

Contexte et objectif :

Les exploitants fournissent leurs données sous des formats différents (des exemples de fichiers de données pour chacun d'eux sont à disposition sur Moodle) :

-
- The graph shows a network of 10 locations: Limo, Arly, Singha, Neuville, Syen, Gare, Avlon, Piscine, Ecole, and Parc. The edges represent different modes of transport with associated weights (distances or costs). The legend indicates four types of transport: train (black solid line), car inter-cités (green solid line), métro (red solid line), and tram (orange dashed line). The graph is directed, with arrows indicating the direction of travel.
- Legend:
- train (black solid line)
 - car inter-cités (green solid line)
 - métro (red solid line)
 - tram (orange dashed line)
- Graph Data (Edges and Weights):
- Limo to Arly: train, weight 15
 - Arly to Singha: train, weight 20
 - Singha to Neuville: car inter-cités, weight 15
 - Neuville to Avlon: car inter-cités, weight 17
 - Singha to Syen: car inter-cités, weight 30
 - Singha to Gare: car inter-cités, weight 10
 - Syen to Gare: car inter-cités, weight 17
 - Gare to Avlon: métro, weight 15
 - Gare to Piscine: métro, weight 17
 - Piscine to Ecole: tram, weight 11
 - Ecole to Mairie: tram, weight 3
 - Mairie to Gare: tram, weight 13
 - Gare to Parc: métro, weight 10
 - Parc to Ecole: tram, weight 7
 - Ecole to Syen: tram, weight 11

1. comprendre quelles sont les données à représenter : voir les exemples de fichiers mis à disposition ;
2. créer en Java les classes nécessaires pour stocker et manipuler ces données ;
3. certains mots dans ces fichiers suivent un format spécifique. Décrire ces formats en utilisant vos connaissances (expressions régulières par exemple). Voir comment cela est représenté dans l'API "regex" ([lien dans les ressources ci-dessous](#)) ;
4. décrire les étapes de lecture des fichiers par des automates (au moins un automate par type de fichier) ;
5. sur la base de ces automates et expressions, développer les fonctionnalités permettant de lire les fichiers de données transport. La lecture permet de stocker les données par la création d'objets instances des classes prévues au point 2.

6. fournir un programme de recherche d'itinéraire, par le plus court chemin dans le réseau, selon les stations de départ et d'arrivée, et l'heure de départ au plus tôt : "je souhaite partir de X pour aller à Y après Z h".

1. un projet Java répondant au problème (.zip)
2. un compte-rendu (10 pages maxi, avec copies-écrans) expliquant les fonctionnalités de votre programme et les techniques utilisées (paquetages, méthodes).

- les classes Java "File" et "Scanner" pour la lecture de fichiers texte ;
- l'API "regex" pour la vérification de format de certains mots ;
- il existe de nombreux "parsers" ou "readers" pour XML. Voir par exemple : <https://www.oracle.com/java/technologies/jaxp-introduction.html>