M5Test.java

testPlacementOfSingleTowerBad

- This test ensures that, when a single “bad” tower is placed on the map at a specified location, the corresponding positions on the path the enemies take are updated correctly to deal the right amount of damage to enemies that pass over it. This is done by calling the testPlaceTower() method in the LandscapeController class and verifying that the HashMap object it returns contains the correct keys and values for the positions adjacent to where the tower is placed

testPlacementOfSingleTowerNormal

- This test ensures that, when a single “normal” tower is placed on the map at a specified location, the corresponding positions on the path the enemies take are updated correctly to deal the right amount of damage to enemies that pass over it. This is done by calling the testPlaceTower() method in the LandscapeController class and verifying that the HashMap object it returns contains the correct keys and values for the positions adjacent to where the tower is placed

testPlacementOfSingleTowerElite

- This test ensures that, when a single “elite” tower is placed on the map at a specified location, the corresponding positions on the path the enemies take are updated correctly to deal the right amount of damage to enemies that pass over it. This is done by calling the testPlaceTower() method in the LandscapeController class and verifying that the HashMap object it returns contains the correct keys and values for the positions adjacent to where the tower is placed

testPlacementOfMultipleTowersBad

- This test ensures that, when multiple “bad” towers are placed on the map at different locations, the corresponding positions on the path the enemies take are updated correctly to deal the right amount of damage to enemies that pass over it. This is done by calling the testPlaceTower() method in the LandscapeController class repeatedly and verifying that the HashMap object it returns on the final call contains the correct keys and values for the positions adjacent to where the towers are placed

testPlacementOfMultipleTowersNormal

- This test ensures that, when multiple “normal” towers are placed on the map at different locations, the corresponding positions on the path the enemies take are updated correctly to deal the right amount of damage to enemies that pass over it. This is done by calling the testPlaceTower() method in the LandscapeController class repeatedly and verifying that the HashMap object it returns on the final call contains the correct keys and values for the positions adjacent to where the towers are placed testPlacementOfMultipleTowersElite

- This test ensures that, when multiple “elite” towers are placed on the map at different locations, the corresponding positions on the path the enemies take are updated correctly to deal the right amount of damage to enemies that pass over it. This is done by calling the testPlaceTower() method in the LandscapeController class repeatedly and verifying that the HashMap object it returns on the final call contains the correct keys and values for the positions adjacent to where the towers are placed

testPlacementOfMultipleTowersMixed

- This test ensures that, when multiple towers of varying levels (“bad”, “normal”, “elite”) are placed on the map at different locations, the corresponding positions on the path the enemies take are updated correctly to deal the right amount of damage to enemies that pass over it. This is done by calling the testPlaceTower() method in the LandscapeController class repeatedly and verifying that the HashMap object it returns on the final call contains the correct keys and values for the positions adjacent to where the towers are placed

testPlacementOfSingleTowerOutOfRangeBad

- This test ensures that, when a single “bad” tower is placed on the map at a specified location which is far away from the path no updates are made to the path HashMap that is returned by the call to placeTestTower() as specified in the tests above. Essentially, this test checks that enemies crossing the path are not dealt any damage when a tower is out of range of the path

testPlacementOfSingleTowerOutOfRangeNormal

- This test ensures that, when a single “normal” tower is placed on the map at a specified location which is far away from the path no updates are made to the path HashMap that is returned by the call to placeTestTower() as specified in the tests above. Essentially, this test checks that enemies crossing the path are not dealt any damage when a tower is out of range of the path

testPlacementOfSingleTowerOutOfRangeElite

- This test ensures that, when a single “elite” tower is placed on the map at a specified location which is far away from the path no updates are made to the path HashMap that is returned by the call to placeTestTower() as specified in the tests above. Essentially, this test checks that enemies crossing the path are not dealt any damage when a tower is out of range of the path

testPlacementOfMultipleTowersOneOutOfRangeOneInRangeBad

- This test ensures that, when two towers are placed on the map at specified locations, one of which is close to the path, and one of which is out of range, only the tower that is near the path deals damage to enemies. This is done by calling the testPlaceTower() method in the LandscapeController class twice and verifying that the HashMap object it returns on the final call contains the correct keys and values for the positions adjacent to where the in-range tower is placed. This test verifies that the above is true when both towers are “bad”

testPlacementOfMultipleTowersOneOutOfRangeOneInRangeNormal

- This test ensures that, when two towers are placed on the map at specified locations, one of which is close to the path, and one of which is out of range, only the tower that is near the path deals damage to enemies. This is done by calling the testPlaceTower() method in the LandscapeController class twice and verifying that the HashMap object it returns on the final call contains the correct keys and values for the positions adjacent to where the in-range tower is placed. This test verifies that the above is true when both towers are “normal”

testPlacementOfMultipleTowersOneOutOfRangeOneInRangeElite

- This test ensures that, when two towers are placed on the map at specified locations, one of which is close to the path, and one of which is out of range, only the tower that is near the path deals damage to enemies. This is done by calling the testPlaceTower() method in the LandscapeController class twice and verifying that the HashMap object it returns on the final call contains the correct keys and values for the positions adjacent to where the in-range tower is placed. This test verifies that the above is true when both towers are “elite”

testPlacementOfZeroTowers

- This test ensures that, when a single tower is placed in an invalid position (i.e. no tower is placed on the map), enemies on the path are not damaged, as expected. This is done by calling the testPlaceTower() method in the LandscapeController class and verifying that the HashMap object it returns contains no keys and values (i.e. its size() method returns 0)