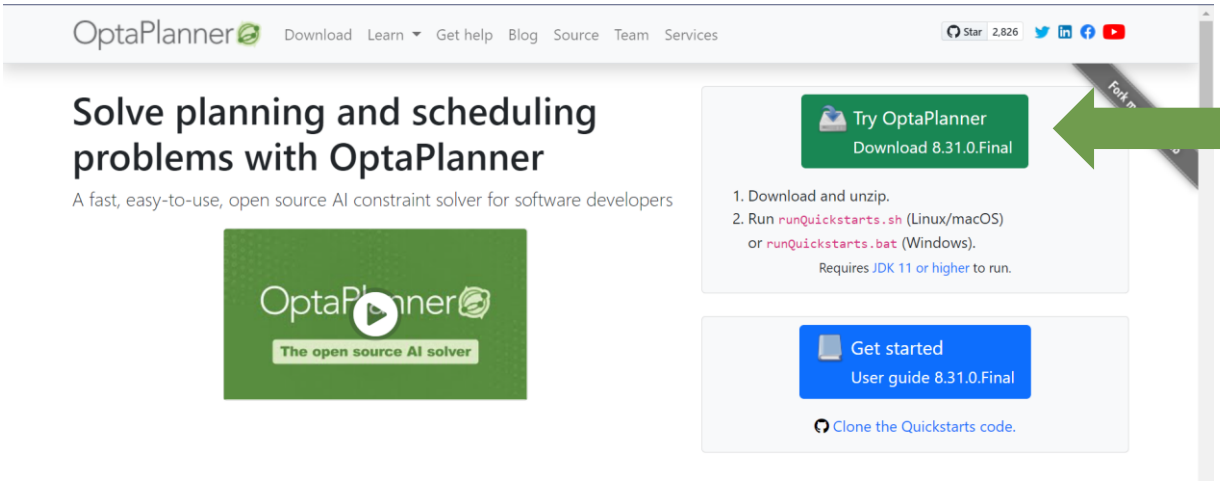
DOCUMENTATION UTILISATEUR OPTAPLANNER

1) Qu'est-ce que c'est ?

OptaPlanner est un solveur de contraintes Open Source développé par Red Hat en 2006. Il résout les problèmes de satisfaction de contraintes avec des algorithmes heuristiques de construction et méta-heuristiques. La promesse d'OptaPlanner est de proposer une solution quasi-optimale en un temps raisonnable à des problèmes d'optimisation complexes et sur des grands jeux de données.

2) Installation

Il ne s'agit pas vraiment « d'installer » OptaPlanner mais plutôt de récupérer des fichiers d'exemples de problèmes d'optimisation mis à disposition pour les modifier et les adapter à vos propres besoins.
Etape 1 : Télécharger ces fichiers sous forme de zip sur : https://www.optaplanner.org/

