Testing Report

Railyard class

CycleSortMethod (Generic Array)

First test for this method: testCycleSortArrayMethod()

This test method performs test 0, 1 & Many and Test First, Middle & Last

Description

To test this method, basically 6 tests are performed with sub-tests. For each method, an array of Classification Yard is created. Then the generic array to be sorted is declared and initialized. Now, the expected sorted array is declared and initialized. Finally, the object of Railyard is instantiated and the cycleSort method is called and the array to be sorted is passed as a parameter. Now, compare the modified array with the expected array.

Test 0

- First test: 0 classification Yards
- Second test: Multiple classification yards with 0 tracks
- Third test: Train with 0 cars

Test 1

- First test: 1 classification Yards
- Second test: Multiple classification yards with 1 tracks
- Third test: Train with 1 cars

Test Many

- Many classification yards with Many tracks and many cars in the train to be sorted

Test First

- First Test: The train to be sorted is mostly sorted except the first element is at the wrong position
- Second Test: Testing the sorting of train just after the first classification yard

Test Last

- First Test: The train to be sorted is mostly sorted except the Last element is at the wrong position
- Second Test: Testing the sorting of train just after the Last classification yard

Test Middle

- First Test: The train to be sorted is mostly sorted except the Middle element is at the wrong position
- Second Test: Testing the sorting of train just after the Middle classification yard

Second test

TestCycleSortArrayConditions()

This testMethod tests all the possible if – else conditions present in the “cycleSort” method.

Description

The first element 20 is added on the first track of the classification yard. Then the second element 11 undergoes the first if condition. The first if condition gets false. The else condition is executed. In the else condition, the first if condition is checked. This condition gets true as empty track is available. So, 11 is added on the second track. Now the loop iterates and 6 is compared with 11. The first bigger if gets false. The bigger else is executed. The if inside the else gets false as no other empty track is available. So, the else inside the bigger else is executed. Now the loop again iterates. The value 10 is compared with 6. the first bigger if condition gets true. So, 10 is added after 6.

ClosestSortMethod (Generic Array)

First test for this method: testClosestSortListMethod()

This test method performs test 0, 1 & Many and Test First, Middle & Last

Description

To test this method, basically 6 tests are performed with sub-tests. For each method, an array of Classification Yard is created. Then the generic array to be sorted is declared and initialized. Now, the expected sorted array is declared and initialized. Finally, the object of Railyard is instantiated and the closestSort method is called and the array to be sorted is passed as a parameter. Now, compare the modified array with the expected array.

Test 0

- First test: 0 classification Yards
- Second test: Multiple classification yards with 0 tracks
- Third test: Train with 0 cars

Test 1

- First test: 1 classification Yards
- Second test: Multiple classification yards with 1 tracks
- Third test: Train with 1 cars

Test Many

- Many classification yards with Many tracks and many cars in the train to be sorted

Test First

- First Test: The train to be sorted is mostly sorted except the first element is at the wrong position
- Second Test: Testing the sorting of train just after the first classification yard

Test Last

- First Test: The train to be sorted is mostly sorted except the Last element is at the wrong position
- Second Test: Testing the sorting of train just after the Last classification yard

Test Middle

- First Test: The train to be sorted is mostly sorted except the Middle element is at the wrong position
- Second Test: Testing the sorting of train just after the Middle classification yard

Second test

TestClosestSortArrayConditions()

This testMethod tests all the possible if – else conditions present in the "closestSort" method.

Description

The first element 20 is added on the first track of the classification yard. Then we enter the second loop which starts from the second car of the train. The while loop statement gets true. The if statement inside the while loop gets false because 11 is not greater than 20. K is incremented. The while statement is checked again(gets true). The if statement inside the while loop gets false because the next track is empty. The k is incremented. Again the while statement is checked and the if statement gets false because of empty track. Since the flag is 'n', the if statement after while loop gets false. The following else statement is executed. The loop looks for an empty track (it gets the track and flag 2 is set to 'y'). The following if condition is gets true and the 11 is finally added on new track. Now the loop on train iterates. The first while loop statement is checked and its if statement gets true. So, 16 is added after 11.

