# Projet - *Aventuriers du Graphe*

### IUT Montpellier-Sète – Département Informatique

Ce projet est la suite de la Phase 1 du projet *Aventuriers du Rail (Europe)*.

Dans cette *Phase 2* de la SAÉ, en parallèle de la Phase 3, nous vous proposons de programmer des algorithmes de graphes qui devraient vous permettre d'évaluer le score de chaque joueur à la fin de la partie (voir les règles). Notamment, il faudra être capable à partir d'un graphe de savoir si deux sommets sont connectés par une chaîne (cela permettra de savoir si une mission a été réalisée) et de trouver un parcours de longueur maximum qui ne repasse jamais deux fois sur la même arête. Afin d'atteindre ce but, on vous propose de suivre les étapes suivantes :

- L'objet `Graphe` qui modélise la situation finale d'un joueur, c'est-à-dire toutes les villes du jeu et ses propres routes, a pour attribut une matrice d'adjacence dans laquelle les coefficients non nuls sont égaux à la longueur de l'arête correspondante. Implicitement, les sommets de ce graphe sont les entiers de `0` à `n-1`, `n` étant l'ordre du graphe.

- Des méthodes élémentaires vous sont données dans la classe `Graphe` : calcul de l'ordre, calcul de la taille, degré d'un sommet, suppression ou création d'une arête, détermination du voisinage d'un sommet.

- En utilisant éventuellement ces méthodes, on vous demande de créer d'autres méthodes pour déterminer :

    - la classe de connexité d'un sommet ;
    - la liste des classes de connexité ;
    - si deux sommets sont voisins ;
    - si une arête est un isthme ;
    - si un graphe est un arbre, une forêt ;
    - s'il existe dans une composante un parcours eulérien ouvert ou fermé ;
    - le plus long parcours dans chacune des composantes puis dans le graphe, qui ne repasse pas deux fois par la même arête.

- Tester ces méthodes sur les graphes particuliers suivants :

    - un cycle d'ordre 10;
    - un graphe complet d'ordre 21;
    - un graphe complet d'ordre 10;
    - un arbre d'ordre 10 ayant au moins 4 feuilles;
    - un graphe non connexe avec 4 sommets isolés et 2 composantes d'ordre minimum 3;
    - un graphe eulérien d'ordre 10;
    - un graphe d'ordre minimum 10 qui n'est ni un graphe eulérien, ni un arbre.

- Pour utiliser les résultats de vos algorithmes avec le jeu principal, vous compléterez la fonction `calculerLesScores()` qui se chargera de construire l'objet `Graphe` à partir des informations du jeu (routes et villes, gares...)

**Pour des questions :**

- Le forum Piazza - à privilégier lorsque vous avez des questions sur le projet.
- Email pour une question d'ordre privée concernant le projet.

# Architecture du code

Le code qui vous est fourni contient la correction du projet correspondant à la Phase 1 du projet *Aventuriers du Rail (Europe)*. La mécanique du jeu est restée intacte dans le paquetage `fr.umontpellier.iut.rails`. La seule chose qui a été changée, c'est l'ajout de la méthode `calculerLesScores()` dans la classe `Jeu` qui utilisera la méthode `setScore(int n)` de la classe `Joueur`. Elles sont à implémenter.

Les autres méthodes à implémenter, comme d'habitude, sont celles qui lèvent des exceptions de type `RuntimeException` avec le message `"Méthode non implémentée !"`. La plupart, se trouvent toutes dans la classe `Graphe` du paquetage `fr.umontpellier.iut.graphes`. Dans cette classe, les méthodes suivantes vous sont fournies :

- Le constructeur `Graphe(int n)` qui crée un graphe vide à `n` sommets
- `nbSommets()` qui retourne le nombre de sommets du graphe
- `supprimerArete(int i, int j)` qui retire l'arête entre les sommets `i` et `j`
- `ajouterArete(int i, int j, int k)` qui ajoute une arête de longueur `k` entre les sommets `i` et `j`
- `toString()` qui met la matrice d'adjacence sous forme de chaîne de caractère et peut vous aider à debugger votre programme

# Calendrier de travail

- **Phase 2 :** ***mise en œuvre des algorithmes de graphes afin de calculer les scores et définir des stratégies de jeu pour les joueurs***
  - **Période (prévisionnelle) :** mai-début juin 2022
  - **Cours concerné** : *Graphes*
  - **Enseignants :** Alexandre Bazin, Irène Larramendy, Alain Marie-Jeanne

# Évaluation

L'évaluation du projet se fera à l'aide de tests unitaires automatisés. Un premier jeu de tests vous est fourni (comme d'habitude dans le répertoire `src/test/java`) pour que vous puissiez vérifier le bon fonctionnement des fonctionnalités de base. Puis nous utiliserons un second jeu de tests (secret) pour l'évaluation finale.

Il est donc attendu que les projets rendus passent le premier jeu de tests sans erreurs, mais vous devez également vérifier par vous-mêmes (en écrivant d'autres tests unitaires) que le projet se comporte correctement dans les différents cas particuliers qui peuvent se produire et qui ne sont pas nécessairement couverts par les tests qui vous ont été fournis.

**Remarque importante** : puisque l'évaluation des rendus se fait par des tests automatisés, **les projets qui ne compilent pas ou qui ne respectent pas les signatures données, seront automatiquement rejetés** et la note sera 0.

# Remarque

Comme indiqué précédemment cette Phase se déroule en parallèle en parallèle de la Phase 3 où vous devez programmer une interface graphique sous JavaFX. Il est important que les projets GitLab des Phases 2 et 3 restent distincts. Lorsque vos fonctions de calculs de score des joueurs en utilisant les graphes marcheront, vous pourrez (si vous le souhaitez), ajouter le code correspondant dans votre projet JavaFX (de la Phase 3). **Nous vous déconseillons fortement de le faire avant la fin !**