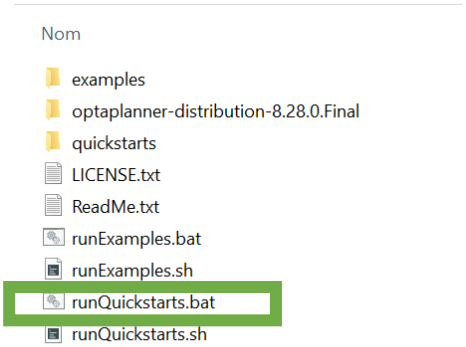


Etape 2 : Extraire les fichiers

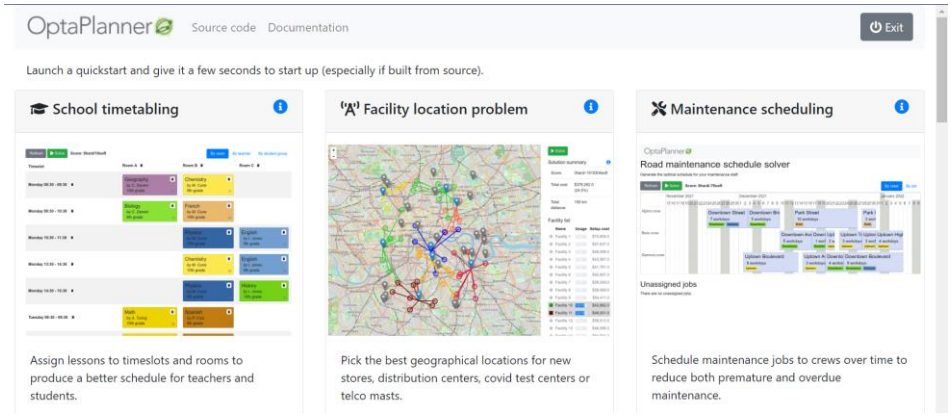
3) Exécution

A ce niveau-là 2 choix s'offrent à vous soit vous pouvez lancer la version « web » d'OptaPlanner qui correspond à une version plus ergonomique, s'ouvrant sur votre navigateur avec quelques exemples de problèmes d'optimisation à lancer :
Etape 3 : Pour accéder à la version « web » :

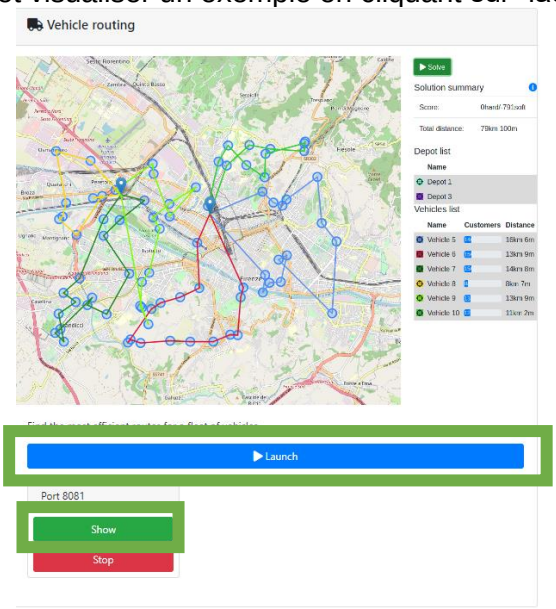
- lancer le fichier "runQuickstarts.bat"



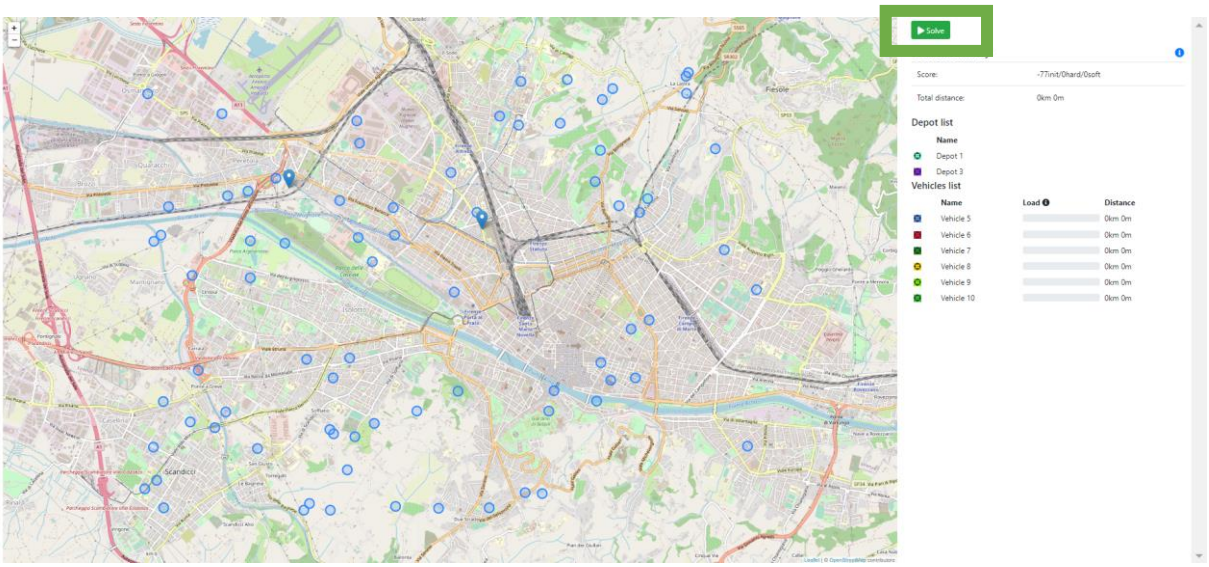
- une page s'ouvre sur le navigateur avec plusieurs exemples



