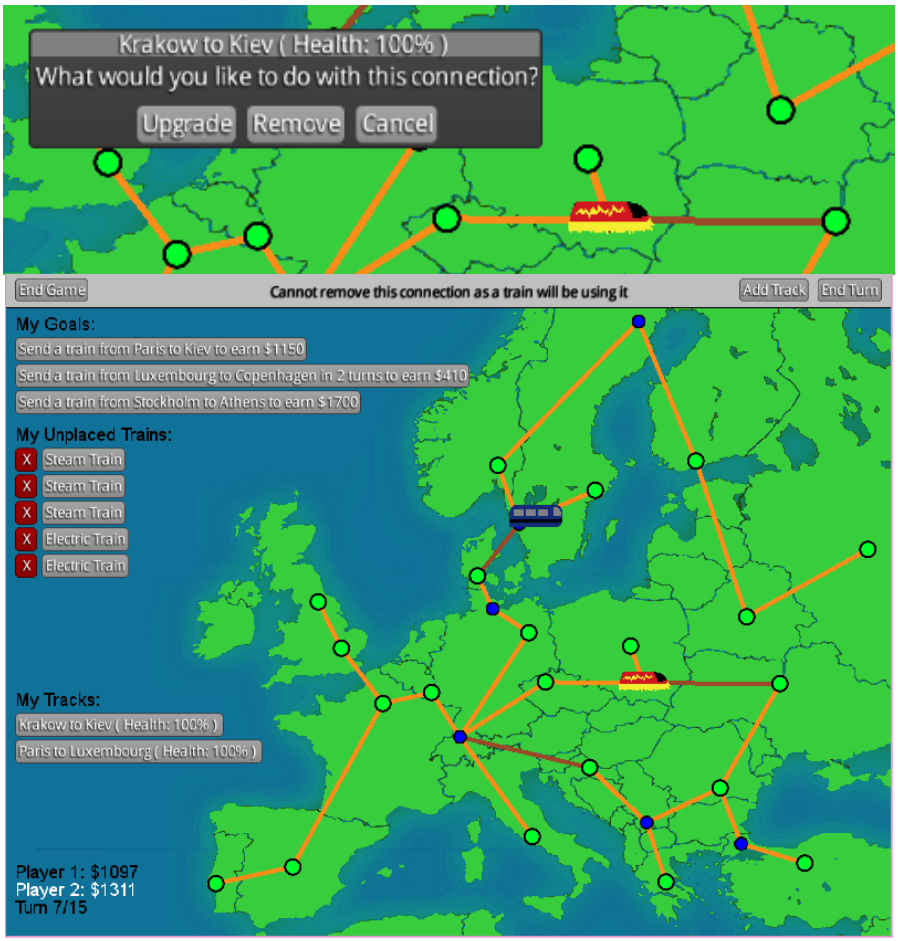
| | | | |
|---|---|---|---|
| **Id** | I2 | **Test Name** | InterfaceTest::testPlotInvalidConnection |
| **Related Code** | ConnectionController:: createConnection<br>Map:: prospectiveConnectionIsValid<br>Map:: doesProspectiveConnectionIntersectExisting<br>Map:: parseInvalidConnection | | |
| **Related Requirements** | UR2,<br>SR4, SR5,<br>C6 | **Category** | Interface Test |
| **Description** | “Add connection” button is selected.<br>Two Stations that do not have a connection between them are selected, such that the straight line between them does cross at least one existing connection.<br><br>Assert that the newly drawn line is coloured red.<br>Assert that the “Add connection” button is still disabled/invisible. | | |
| **Evidence** |  <br>The connection is from Prague to Luxembourg | | |
| **Status** | Pass | | |

| | | | |
|---|---|---|---|
| **Id** | I3 | **Test Name** | InterfaceTest::testBuyConnection |
| **Related Code** | ConnectionController:: createConnection<br>Map:: prospecctiveConnectionIsValid<br>Map:: addConnection<br>Player:: addOwnedConnection<br>Player:: spendMoney<br>Connection:: setOwner | | |
| **Related Requirements** | UR2,<br>SR4, SR5 | **Category** | Interface Test |
| **Description** | "Add connection" button is selected.<br>Two Stations that do not have a connection between them are selected, such that the straight line between them does not cross any existing connection.<br>"Add connection" button is selected.<br>A connection material is selected in the dialog box that appears.<br><br>Assert that the newly added connection is drawn in the appropriate colour for that material.<br>Assert that the purchasing player's funds have been updated. | | |
| **Evidence** |  | | |
| **Status** | Pass | | |

| | | | |
|---|---|---|---|
| **Id** | I4 | **Test Name** | InterfaceTest::testRepairConnection |
| **Related Code** | ConnectionConrtoller:: repairConnection<br>Connection:: calculateRepairCost<br>Connection:: repair<br>Player:: spendMoney | | |
| **Related Requirements** | UR6, SR15 | **Category** | Interface Test |
| **Description** | A non-gold connection is bought.<br>A train travels across the connection, so that it is damaged.<br>The same player that has bought the non-gold connection selects the connection they want from their My Tracks list.<br>“Repair Connection” button is selected.<br>“100%” is selected in the dialog box that appears.<br><br>Assert that the repaired connection has 100% health.<br>Assert that the repairing player’s funds have been updated. | | |
| **Evidence** |  <br>Player 2 repaired the Paris to Luxembourg track which had been travelled along, the money was removed from their total. | | |
| **Status** | Pass | | |