Now, the loop of train again iterates. The while loop statement is checked and its if statement is checked which gets false for 20. The while loop is iterated and again the if statement of loop gets false. The while loop is iterated and again the if statement of loop gets false. The following if statement outside the while loop gets false as flag is set to 'n'. The next else statement is executed. The while loop inside finds an empty track (if condition gets true) and 3 is added on a new track. Similarly 5 gets added after 3 (just like 16 was added after 11). Now the

last element 0 goes through while loop and the if statement is false every time. The following if statement also gets false. Following else statement is executed. The while loop of the else statement on each iteration gives out false if statement. So, finally the if statement after while loop gets false. Finally, the else statement is executed and 0 is added after 5.

CycleSortMethod (Generic List)

First test for this method: testCycleSortListMethod()

This test method performs test 0, 1 & Many and Test First, Middle & Last

Description

To test this method, basically 6 tests are performed with sub-tests. For each method, an array of Classification Yard is created. Then the generic List to be sorted is declared and initialized. Now, the expected sorted List is declared and initialized. Finally, the object of Railyard is instantiated and the cycleSort method is called and the List to be sorted is passed as a parameter. Now, compare the modified array with the expected array.

Test 0

- First test: 0 classification Yards
- Second test: Multiple classification yards with 0 tracks
- Third test: Train with 0 cars

Test 1

- First test: 1 classification Yards
- Second test: Multiple classification yards with 1 tracks
- Third test: Train with 1 cars

Test Many

- Many classification yards with Many tracks and many cars in the train to be sorted

Test First

- First Test: The train to be sorted is mostly sorted except the first element is at the wrong position
- Second Test: Testing the sorting of train just after the first classification yard

Test Last

- First Test: The train to be sorted is mostly sorted except the Last element is at the wrong position
- Second Test: Testing the sorting of train just after the Last classification yard

Test Middle

- First Test: The train to be sorted is mostly sorted except the Middle element is at the wrong position
- Second Test: Testing the sorting of train just after the Middle classification yard

Second test

TestCycleSortListConditions()

This testMethod tests all the possible if – else conditions present in the "cycletSort" method.

Description

(Similar description as CycleSortCondition Array Method)

The first element 20 is added on the first track of the classification yard. Now the while loop condition is checked which gets true. Then the second element 11 undergoes the first condition. Inside the while loop, the first if condition gets false. The else condition is executed. The first if condition gets true and 11 is added on the second track. Now the while loop condition is checked, iterates and 16 is compared with 11. The if gets true. 16 is added after 11. Now the loop again iterates. The value 3 is compared with 16. The if condition gets false. Finally, the else statement of the following else statement is executed. Now, while loop iterates again. 10 is compared with 3 and if statement of loop gets true. So, 10 is added after 3.

ClosestSortMethod (Generic List)

First test for this method: testClosestSortListMethod()

This test method performs test 0, 1 & Many and Test First, Middle & Last

Description

To test this method, basically 6 tests are performed with sub-tests. For each method, an array of Classification Yard is created. Then the generic List to be sorted is declared and initialized. Now, the expected sorted List is declared and initialized. Finally, the object of Railyard is instantiated and the closestSort method is called and the List to be sorted is passed as a parameter. Now, compare the modified array with the expected array.

Test 0

- First test: 0 classification Yards
- Second test: Multiple classification yards with 0 tracks
- Third test: Train with 0 cars

Test 1

- First test: 1 classification Yards
- Second test: Multiple classification yards with 1 tracks
- Third test: Train with 1 cars

Test Many

- Many classification yards with Many tracks and many cars in the train to be sorted

Test First

- First Test: The train to be sorted is mostly sorted except the first element is at the wrong position
- Second Test: Testing the sorting of train just after the first classification yard

Test Last

- First Test: The train to be sorted is mostly sorted except the Last element is at the wrong position
- Second Test: Testing the sorting of train just after the Last classification yard

Test Middle

- First Test: The train to be sorted is mostly sorted except the Middle element is at the wrong position
- Second Test: Testing the sorting of train just after the Middle classification yard

Second test

TestClosestSortListConditions()

This testMethod tests all the possible if – else conditions present in the "closestSort" method.