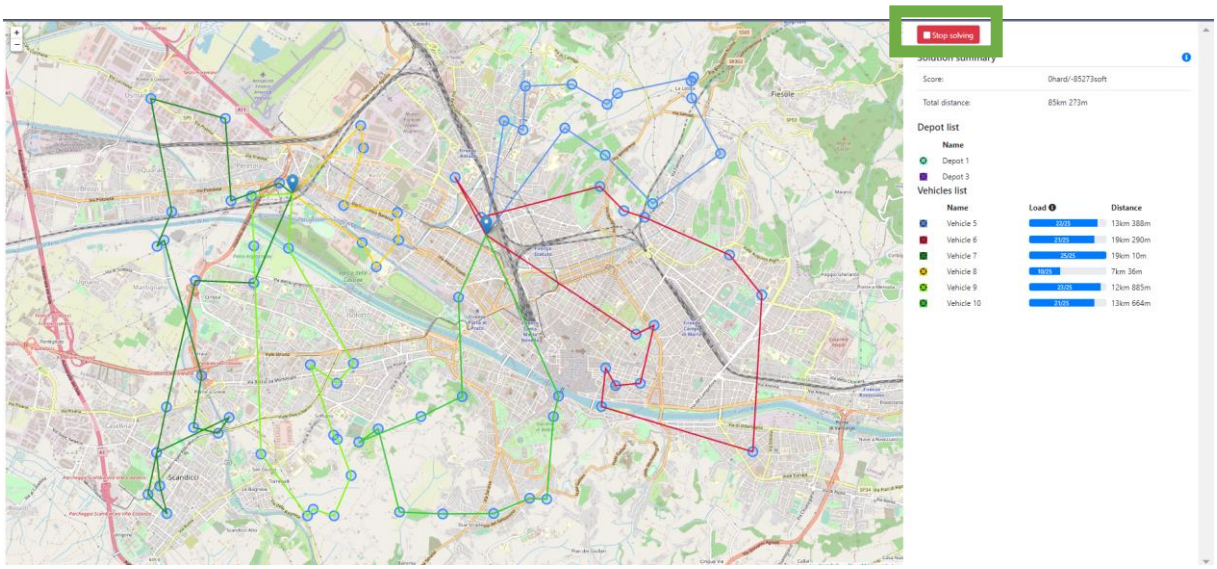
- on peut exécuter et visualiser un exemple en cliquant sur “launch” puis sur “show”



- l’exemple choisi s’ouvre alors sur le port indiqué au-dessus de show et on peut résoudre le problème avec “solve



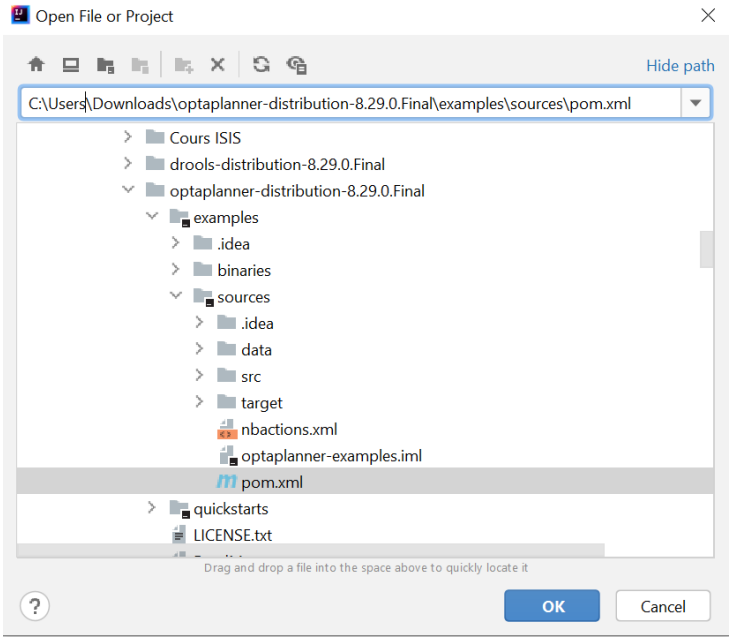
- on suit la résolution en direct et on peut arrêter la résolution à tout moment en cliquant sur « stop solving »