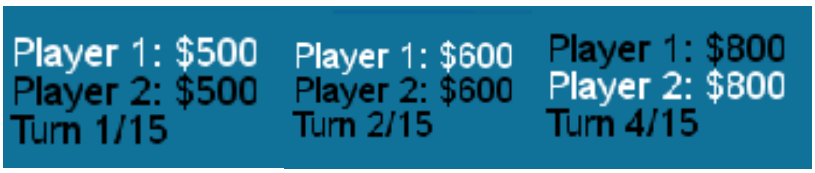
| | | | |
|---|---|---|---|
| **Id** | I5 | **Test Name** | InterfaceTest::testUpgradeConnection |
| **Related Code** | ConnectionController:: upgradeConnection<br>Connection:: calculateUpgradeCost<br>Connection:: upgrade<br>Connection:: isUpgradeable<br>Connection:: repair<br>Player:: spendMoney | | |
| **Related Requirements** | SR16 | **Category** | Interface Test |
| **Description** | A non-gold connection is bought.<br>The connection is selected from the My Tracks list.<br>"Upgrade" button is selected.<br>"Gold" is selected in the dialog box that appears.<br><br>Assert that the upgraded connection is now Gold.<br>Assert that the upgrading player's funds have been updated. | | |
| **Evidence** |  <br>Player 2 upgraded the Paris to Luxembourg connection to gold, the price of the upgrade was deducted from player 2's total<br>Evidence for UR4 and SR9 are shown above. | | |
| **Status** | Pass | | |

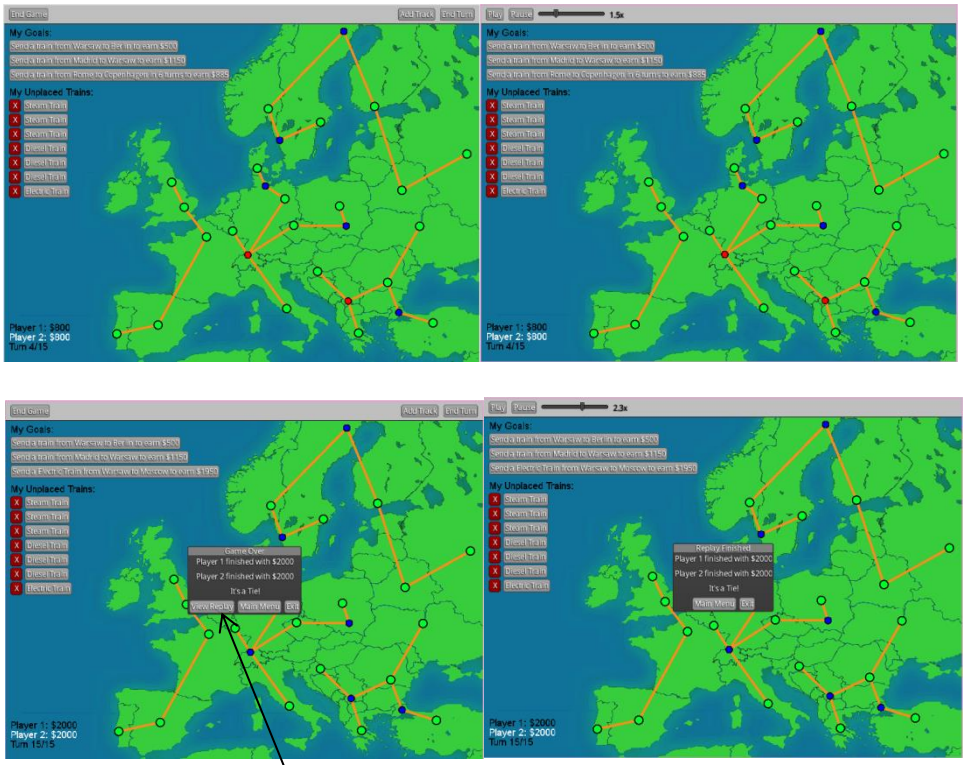
| | | | |
|---|---|---|---|
| **Id** | I6 | **Test Name** | InterfaceTest::testRemoveConnection |
| **Related Code** | ConnectionController:: removeConnection<br>ConnectionController:: showRemoveConnectionDialog<br>ConnectionController:: canBeremoved<br>Map:: removeConnection<br>Player:: removeOwnedConnection | | |
| **Related Requirements** | UR3,<br>SR6, SR7 | **Category** | Interface Test |
| **Description** | A connection is bought.<br>The connection is selected from the My Tracks list.<br>“Remove” button is selected.<br>Confirm?<br><br>Assert that the connection has been removed from the map. | | |
| **Evidence** |  <br>Player 2 has chosen to remove the connection from Paris to Luxembourg | | |
| **Status** | Pass | | |

