Developer test #2 Onebox

Trains Exercise

Problem: The local commuter railroad services a number of towns in Kiwiland. Because of monetary concerns, all of the tracks are 'one-way.' That is, a route from Kaitaia to Invercargill does not imply the existence of a route from Invercargill to Kaitaia. In fact, even if both of these routes do
happen to exist, they are distinct and are not necessarily the same distance!

The purpose of this problem is to help the railroad provide its customers with information about the routes. In particular, you will compute the distance along a certain route, the number of different routes between two towns, and the shortest route between two towns.

Input: A directed graph where a node represents a town and an edge represents a route between two towns. The weighting of the edge represents the distance between the two towns. A given route will never appear more than once, and for a given route, the starting and ending
town will not be the same town.

Output: For test input 1 through 5, if no such route exists, output 'NO SUCH ROUTE'. Please use the most direct route and do not make any
extra stops! For example, the first problem means to start at city A, then travel directly to city B (a distance of 5), then directly to city C (a distance
of 4).

1. The distance of the route A-B-C.
2. The distance of the route A-D.
3. The distance of the route A-D-C.
4. The distance of the route A-E-B-C-D.
5. The distance of the route A-E-D.

Test Input:

For the test input, the towns are named using the first few letters of the alphabet from A to D. A route between two towns (A to B) with a
distance of 5 is represented as AB5.

Graph: AB5, BC4, CD8, DC8, DE6, AD5, CE2, EB3, AE7
Expected Output:
Output #1: 9
Output #2: 5
Output #3: 13
Output #4: 22
Output #5: NO SUCH ROUTE

Developer test #2 Onebox

Your example **must** include the following:

- The exercise **must** be resolved using object oriented programming.
- Unit tests
- Argument validation where applicable
- Error handling where applicable
- Design Patterns where applicable
- Comments as deemed necessary
- Documentation on how to run and test the code
- Any assumptions that you have made
- Deliver a **runnable jar** (and the instructions on how to execute it as already commented)

We should be able to run your program as delivered (i.e. without modifying any code). Please submit your code:

- Github url
- Bitbucket url
- Other version control url
- zip

If any third-party dependencies are required, you should either

(a) include the dependencies with the delivered solution, or,
(b) include a build solution in either Ant or Maven that will allow us to resolve all required dependencies.