### Instructions:

The purpose of this assignment is to allow you to demonstrate your understanding of OO principles and techniques (think re-usability, maintainability, flexibility etc). Creativity and elegance is also valued. You should not focus on producing performance-optimized code or ‘as-little-code-as-possible’ to the detriment of readability and maintainability.

The assignment is designed to be completed within 2 to 4 hours. You may however choose to expend more time, at your own discretion. You are encouraged to add documentation (think todo’s and fixme’s) to indicate portions of the design or code that you would do differently given more time.

Most importantly, your code must work. Code that does not compile or produce output will not be evaluated.

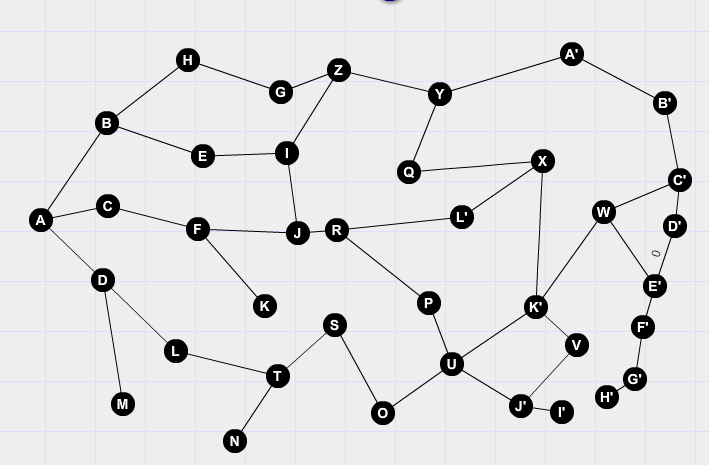
## 

### Instructions

### Mandatory Criteria

* Java

### Scenario



### Your task

The above graph represents an interstellar transport system used by Earth’s inhabitants in the year 2145. They require you to build a system that will allow them to find the shortest path from point “A”, being Earth, through the galaxy to any of the planets represented by the other nodes. You are provided with a list of node names and their respective distances between their linked planets.

The solution has been implemented in Firebase, Angular and TypeScript. The link to the GitHub respository is:

https://github.com/kaylemaclou/interstellar-transport-system

1. **Persist the Graph into an in-memory database**

Take the data set provided and create the table structure(s) necessary to hold the data.

* + Use a Derby DB
  + Use a suitable ORM tool (JPA or Hibernate)
  + Generate the bindings

-----------------------

The data for Planets, Routes and Traffic Delays are stored in documents within the Firestore NoSql database.

1. **Read the file and import it into the DB**

-----------------------

When the CSV files containing data for Planets, Routes and Traffic Delays are uploaded to Firebase bucket storage, they trigger an event that imports them into their corresponding documents within the Firestore NoSql database

1. **Expose the database using a RESTful Webservice**

Generate a restful service and expose CRUD operations to manage the database.

-----------------------

Firestore provides implicit REST endpoints for CRUD operations on documents (collections) stored within the database. For example, the Angular front-end application calls the default HTTP GET endpoint on the “Planets” collection and then uses the data to populate dropdown boxes on the user-interface.

For more information on the default Firestore endpoints, go to the following link:

https://firebase.google.com/docs/firestore/use-rest-api

1. **Implement the algorithm**

Construct a suitable object model to describe the routes. Create an algorithm that will determine the shortest path between Earth and any destination provided.

-----------------------

The algorithm and data model are implemented within a Firebase Functions. The shortest path is obtained using a recursive implementation of Dijkstra’s algorithm.

1. **Expose the algorithm using a Document Literal Web service**

Expose a document literal service:

* Construct an XSD to constrain usage of the service.
* Expose the Route Request document and return a Route Response document with the hops

-----------------------

An endpoint proving the shortest path between two planets, has been exposed from within a Firebase Functions, using an HTTP GET REST endpoint. Here is the URI of the endpoint:

1. **Create a front end to capture the source and destination and then print the shortest path**

Create a simple front end where a user can capture the source and destination details and then, upon submission, collect the results and display them in tabular format.

Bonus: Design and implement a cascading style sheet to make the UI more attractive.

-----------------------

The front-end is implemented in Angular and has been uploaded to Firebase hosting, at the following link:

https://interstellar-transport-system.web.app

1. **Optional: Overlay the following data and re-compute**

The spreadsheet contains a tab called ‘traffic’. Extend your model to encapsulate this data so that you can overlay this on the route information. Update the request interface to include an option to calculate routes with or without traffic. Update your routing algorithm to accommodate the changes.

|  |  |
| --- | --- |
| Supporting Data |  |
| **Light Year Constant** | **9,460,730,472,580,800 meters** |
| **Passenger Travel Speed** | **7,500,000,000,000 meters/second** |
|  |  |

### Overall Design

### Considerations:

1. Reusability
2. Robustness
3. Performance
4. Readability
5. Separation of concerns

### Deliverables

### Package and Submission

Your project submission must consist of these components:

* An executable JAR file or deployable WAR file as per the J2EE specification
* A design.txt document that contains any assumptions or core design considerations that will position the assignment submission. Feel free to add suggestions on how the system could be improved in subsequence releases.

### Minimum Submission Criteria

* Your program must compile, run and return a result.

### Marking Criteria

Your project submission will be evaluated based on the following:

* Does your program work?
* Comment style and formatting
* Code style, formatting and adherence to standards and conventions
* Organisation and layout of classes
* Design choices and use of OO principles and methodologies

### Choice of technology frameworks/technology Choices

The following technologies are recommended to assist you in completing the assignment. They are not mandatory and it is subject to your discretionary use, if at all.

* Spring
* Hibernate
* Maven
* Web based technology

### Additional Information

NOTE: The requirements are purposely open-ended.Please feel free to make assumptions with the proviso that you can justify your decisions.