| | | | |
|---|---|---|---|
| **Id** | I7 | **Test Name** | InterfaceTest::testRemoveConnectionTrain |
| **Related Code** | ConnectionController:: removeConnection<br>ConnectionController:: showRemoveConnectionDialog<br>ConnectionController:: canBeRemoved<br>Player:: getTrains<br>Player:: getRoute<br>Map:: getConnectionsBetween | | |
| **Related Requirements** | UR3, SR6, SR7, SR8 | **Category** | Interface Test |
| **Description** | A connection is bought.<br>A train is positioned such that it is on the new connection.<br>The connection is selected from the My Tracks list.<br>“Remove” button is selected.<br>Warning error is indicated.<br><br>Assert that the connection is not removed from the map. | | |
| **Evidence** |  <br>When Player 2 tried to remove the connection from Krakow to Kiev, a warning error indicated this was not possible as “a train will be using it” | | |
| **Status** | Pass | | |

| | | | |
|---|---|---|---|
| **Id** | I8 | **Test Name** | InterfaceTest::testPayConnectionRent |
| **Related Code** | Player::payConnectionRent<br>Connection::getOwner<br>Connection::getRentPayable | | |
| **Related Requirements** | UR6, SR12 | **Category** | Interface Test |
| **Description** | A connection is bought.<br>The player ends their turn.<br>The next player selects a route for one of their trains, such that it travels along the new connection once.<br>Both players end their turn until the train has completed movement over the new connection.<br><br>Assert that both players have had their funds adjusted. | | |
| **Evidence** |  <br>Player 1's train crossed Player 2's track and paid $100 rent<br>The evidence for SR11 can be seen her. | | |
| **Status** | Pass | | |

| | | | |
|---|---|---|---|
| **Id** | I9 | **Test Name** | InterfaceTest::testMaterials |
| **Related Code** | Connection:: Material | | |
| **Related Requirements** | UR4, SR10 | **Category** | Interface Test |
| **Description** | When selecting to create a track, assert that there are three different materials. | | |
| **Evidence** |  | | |
| **Status** | Pass | | |

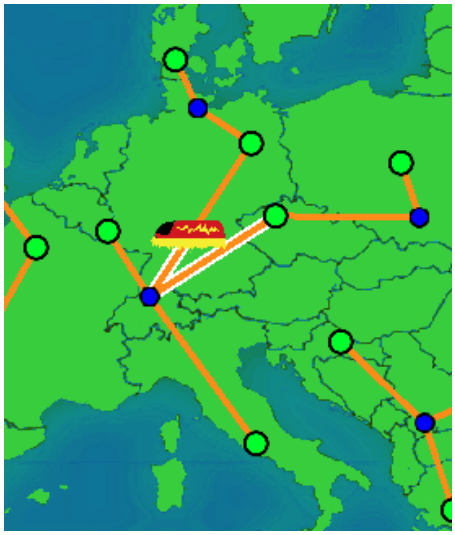
| | | | |
|---|---|---|---|
| **Id** | I10 | **Test Name** | InterfaceTest::testMoney |
| **Related Code** | PlayerManager:: turnChanged | | |
| **Related Requirements** | C2, C3 | **Category** | Interface Test |
| **Description** | After every turn, ensure that each player gets an extra $100. | | |
| **Evidence** |  | | |
| **Status** | Pass | | |

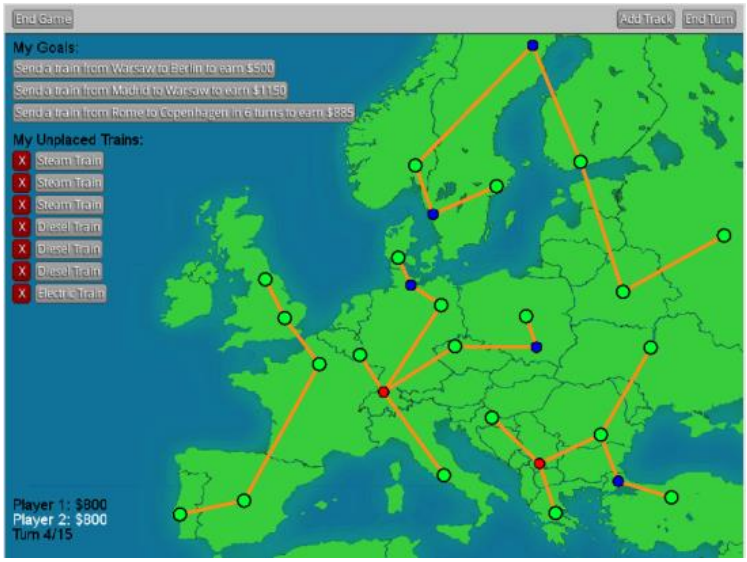
| | | | |
|---|---|---|---|
| **Id** | I11 | **Test Name** | InterfaceTest::testAbilityToReplay |
| **Related Code** | EventReplayer:: subscribeReplayEvent<br>This is used in every Controller class to record anything that has happened to the data stored within these classes.<br>EventReplayer::subscribeReplayModeEvent<br>Used with whole of EventReplayer class. | | |
| **Related Requirements** | UR1,<br>SR1,<br>SR2 | **Category** | Interface Test |
| **Description** | The game has finished.<br>A player selects to watch the replay of the match.<br><br>Assert that the replay follows the events that happened during the match. | | |
| **Evidence** |  <br>In game (evidence of SR1) In replay | | |
| **Status** | Pass | | |

