

# Description

## Objectifs :

Implémenter une carte permettant de réaliser diverses actions via un GUI.

## Fonctions précédemment demandées concernées :

Libellé	Description
Contact 2	Une classe PathStep qui stocke une étape d'un itinéraire : type (transport en commun local, train, avion, taxi), lieu de départ, lieu d'arrivée.
Resources1	Inclure dans votre class path un (petit !) fichier de carte du monde de Mapsforge. Créer une méthode main qui lit le fichier et teste sa validité.
Geo	Utiliser <a href="https://github.com/AReallyGoodName/OfflineReverseGeocode">https://github.com/AReallyGoodName/OfflineReverseGeocode</a> pour stocker les villes.
LocalMap	Utiliser Mapsforge pour afficher des cartes de trajet localement. (Tester avec cartes Île-de-France et Monde.) Le logiciel télécharge la carte si elle n'existe pas déjà. Il doit être possible d'afficher une carte contenant un point de départ (considéré comme Paris par défaut) et un point d'arrivée donnés
SmallMap	Obtenir une carte Mapsforge avec uniquement niveau de zoom faible, pour réduire la taille. Documenter la procédure à suivre au format AsciiDoc, en anglais. Intégrer la carte dans le logiciel.
Autre idée	Moyen aller : a → b (bus) → c (avion) etc. Pour chaque transport, indiquer : fourchette prix, à rembourser ?, à acheter ?, original stocké ? \+ possibilité stocker fichier (ou donner URL ?). Moyen retour, même chose.

## Package concernées et classes concernées :

- geocode
  - ReverseGeoCode
  - GeoName
- jconfs.map
  - Download
  - PathStep
  - setCoordinates
  - TransportType
- jconfs.gui
  - GuiConference
  - MapGUI

## Problèmes rencontrés :

L'API Mapsforge demandée dans le sujet est très peu documentée en ce qui concerne l'affichage et l'optimisation d'un trajet. Le seul moyen d'en apprendre plus sur l'API, est de se plonger dans la javadoc des classes du projet github. Cependant l'API est beaucoup plus

orientée Android (comme d'ailleurs la plupart des API travaillant avec une carte). De plus, la javadoc présentes pour documenter les méthodes n'est pas très explicite. Le temps accordé sur le projet étant assez court, nous vous recommandons d'utiliser une autre API mieux documentée.

Compétences java / eclipse nécessaires :

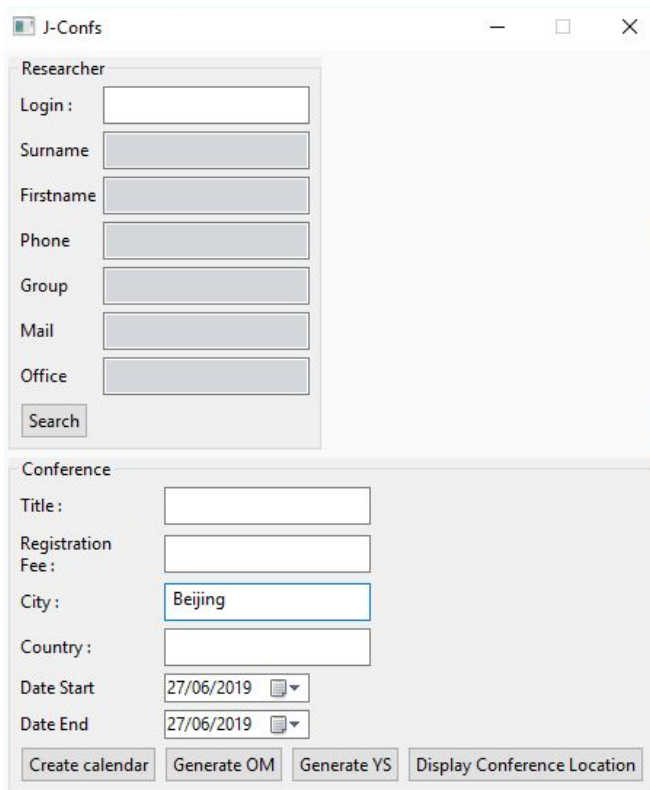
- Maîtrise des interfaces graphiques
- Maîtrise des exceptions
- Maîtrise des accès à une ressource (lecture et écriture)
- Maîtrise du classpath

Recommandation :

Cette tâche étant difficile, nous vous recommandons de définir les sous-groupes qui travailleront dessus dès la fin de la première itération.

# Le travail jusqu'à maintenant

Voici le GUI lancé avec la classe GuiConférence :



The screenshot shows a Java Swing window titled "J-Confs". It contains two main sections: "Researcher" and "Conference".