Soit vous pouvez lancer la version « logiciel » d'OptaPlanner qui correspond à une version plus ancienne, s'ouvrant dans une fenêtre java mais avec une plus grande quantité d'exemple à lancer :

Etape 3 bis : Pour accéder à la version « logiciel »:

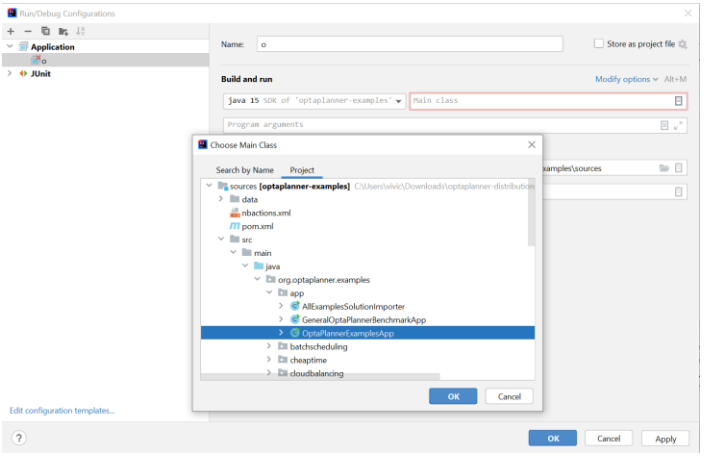
- ouvrir le fichier "examples/src/pom.xml" dans un IDE Java



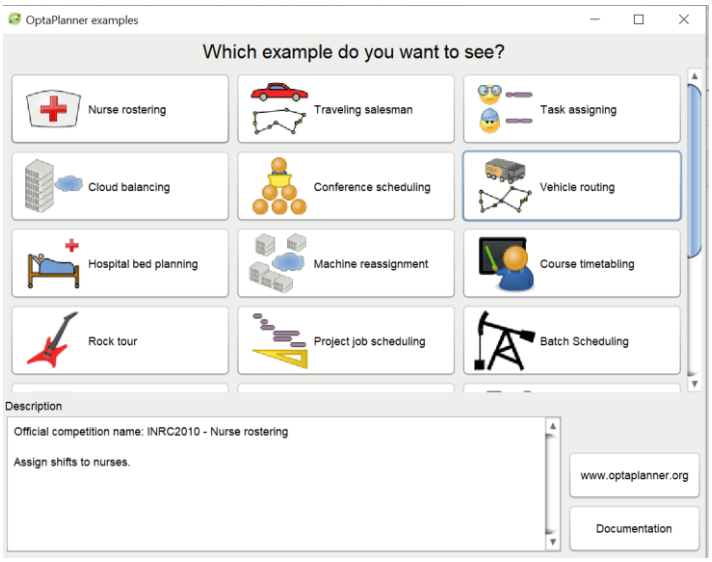
⚠ De préférence utilisez l'IDE intelliJ pour éviter les bugs de construction et d'éxécution

⚠ De préférence utilisez un jdk (java) 15 ou supérieur et n'oubliez pas d'ajouter son chemin à votre variable d'environnement système JAVA_HOME dans les paramètres avancés de votre ordinateur