| | | | |
|---|---|---|---|
| **Id** | I12 | **Test Name** | InterfaceTest::testReplaySameTime |
| **Related Code** | EventReplayer:: setPlayBackSpeed<br>Used with whole of EventReplayer class. | | |
| **Related Requirements** | SR3 | **Category** | Interface Test |
| **Description** | The game has finished.<br>A player selects to watch the replay of the match.<br>The player chooses to watch the game in real time.<br>The player selects 1.0x on the speed bar.<br><br>Assert that the replay plays in the same amount of time as the game took. | | |
| **Evidence** | The game took : 1 minute 27 seconds<br>The replay took : 1 minute 27 seconds<br><br>Both were timed using the same stopwatch. | | |
| **Status** | Pass | | |

| | | | |
|---|---|---|---|
| **Id** | I13 | **Test Name** | InterfaceTest::testReplayTwoTimes |
| **Related Code** | EventReplayer:: setPlayBackSpeed<br>Used with whole of EventReplayer class. | | |
| **Related Requirements** | SR3 | **Category** | Interface Test |
| **Description** | The game has finished.<br>A player selects to watch the replay of the match.<br>The player chooses to watch the game at 2.0x on the speed bar.<br><br>Assert that the replay plays in half the amount of time that the game took. | | |
| **Evidence** | The game took : 1 minute 27 seconds<br>The replay took : 44 seconds<br><br>Both were timed using the same stopwatch. | | |
| **Status** | Pass | | |

| | | | |
|---|---|---|---|
| **Id** | I14 | **Test Name** | InterfaceTest::testReplayThreeTimes |
| **Related Code** | EventReplayer:: setPlayBackSpeed<br>Used with whole of EventReplayer class. | | |
| **Related Requirements** | SR3 | **Category** | Interface Test |
| **Description** | The game has finished.<br>A player selects to watch the replay of the match.<br>The player chooses to watch the game at 3.0x on the speed bar.<br><br>Assert that the replay plays in a third of the amount of time that the game took. | | |
| **Evidence** | The game took : 1 minute 27 seconds<br>The replay took : 29 seconds<br><br>Both were timed using the same stopwatch. | | |
| **Status** | Pass | | |

| | | | |
|---|---|---|---|
| **Id** | I15 | **Test Name** | InterfaceTest::testRouteHighlight |
| **Related Code** | GameScreen:: Render | | |
| **Related Requirements** | - | **Category** | Interface Test |
| **Description** | Assert that after hovering over a train on the screen, its route is highlighted. | | |
| **Evidence** |  | | |
| **Status** | Pass | | |

| | | | |
|---|---|---|---|
| **Id** | I16 | **Test Name** | InterfaceTest::testCurrentPlayer |
| **Related Code** | GameScreen:: Render | | |
| **Related Requirements** | SR11 | **Category** | Interface Test |
| **Description** | Assert that the current player information is displayed on the screen. | | |
| **Evidence** |  <br>Player 2's information is displayed on the left hand side of the GUI. | | |
| **Status** | Pass | | |

## Unit Tests

| | | | |
|---|---|---|---|
| **Id** | U1 | **Test Name** | ConnectionTest::testUpgradeConnectionBronze |
| **Related Code** | Connection::upgrade<br>Connection::isUpgradable<br>Connection::repair | | |
| **Related Requirements** | SR16 | **Category** | Unit Testing<br>Integration Testing |
| **Description** | A bronze connection is upgraded to silver.<br>An assertion is made to ensure that the bronze connection is now silver.<br><br>Assertions are made to ensure that a bronze connection can only be upgraded to a silver or gold connection. | | |
| **Status** | Passed. | | |

| | | | |
|---|---|---|---|
| **Id** | U2 | **Test Name** | ConnectionTest::testUpgradeConnectionSilver |
| **Related Code** | Connection::upgrade<br>Connection::isUpgradable<br>Connection::repair | | |
| **Related Requirements** | SR16 | **Category** | Unit Testing<br>Integration Testing |
| **Description** | A silver connection is upgraded to gold.<br>An assertion is made to ensure that the silver connection is now gold.<br><br>Assertions are made to ensure that a silver connection can only be upgraded to a gold connection. | | |
| **Status** | Passed. | | |

| | | | |
|---|---|---|---|
| **Id** | U3 | **Test Name** | ConnectionTest::testUpgradeConnectionGold |
| **Related Code** | Connection::upgrade<br>Connection::isUpgradable<br>Connection::repair | | |
| **Related Requirements** | SR16 | **Category** | Unit Testing<br>Integration Testing |
| **Description** | An attempt is made to “upgrade” gold connection to silver.<br>An assertion is made to ensure that the gold connection remains gold.<br><br>Assertions are made to ensure that a gold connection cannot be upgraded to a connection of any material. | | |
| **Status** | Passed. | | |