Description

(Similar to ClosestArrayConditionsMethod)

The first element 20 is added on the first track of the classification yard. Then the second element 11 undergoes the first if condition. The first if condition gets false. The else condition is executed. In the else condition, the first if condition is checked. This condition gets true as empty track is available. So, 11 is added on the second track. Now the loop iterates and 6 is compared with 11. The first bigger if gets false. The bigger else is executed. The if inside the else gets false as no other empty track is available. So, the else inside the bigger else is executed. Now the loop again iterates. The value 10 is compared with 6. the first bigger if condition gets true. So, 10 is added after 6.

testGetHeadsUp Method

This test method tests the helper method: getHeadsUp which is used to check for empty track and report the position of non-empty tracks

First Test: This test is performed when the classification yard is non-empty

Description:

A new classification yard array is declared and initialized. Instance of Railyard is created and the array of classification yard is passed into it as parameter. The constructor of the class creates the ArrayList of Classification Yard.

Now, a new ArrayList of classification Yard is created. It is initialized with second classification yard having second track a car. Now the ArrayList of Classification Yard in the instance of Railyard is modified using the setter method. Finally the getHeadsUp method is called and the expected return value is 1.

Second Test: This test is performed when the classification yard is empty

Description:

A new classification yard array is declared and initialized. Instance of Railyard is created and the array of classification yard is passed into it as parameter. The constructor of the class creates the ArrayList of Classification Yard.

Now, a new ArrayList of classification Yard is created. It is initialized with classification yard with no cars. Now the ArrayList of Classification

Yard in the instance of Railyard is modified using the setter method. Finally the getHeadsUp method is called and the expected return value is -1.

TestMergeArrayMethod

This method tests the Merge method which merges a generic array.

Description:

Created an integer array to be sorted. Created an expected sorted array. Then the array of classification yard is created and it is passed into the instance of Railyard class as parameter for the constructor. The constructor of the class creates the ArrayList of classificationYard. Now, a new ArrayList of classificationYard is declared and initialized. The first classification yard is initialized with two tracks and cars are put up on the tracks such that they are sorted version of the original integer array. Finally, the ArrayList of classificationYard of the instance is changed with the created one using the setter method. Now, the merge method is called and the output array is compared with the expected array we created earlier.

TestMergeListMethod

This method tests the Merge method which merges a generic List.

Description:

The array of classification yard is created and it is passed into the instance of Railyard class as parameter for the constructor. The constructor of the class creates the ArrayList of classificationYard. Now, a new ArrayList of classificationYard is declared and initialized. The first classification yard is initialized with two tracks and cars are put up on the tracks such that they are sorted version of the original integer array. Finally, the ArrayList of classificationYard of the instance is changed with the created one using the setter method. Also, the list of integers to be sorted and expected sorted list are initialized and declared. Now, the merge method is called and the return list is compared with the expected list we created.

AdditionalTestCase1

This test method tests a set of numbers which are sorted from closestSort method and not from cycleSort

Description:

The array of classification yard is created and it is passed into the instance of Railyard class as parameter for the constructor. The constructor of the class creates the ArrayList of classificationYard. Now, the integer array/ List of those numbers are created along with the expected sorted array from closest sorts. Finally, the closest sort method is called which gives the expected “perfectly” sorted array unlike from cycleSort method.

AdditionalTestCase2(not found)

This test method tests a set of numbers which are sorted from cleSort method and not from closestSort

Description:

The array of classification yard is created and it is passed into the instance of Railyard class as parameter for the constructor. The constructor of the class creates the ArrayList of classificationYard. Now, the integer array/ List of those numbers are created along with the expected sorted array from closest sorts. Finally, the cycle sort method is called which gives the expected “perfectly” sorted array unlike from closest Sort method.

testGetterSetterMethod()

This method tests the getters/ setter method of field of class

Description:

The array of classification yard is created and it is passed into the instance of Railyard class as parameter for the constructor. The constructor of the class creates the ArrayList of classificationYard. Now, a new ArrayList of classificationYard is declared and initialized. Finally, the ArrayList of instance is changed using the setter method. And getter method is called to check the expected modified ArrayList.

testConstructor()

This test method Tests the constructor of the class

Description:

The array of classification yard is created and it is passed into the instance of Railyard class as parameter for the constructor. The constructor of the class creates the ArrayList of classificationYard. Now, a new ArrayList of classificationYard is declared and initialized. Finally, the ArrayList of instance is changed using the setter method. By using the getter method, we compare the the reuturned address with address of the newly created ArrayList. If the comparison gives true, the constructor worked perfectly. Only the well working of constructor, allowed the changing of the field.