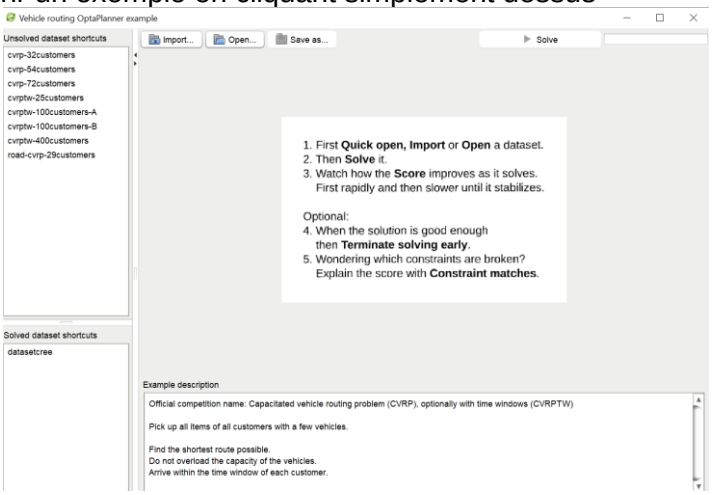
- aller dans l'onglet Run/Edit Configurations puis ajouter une nouvelle "Application" avec pour main class "org.optaplanner.examples.app.OptaPlannerExamplesApp"



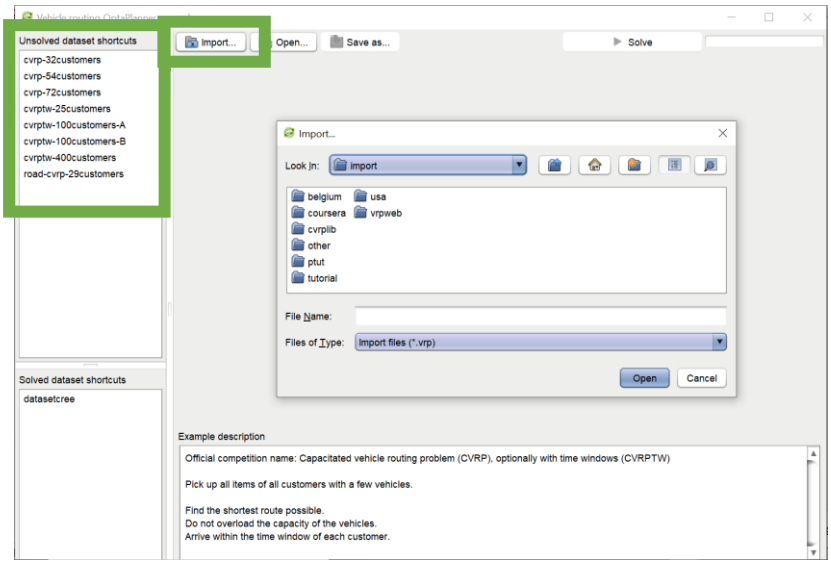
- exécuter le projet (“run”)
- une page s’ouvre avec pleins d’exemples de problèmes d’optimisation et leur descriptions



- on peut ouvrir un exemple en cliquant simplement dessus

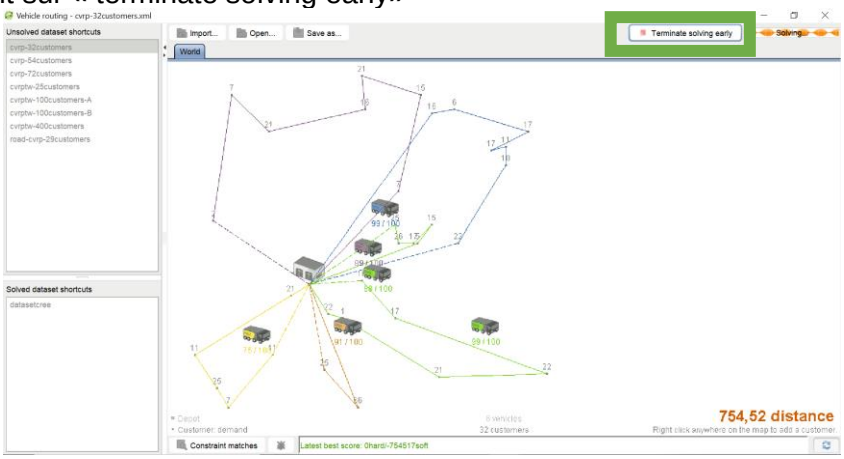


- on peut ensuite choisir un jeu de données à résoudre, soit un de ceux déjà importés par d’autres utilisateurs soit un qu’on a créé soi-même