| | | | |
|---|---|---|---|
| **Id** | U4 | **Test Name** | ConnectionTest::testDamageConnectionBronze |
| **Related Code** | Connection::inflictDamage<br>Connection::calculateDamageInflicted | | |
| **Related Requirements** | SR13 | **Category** | Unit Testing<br>Integration Testing |
| **Description** | A train inflicts damage on a bronze connection.<br>An assertion is made to ensure that the current health of the connection is lower than before it was damaged. | | |
| **Status** | Passed. | | |

| | | | |
|---|---|---|---|
| **Id** | U5 | **Test Name** | ConnectionTest::testDamageConnectionSilver |
| **Related Code** | Connection::inflictDamage<br>Connection::calculateDamageInflicted | | |
| **Related Requirements** | SR13 | **Category** | Unit Testing<br>Integration Testing |
| **Description** | A train inflicts damage on a silver connection.<br>An assertion is made to ensure that the current health of the connection is lower than before it was damaged. | | |
| **Status** | Passed. | | |

| | | | |
|---|---|---|---|
| **Id** | U6 | **Test Name** | ConnectionTest::testDamageConnectionGold |
| **Related Code** | Connection::inflictDamage<br>Connection::calculateDamageInflicted | | |
| **Related Requirements** | SR13, C5 | **Category** | Unit Testing<br>Integration Testing |
| **Description** | A train inflicts damage on a gold connection.<br>An assertion is made to ensure that the current health of the connection is the same as it was before it was damaged. | | |
| **Status** | Passed. | | |

| | | | |
|---|---|---|---|
| **Id** | U7 | **Test Name** | ConnectionTest::testRepairConnection |
| **Related Code** | Connection::repair | | |
| **Related Requirements** | UR7, SR15 | **Category** | Unit Testing |
| **Description** | A train inflicts damage on a connection.<br>The connection is then repaired to full health (1).<br>An assertion is made to ensure that the current health of the connection is 1, after it has been repaired. | | |
| **Status** | Passed. | | |

| | | | |
|---|---|---|---|
| **Id** | U8 | **Test Name** | ConnectionTest::testUpgradeDamagedConnection |
| **Related Code** | Connection::upgrade<br>Connection::isUpgradable<br>Connection::repair | | |
| **Related Requirements** | SR16, SR17 | **Category** | Unit Testing<br>Integration Testing |
| **Description** | A train inflicts damage on a bronze connection.<br>The connection is then upgraded to a silver connection.<br>An assertion is made to ensure that the current health of the connection is 1, after it has been upgraded. | | |
| **Status** | Passed. | | |

| | | | |
|---|---|---|---|
| **Id** | U9 | **Test Name** | ConnectionTest::testAdjustedTrainSpeedHealthyConnection |
| **Related Code** | Connection:: calculateAdjustedTrainSpeed | | |
| **Related Requirements** | SR14 | **Category** | Unit Testing |
| **Description** | An assertion is made to ensure that the adjusted speed of a train travelling on a connection with full health is equal to its usual speed. | | |
| **Status** | Passed. | | |

| | | | |
|---|---|---|---|
| **Id** | U10 | **Test Name** | ConnectionTest::testAdjustedTrainSpeedDamagedConnection |
| **Related Code** | Connection:: calculateAdjustedTrainSpeed<br>Connection:: inflictDamage<br>Connection:: calculateDamageInflicted | | |
| **Related Requirements** | SR14 | **Category** | Unit Testing<br>Integration Testing |
| **Description** | A train inflicts damage on a connection.<br>An assertion is made to ensure that the adjusted speed of a train travelling on the damaged connection is lower than it's usual speed. | | |
| **Status** | Passed. | | |

| | | | |
|---|---|---|---|
| **Id** | U11 | **Test Name** | ConnectionTest::testGetRentPayableVariableDistance |
| **Related Code** | Connection::getRentPayable<br>Connection::calculateRentPayable | | |
| **Related Requirements** | UR6, SR12 | **Category** | Unit Testing<br>Integration Testing |
| **Description** | Two bronze connections with varying lengths are created.<br><br>An assertion is made to ensure that there is a positive amount of rent payable on the short connection.<br>An assertion is made to ensure that the amount of rent payable on the longer connection is greater than that of the amount payable of the short connection. | | |
| **Status** | Passed. | | |

| | | | |
|---|---|---|---|
| **Id** | U12 | **Test Name** | ConnectionTest::testGetRentPayableVariableMaterial |
| **Related Code** | Connection::getRentPayable<br>Connection::calculateRentPayable | | |
| **Related Requirements** | UR6, SR12 | **Category** | Unit Testing<br>Integration Testing |
| **Description** | One connection made out of each of the three material types (bronze , silver and gold) are created. Their lengths are identical.<br><br>An assertion is made to ensure that there is a positive amount of rent payable on the bronze connection.<br><br>An assertion is made to ensure that the amount of rent payable on the silver connection is greater than that of the amount payable of the bronze connection.<br><br>An assertion is made to ensure that the amount of rent payable on the gold connection is greater than that of the amount payable of the silver connection. | | |
| **Status** | Passed. | | |

