import core.Line;

import core.Station;

import org.junit.Before;

import org.junit.Test;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

import static junit.framework.TestCase.assertEquals;

public class RouteCalculatorTest {

private StationIndex testStationIndex;

private RouteCalculator testRouteCalculate;

@Before

public void setUp() throws Exception {

testStationIndex = new StationIndex();

Line line1 = new Line(1, "Green");

Line line2 = new Line(2, "Orange");

Line line3 = new Line(3, "Red");

testStationIndex.addLine(line1);

testStationIndex.addLine(line2);

testStationIndex.addLine(line3);

//S-Station, SC - station connect (Green-Orange/Orange-red)

List<Station> testsStations = Arrays.asList(

new Station("GreenOne\_S", line1),

new Station("GreenTwo\_SC\_GO", line1),

new Station("GreenThree\_S", line1),

new Station("OrangeOne\_SC\_GO", line2),

new Station("OrangeOne\_S", line2),

new Station("OrangeOne\_SC\_OR", line2),

new Station("RedOne\_S", line3),

new Station("RedTwo\_S", line3),

new Station("RedThree\_SC\_OR", line3)

);

List<Station> testsStationsConnectionOne = new ArrayList<>();

List<Station> testsStationsConnectionSecond = new ArrayList<>();

testsStations.forEach(itemStation -> {

testStationIndex.addStation(itemStation);

itemStation.getLine().addStation(itemStation);

if (itemStation.getName().contains("\_SC\_GO")) {

testsStationsConnectionOne.add(itemStation);

} else if (itemStation.getName().contains("\_SC\_OR")) {

testsStationsConnectionSecond.add(itemStation);

}

});

testStationIndex.addConnection(testsStationsConnectionOne);

testStationIndex.addConnection(testsStationsConnectionSecond);

testRouteCalculate = new RouteCalculator(testStationIndex);

}

@Test

public void testGetShortestRouteOneLine() {

//Here we test internal method (getRouteOnTheLine) from GetShortestRoute (use 1 line)

Line line1 = testStationIndex.getLine(1);

Station fromStation = testStationIndex.getStation("GreenOne\_S");

Station toStation = testStationIndex.getStation("GreenThree\_S");

List<Station> actual = testRouteCalculate.getShortestRoute(fromStation, toStation);

List<Station> expected = Arrays.asList(

new Station("GreenOne\_S", line1),

new Station("GreenTwo\_SC\_GO", line1),

new Station("GreenThree\_S", line1)

);

assertEquals("One line (0 connections): ", expected, actual);

}

@Test

public void testGetShortestRouteTwoLines() {

//Here we test internal method (getRouteWithOneConnections) from GetShortestRoute (use 2 lines)

Line line1 = testStationIndex.getLine(1);

Line line2 = testStationIndex.getLine(2);

Station fromStation = testStationIndex.getStation("GreenOne\_S");

Station toStation = testStationIndex.getStation("OrangeOne\_S");

List<Station> actual = testRouteCalculate.getShortestRoute(fromStation, toStation);

List<Station> expected = Arrays.asList(

new Station("GreenOne\_S", line1),

new Station("GreenTwo\_SC\_GO", line1),

new Station("OrangeOne\_SC\_GO", line2),

new Station("OrangeOne\_S", line2)

);

assertEquals("Two Lines (1 connections): ", expected, actual);

}

@Test

public void testGetShortestRouteThreeLines() {

//Here we test internal method (getRouteWithTwoConnections) from tGetShortestRoute (use 3 lines)

Line line1 = testStationIndex.getLine(1);

Line line2 = testStationIndex.getLine(2);

Line line3 = testStationIndex.getLine(3);

Station fromStation = testStationIndex.getStation("GreenOne\_S");

Station toStation = testStationIndex.getStation("RedOne\_S");

List<Station> actual = testRouteCalculate.getShortestRoute(fromStation, toStation);

List<Station> expected = Arrays.asList(

new Station("GreenOne\_S", line1),

new Station("GreenTwo\_SC\_GO", line1),

new Station("OrangeOne\_SC\_GO", line2),

new Station("OrangeOne\_S", line2),

new Station("OrangeOne\_SC\_OR", line2),

new Station("RedThree\_SC\_OR", line3),

new Station("RedTwo\_S", line3),

new Station("RedOne\_S", line3)

);

assertEquals("Three lines (2 connections): ", expected, actual);

}

@Test

public void testCalculateDuration() {

Station fromStation = testStationIndex.getStation("GreenOne\_S");

Station toStation = testStationIndex.getStation("OrangeOne\_S");

double actual = RouteCalculator.calculateDuration(testRouteCalculate.getShortestRoute(fromStation, toStation));

//double actual = RouteCalculator.calculateDuration( );

double expected = 8.5d;

assertEquals(expected, actual);

}

}