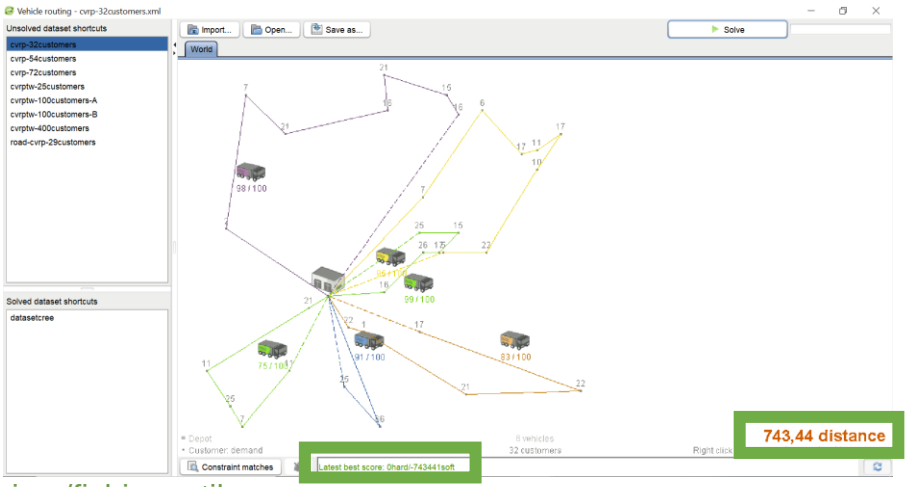


- le jeu de donnée choisi s'ouvre alors et on peut le résoudre en cliquant sur solve

- on suit la résolution en direct et on peut arrêter la résolution à tout moment en cliquant sur « terminate solving early»



- la résolution est terminée lorsque le bouton « solving » réapparait et la distance optimale s'affiche en bas ainsi que le score de résolution des contraintes (cf explications des contraintes)



4) Dossiers/fichiers utiles

Dans cette partie nous ne parlerons que des fichiers de la version « logiciel », qui est la version sur laquelle nous nous sommes penché pour notre projet d'optimisation des flux de brancardiers. Dans l'IDE les fichiers que nous avons identifié comme important sont :

- Le dossier « data » dans lequel on retrouve tous les jeux de données importés par d'autres utilisateurs et où l'on peut créer le sien au format vrp ou xml pour y avoir accès depuis la fenêtre d'exécution

Exemple jeux de données (vrp et xml) :

datasetcree

VEHICLE

NUMBER	CAPACITY
5	100

CUSTOMER

CUST NO.	XCOORD.	YCOORD.	DEMAND	READY TIME	DUE DATE	SERVICE	TIME
0	0	0	0	0	105	0	
1	1	2	1	45	50	5	
2	2	1	1	72	79	5	
3	1	3	1	26	31	5	
4	3	1	1	45	50	5	
5	1	4	1	100	105	5	
6	4	1	1	43	48	5	
7	1	5	1	88	93	5	
8	5	1	1	40	45	5	
9	1	6	1	67	72	5	
10	6	1	1	73	78	5	

```
<VrpVehicleRoutingSolution id="1">
  <id>0</id>
  <name>A-n33-k6</name>
  <distanceType>AIR_DISTANCE</distanceType>
  <distanceUnitOfMeasurement>distance</distanceUnitOfMeasurement>
  <locationList id="2">
    <VrpAirLocation id="3">
      <id>1</id>
      <latitude>34.0</latitude>
      <longitude>31.0</longitude>
    </VrpAirLocation>
    <VrpAirLocation id="4">
      <id>2</id>
      <latitude>45.0</latitude>
      <longitude>55.0</longitude>
    </VrpAirLocation>
    <VrpAirLocation id="5">
      <id>3</id>
      <latitude>70.0</latitude>
      <longitude>80.0</longitude>
    </VrpAirLocation>
    <VrpAirLocation id="6">
      <id>4</id>
      <latitude>81.0</latitude>
```