| | | | |
|---|---|---|---|
| **Id** | U13 | **Test Name** | ConnectionTest::testConnectionCostVariableDistance |
| **Related Code** | Connection::calculateCost<br>Connection::calculateTotalCost | | |
| **Related Requirements** | SR18 | **Category** | Unit Testing<br>Integration Testing |
| **Description** | Two bronze connections with varying lengths are created.<br><br>An assertion is made to ensure that the cost of the short connection is positive.<br>An assertion is made to ensure that the cost of the longer connection is greater than the cost of the short connection. | | |
| **Status** | Passed. | | |

| | | | |
|---|---|---|---|
| **Id** | U14 | **Test Name** | ConnectionTest::testConnectionCostVariableMaterial |
| **Related Code** | Connection::calculateCost<br>Connection::calculateTotalCost | | |
| **Related Requirements** | SR10 | **Category** | Unit Testing<br>Integration Testing |
| **Description** | One connection made out of each of the three material types (bronze , silver and gold) are created. Their lengths are identical.<br><br>An assertion is made to ensure that the cost of the bronze connection is positive.<br><br>An assertion is made to ensure that the cost of the silver connection is greater than the cost of the bronze connection.<br><br>An assertion is made to ensure that the cost of the gold connection is greater than the cost of the silver connection. | | |
| **Status** | Passed. | | |

| | | | |
|---|---|---|---|
| **Id** | U15 | **Test Name** | ConnectionTest::testRepairCostVariableDistance |
| **Related Code** | Connectoin::calculateRepairCost<br>Connection::inflictDamage<br>Connection::calculateDamageInflicted | | |
| **Related Requirements** | UR6,<br>SR15 | **Category** | Unit Testing<br>Integration Testing |
| **Description** | Two bronze connections with varying lengths are created.<br>Damage is inflicted to both of the connections.<br><br>An assertion is made to ensure that the cost of fully repairing the short connection is positive.<br>An assertion is made to ensure that the cost of fully repairing the longer connection is greater than the cost of repairing the short connection. | | |
| **Status** | Passed. | | |

| | | | |
|---|---|---|---|
| **Id** | U16 | **Test Name** | ConnectionTest::testRepairCostVariableMaterial |
| **Related Code** | Connectoin::calculateRepairCost<br>Connection::inflictDamage<br>Connection::calculateDamageInflicted | | |
| **Related Requirements** | UR6, SR15 | **Category** | Unit Testing<br>Integration Testing |
| **Description** | One connection made out of each of the three material types (bronze , silver and gold) are created. Their lengths are identical.<br>The silver and the bronze connections are both damaged until they have 0 health. The gold connection is also damaged, but it will have no effect.<br><br>An assertion is made to ensure that the cost of fully repairing the bronze connection is positive.<br>An assertion is made to ensure that the cost of fully repairing the silver connection is greater than the cost of repairing the bronze connection.<br>An assertion is made to ensure that the cost of fully repairing the gold connection is 0. | | |
| **Status** | Passed. | | |

| | | | |
|---|---|---|---|
| **Id** | U17 | **Test Name** | ConnectionTest::testUpgradeCostVariableDistance |
| **Related Code** | Connection::calculateUpgradeCost<br>Connectoin::calculateRepairCost | | |
| **Related Requirements** | SR16 | **Category** | Unit Testing<br>Integration Testing |
| **Description** | Two bronze connections with varying lengths are created.<br><br>An assertion is made to ensure that the cost of upgrading the short connection to gold is positive.<br>An assertion is made to ensure that the cost of upgrading the longer connection to gold is greater than the cost of repairing the short connection. | | |
| **Status** | Passed. | | |

| | | | |
|---|---|---|---|
| **Id** | U18 | **Test Name** | ConnectionTest::testUpgradeCostVariableMaterial |
| **Related Code** | Connection::calculateUpgradeCost<br>Connectoin::calculateRepairCost | | |
| **Related Requirements** | SR16 | **Category** | Unit Testing<br>Integration Testing |
| **Description** | One connection made out of each of the three material types (bronze , silver and gold) are created. Their lengths are identical.<br><br>An assertion is made to ensure that the cost of upgrading the silver connection to gold is positive.<br>An assertion is made to ensure that the cost of upgrading the bronze connection to gold is greater than the cost of repairing the silver connection.<br>An assertion is made to ensure that the cost of upgrading the gold to gold connection is 0. | | |
| **Status** | Passed. | | |

| | | | |
|---|---|---|---|
| **Id** | U19 | **Test Name** | PlayerTest::testPayConnectionOwnedConnection |
| **Related Code** | Player::payConnectionRent<br>Connection::getOwner<br>Connection::getRentPayable | | |
| **Related Requirements** | UR6,<br>SR12 | **Category** | Unit Testing<br>Integration Testing |
| **Description** | Two players are created. They each have ownership of a distinct connection.<br><br>Player 1 pays connection rent for the connection that Player 1 owns.<br><br>An assertion is made to ensure Player 1 has the same amount of money they had before they paid the connection rent.<br>An assertion is made to ensure Player 2 has the same amount of money they had before the connection rent was paid. | | |
| **Status** | Passed. | | |