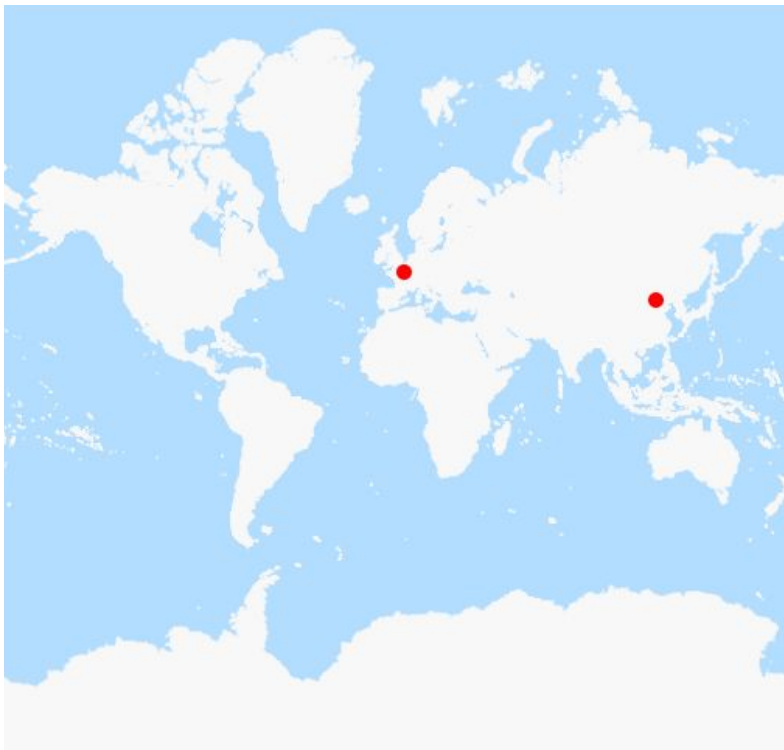
**Researcher Section:**

- Fields: Login, Surname, Firstname, Phone, Group, Mail, Office.
- A "Search" button is located below these fields.

**Conference Section:**

- Fields: Title, Registration Fee, City (containing "Beijing"), Country.
- Date fields: "Date Start" and "Date End", both set to "27/06/2019" with calendar icons.
- Buttons: "Create calendar", "Generate OM", "Generate YS", and "Display Conference Location".

En écrivant une ville dans le champ City et en appuyant sur le bouton Display Conference Location, une carte apparaît avec deux points : le point de départ qui est Paris par défaut (voir le code de la classe MapGUI) et le point d'arrivée (si la ville est présente dans le fichier cities15000.txt).



# ReverseGeocode vs LocationIQ

ReverseGeocode : <https://github.com/AReallyGoodName/OfflineReverseGeocode>

Dans notre projet, le package Geocode sert exclusivement à stocker des villes avec leur latitude et leur longitude. Ces informations sont récupérées via l'extraction d'un fichier. Dans notre projet, il s'agit du fichier cities15000.txt (ce fichier contient toutes les villes de plus de 15000 habitants).

Les limites de reverseGeocode sont :

- Impossible d'inclure une dépendance maven. Il faut télécharger et inclure tout le package de la librairie.
- Les villes sont limitées à cause du fichier. De base, nous avons essayé d'intégrer un fichier contenant toutes les villes du monde mais de part l'immense taille du fichier, le code mettait trop de temps pour répondre (plus de 10 minutes sans réponses).
- De base, le but est de trouver un itinéraire court entre le point de départ du chercheur et le lieu de la conférence. Avec ReverseGeocode impossible d'avoir une adresse précise, le trajet ne sera qu'approximatif (d'une ville à une autre).

LocationIQ : <https://locationiq.com/>

Pour utiliser LocationIQ (qu'il faudra implémenter si vous choisissez cette option), il faut se créer un compte afin de pouvoir requêter les différentes classes. L'API semble beaucoup plus souple, puisqu'elle est capable de donner l'adresse complète via un libellé entré par l'utilisateur. Exemple, si on tape « Université Paris Descartes Paris », l'API sort un fichier json contenant plein d'informations dont la latitude, la longitude mais également le libellé complet. Dans notre exemple les informations importantes sont donc : lat : 48.85139235 lon : 2.34111160499217 et displayName: Université Paris Descartes - Paris V, Rue de l'École de Médecine, Quartier de la Monnaie, Paris 6e Arrondissement, Paris, Île-de-France, France métropolitaine, 75006, France.

Les limites de LocationIQ sont :

- il faut un compte pour pouvoir utiliser les classes en ligne.
- Les villes sont limitées à cause du fichier. De base, nous avons essayé d'intégrer un fichier contenant toutes les villes du monde mais à cause l'immense taille du fichier, le code mettait trop de temps pour répondre (plus de 10 minutes sans réponses).
- De base, le but est de trouver un itinéraire court entre le point de départ du chercheur et le lieu de la conférence. Avec ReverseGeocode impossible d'avoir une adresse précise, le trajet ne sera qu'approximatif (d'une ville à une autre).

# Mapsforge vs Graphhopper

Mapsforge : <https://github.com/mapsforge/mapsforge>

Comme indiqué au début du document Mapsforge n'a pas de documentation en ce qui concerne l'optimisation du trajet. Elle reste excellente pour afficher n'importe quelle carte. De plus le github est assez conséquent, aussi nous vous recommandons d'utiliser GraphHopper ou d'utiliser une autre API plutôt que de perdre du temps comme nous l'avons fait à lire classe par classe le github de mapsforge.

Graphhopper : <https://www.graphhopper.com/>

L'API graphhopper semble être la meilleure api que nous avons trouvé afin de travailler sur une carte (trajet, optimisation, etc.). Elle possède une documentation très fournie et illustrée d'exemples :

- <https://docs.graphhopper.com/>
- <https://graphhopper.com/api/1/examples/#optimization>

Voici une démonstration pour voir ce qu'il est possible de faire : <https://graphhopper.com/maps/>.

Et enfin voici le github pour tout ce qui concerne l'installation : <https://github.com/graphhopper/graphhopper> et pour la dépendance maven : <https://github.com/graphhopper/graphhopper/tree/master/client-hc>.

Autre API :

L'utilisation d'une autre API est autorisée. Toutefois, vous devez en parler avant au professeur. Il serait préférable que cette API ait une dépendance maven a intégré dans le pom.xml et que l'on puisse l'utiliser hors ligne.