- Le fichier org/optaplanner/examples/vehiclerouting/domain/location/Location.java permet de construire les coordonnées de chaque point (missions) avec sa longitude et sa latitude

Le fichier org/optaplanner/examples/vehiclerouting/domain/Customer.java permet de construire chaque mission avec son emplacement (location), sa demande (demand) (pas vraiment utile dans notre cas car nous ne parlons pas de capacité), le brancardier qui lui est attribué (vehicle), la mission qui la précède (previous customer) et celle qui l'a suit (next customer). On y trouve aussi une formule permettant de calculer sa distance à la mission précédente (getDistanceFromPreviousStandstill()) et sa distance à la mission 0 (salle de pause) (getDistanceToDepot())

- Le fichier org/optaplanner/examples/vehiclerouting/domain/Depot.java permet de définir les coordonnées de la mission 0
- Le fichier org/optaplanner/examples/vehiclerouting/domain/Vehicle.java permet de définir un brancardier (on n'a pas besoin dans notre cas de la capacité et le dépôt est le même pour tous (missions 0)) avec sa liste de missions
- Le fichier org/optaplanner/examples/vehiclerouting/domain/VehicleRoutingSolution.java permet de configurer la résolution avec la liste des misisons, de leurs empalcements, des brancardiers, du dépôt, et les paramètres de score.

5) Contraintes

Dans OptaPlanner il existe 2 types de contraintes, les contraintes « hard » qui sont les contraintes qu'il est fondamentale de satisfaire et les contraintes « soft » qui sont plus souples comme l'indique leur nom. Un score est calculé par rapport à la satisfaction de ces contraintes et c'est celui-ci qui va permettre de trouver la solution optimale. En effet, on va définir des contraintes soit avec l'API ConstraintStream (solution choisi), soit avec Drools. Avec la première solution, on énonce les contraintes hard et soft dans le fichier org/optaplanner/examples/vehiclerouting/score/VehicleRoutingConstraintProvider.java et on leur attribue des pénalités si elles ne sont pas respectées, pénalités plus ou moins élevée en fonction de leur priorité de satisfaction.

Lors de la résolution OptaPlanner test donc une première solution au hasard puis calcul el score en fonction des contraintes qu'elle satisfait puis chercher une solution avec un score moins élevé et ainsi de suite. La priorité est au score des contraintes hard, c'est-à-dire que si on a pour une solution une score de 100 hard/2 soft ou un score de 2 hard/ 10 000 000 soft, on prendra la première car c'est elle qui a le score hard le plus bas.

Exemple de définition de contrainte hard avec pénalisation :

```
3 usages
protected Constraint vehicleCapacity(ConstraintFactory factory) {
    return factory.forEach(Customer.class) UniConstraintStream<Customer>
            .filter(customer -> customer.getVehicle() != null)
            .groupBy(Customer::getVehicle, sum(Customer::getDemand)) BiConstraintStream<Vehicle, Int
            .filter((vehicle, demand) -> demand > vehicle.getCapacity())
            .penalizeLong(HardSoftLongScore.ONE_HARD,
                    (vehicle, demand) -> demand - vehicle.getCapacity()) BiConstraintBuilder<Vehicl
            .asConstraint( s: "vehicleCapacity");
}
```

6) Liens utiles

Documentation officielle d'OptaPlanner :
https://www.optaplanner.org/docs/optaplanner/latest/quickstart/overview/overview-quickstarts.html

Chaîne YouTube d'OptaPlanner : https://www.youtube.com/@OptaPlanner

Tuto pour réaliser un exemple (School time tabling/ Emploi du temps scolaire) de A à Z :
https://www.youtube.com/watch?v=7IuOA9n6kh0&t=2667s&ab_channel=OptaPlanner

GitHub d'OptaPlanner pour récupérer les fichiers de codes :
https://github.com/kiegroup/optaplanner

Documentation API ConstraintStream :
https://docs.optaplanner.org/8.30.0.Final/optaplanner-docs/html_single/index.html#constraintStreams