| | | | |
|---|---|---|---|
| **Id** | U20 | **Test Name** | PlayerTest::testPayConnectionNotOwnedConnection |
| **Related Code** | Player::payConnectionRent<br>Connection::getOwner<br>Connection::getRentPayable | | |
| **Related Requirements** | UR6, SR12 | **Category** | Unit Testing<br>Integration Testing |
| **Description** | Two players are created. They each have ownership of a distinct connection.<br><br>Player 1 pays connection rent for the connection that Player 2 owns.<br><br>An assertion is made to ensure Player 1 has less money than they had before they paid the connection rent.<br>An assertion is made to ensure Player 2 has more money than they had before the connection rent was paid.<br>An assertion is made to ensure that the money that Player 1 spent is equal to the amount of money that Player 2 gained. | | |
| **Status** | Passed. | | |

| | | | |
|---|---|---|---|
| **Id** | U21 | **Test Name** | PlayerTest::testPayConnectionFreeConnection |
| **Related Code** | Player::payConnectionRent<br>Connection::getOwner<br>Connection::getRentPayable | | |
| **Related Requirements** | UR6, C4, SR12 | **Category** | Unit Testing<br>Integration Testing |
| **Description** | Two players are created.<br><br>Player 1 pays connection rent for the connection that neither player owns. Connections that neither player owns are classed as free connections.<br><br>An assertion is made to ensure Player 1 has the same amount of money they had before they paid the connection rent.<br>An assertion is made to ensure Player 2 has the same amount of money they had before the connection rent was paid. | | |
| **Status** | Passed. | | |

| | | | |
|---|---|---|---|
| **Id** | U22 | **Test Name** | PlayerTest::testPlayerSpendMoney |
| **Related Code** | Player::spendMoney | | |
| **Related Requirements** | - | **Category** | Unit Testing |
| **Description** | A player spends a certain amount of money.<br><br>An assertion is made to ensure that the player funds have decreased by the same amount which they spent. | | |
| **Status** | Passed. | | |

| | | | |
|---|---|---|---|
| **Id** | U23 | **Test Name** | PlayerTest::testRemoveConnection |
| **Related Code** | Map::removeConnection<br>Player::removeOwnedConnection | | |
| **Related Requirements** | UR3<br>SR6, SR7 | **Category** | Unit Testing<br>Integration Testing |
| **Description** | A Player is created. This player has ownership of a connection.<br><br>A map is created, the connection owned by the player is added to the map.<br>The connection is then removed from the map.<br><br>An assertion is made to ensure that the set of connections owned by the player is empty. | | |
| **Status** | Failed. | | |

| **Id** | U24 | **Test Name** | PlayerTest::testRemoveConnection |
|---|---|---|---|
| **Related Code** | Map::removeConnection<br>Player::removeOwnedConnection | | |
| **Related Requirements** | UR3<br>SR6, SR7 | **Category** | Unit Testing<br>Integration Testing |
| **Description** | A revision of the previous “PlayerTest::testRemoveConnection” test.<br>A Player is created. This player has ownership of a connection.<br><br>A map is created, the connection owned by the player is added to the map.<br>The connection is then removed from the map.<br><br>An assertion is made to ensure that the set of connections owned by the player does not contain the removed connection. | | |
| **Status** | Passed. | | |

## Integration Tests

The integration test cases can be found above. They are identified as:
U1, U2, U3, U4, U5, U6, U8, U10, U11, U12, U13, U14, U15, U16, U17, U18, U19, U20, U21, U23, U24

An example of the test is U10:

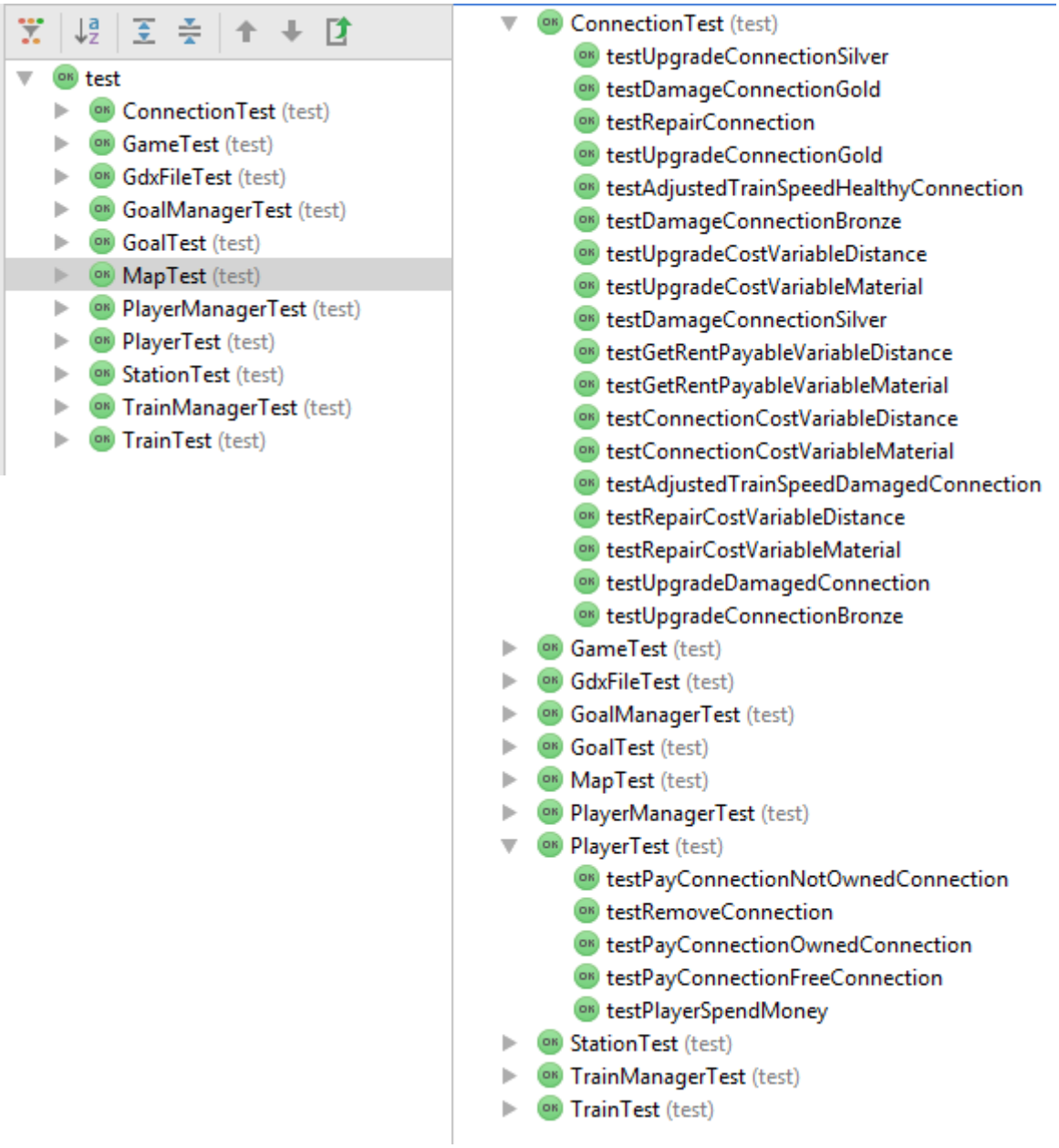
```
@Test
public void testAdjustedTrainSpeedDamagedConnection() {
    int normalSpeed = testTrain.getSpeed();

    bronzeConnection.inflictDamage(testTrain);
    int adjustedSpeed = bronzeConnection.calculateAdjustedTrainSpeed(testTrain);

    assertTrue(normalSpeed > adjustedSpeed);
}
```

The corresponding code sections are:

```
public int calculateAdjustedTrainSpeed(Train train) {
    // The speed a train will travel at, taking into account the health of the connection.

    /* We always want the train to be atleast this fast
    (as a % of it's usual speed) */
    float lowerBound = 0.2f;

    int trainSpeed = train.getSpeed();
    float variableSpeedScale = (1f - lowerBound) * health;
    return (int) ((float) trainSpeed * (lowerBound + variableSpeedScale));
}

private float calculateDamageInflicted(Train train) {
    return (1f - strength) * (train.getSpeed() / 115f) * 0.75f; //115, fastest train speed
}

public void inflictDamage(Train train) {
    float damageToInflict = material.calculateDamageInflicted(train);
    health -= damageToInflict;
    if (health <= 0) health = 0;
}
```

## Regression Tests

Should a problem be encountered upon receiving the results of the tests, the code will be rewritten appropriately and the tests reexecuted. This is in an attempt to meet the requirements laid out within the tests.
All of the unit tests, tests U1 – U24, were repeated after every change made to the code. The tests written by team XYG were also repeated at the same time as unit tests U1-U24.




The test suite provided by team XYG was also run before any changes were made to the code to ensure there were no errors.

## Performance Tests

| Name | Status | 15% CPU | 65% Memory | 38% Disk | 0% Network |
|---|---|---|---|---|---|
| idea.exe (32 bit) | | 0.2% | 509.1 MB | 0 MB/s | 0 Mbps |

Due to the recorded instance of high CPU usage for the FVS product, received for Assessment 3, it was decided that CPU usage needed to be tested. This applied more so considering team XYG, the team that provided the basis for Assessment 4, had also chosen FVS for Assessment 2 and as such could have inherited the high CPU usage error.
The game was run in both a CPU running a Windows OS and a CPU using a Linux OS. The game ran in both and as such meets the requirement regarding the OS.

## Usability Tests

Usability tests were undertaken to ensure that the user manual was accurate and appropriate in explaining how to play the TaxE game.

Due to time constraints, these testers were the team testers' flatmates. These test subjects were asked to have the user manual open, and to try to play the game against the tester (the only one with knowledge of the game).

The test subjects all indicated that the user manual was very accurate and helpful for some moments, such as figuring out how to plan a route. Some, however, indicated that they prefer to figure out how to play the game as they play and as such would be unlikely to use the user manual under normal circumstances.

Despite this, the user manual still reached its requirements in that it accurately describes how to play the game without being 'too complicated'.

These results suggest that whilst the user manual may be helpful for some people, others will not even look at the document. As such, it has been decided that the link to download the user manual from the website will be located next to the link to download the game. This is in an effort to try and encourage others to look at the user manual before playing the TaxE game.