VELIBS DE PARIS

Application Java - Extraction de données JSON

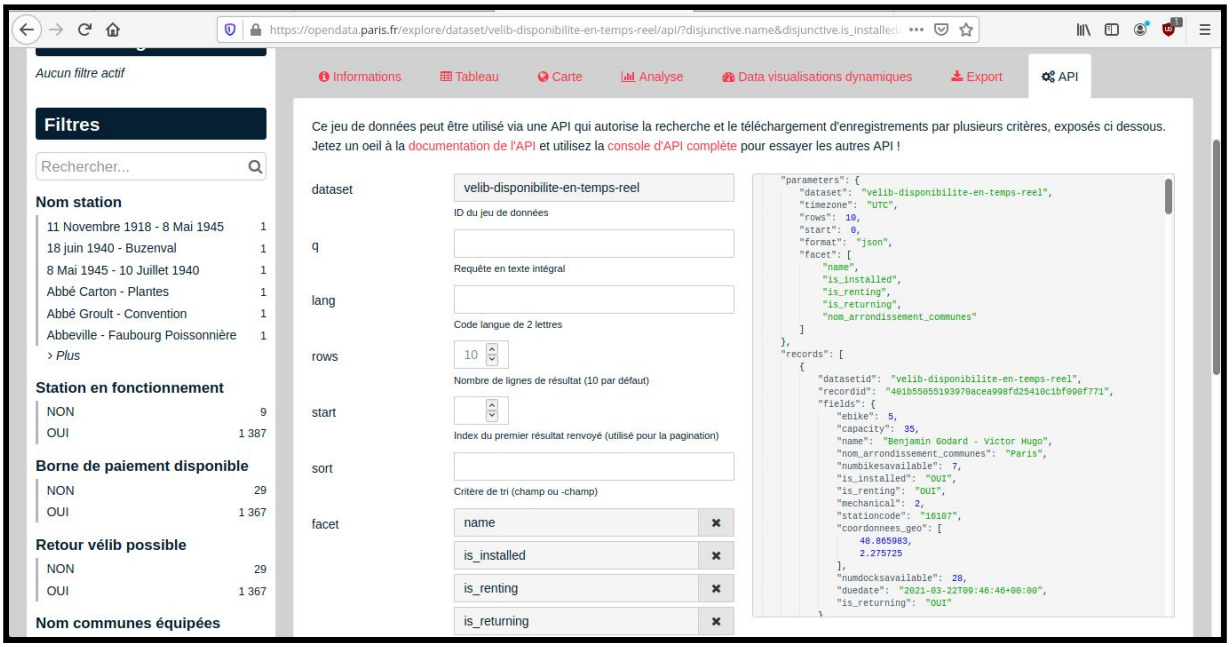
But de l'application

Le but de ce projet est de créer une application pour la mairie de Paris. En effet, elle souhaite proposer une application capable d'indiquer aux utilisateurs la liste des stations Vélibs de Paris, avec des informations supplémentaires tels que :

- Le numéro de la station,
- Son adresse,
- Son état (ouvert/fermée),
- Ses vélos disponibles,
- Ses ports d'attaches disponibles,
- Etc…

Cette application doit être développée en langage Java. Afin de vous aider dans votre tâche, la mairie de Paris propose une API en ligne, basée sur OpenData, qui permet de récupérer en temps réel les informations des stations Vélib de Paris intra muros et des départements limitrophes. Elle est disponible à l'adresse suivante :

https://opendata.paris.fr/explore/dataset/velib-disponibilite-en-temps-reel/



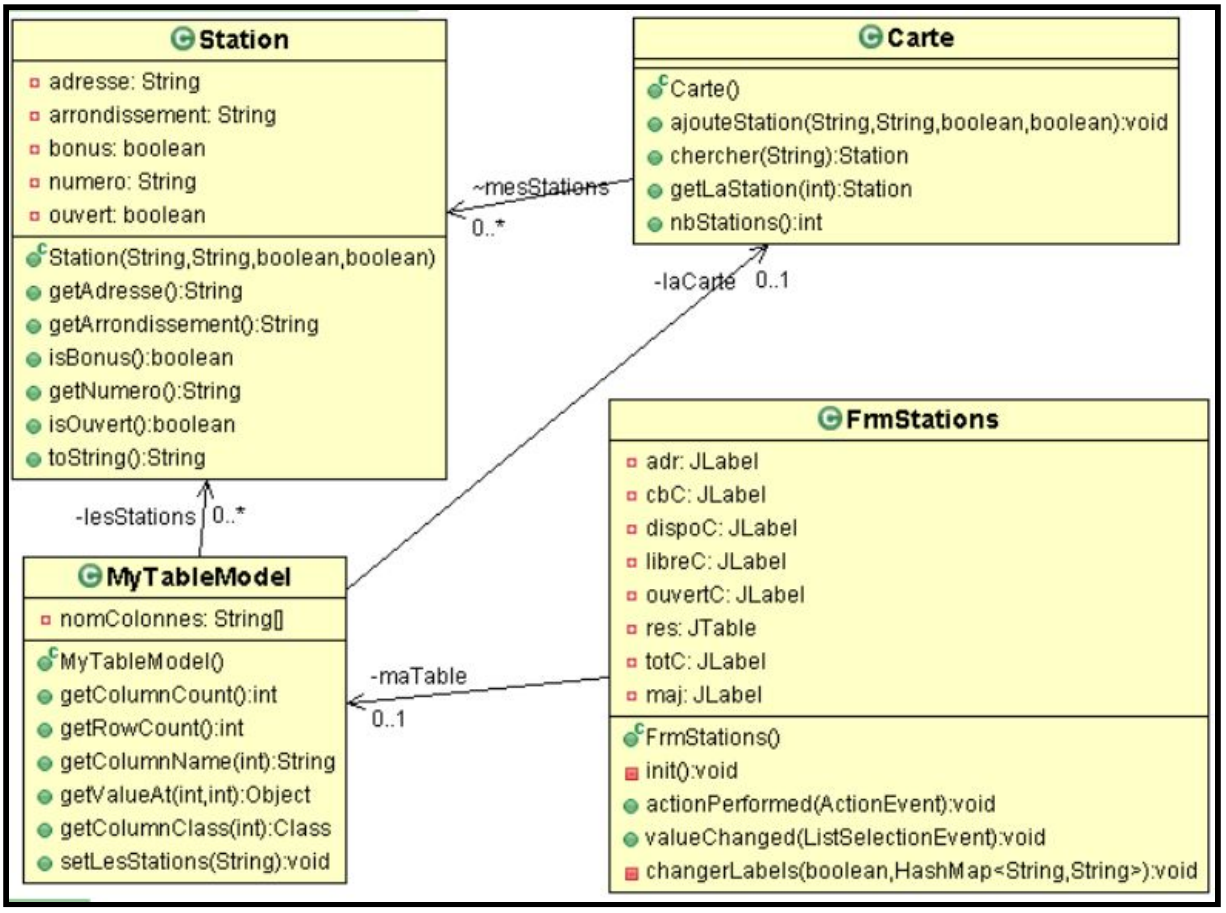
Vue de l'API OpenData Paris

Dans notre cas, pour une question de temps, nous allons juste nous concentrer sur les données retournées par l'API, et non en l'intégration de ce dernier dans notre application.

Ces derniers étant sous format JSON, nous allons devoir effectuer des opérations spécifiques afin de pouvoir les réutiliser.

Les composants

Après concertation, il fût obtenu les composants énoncées dans le diagramme de classes suivant :

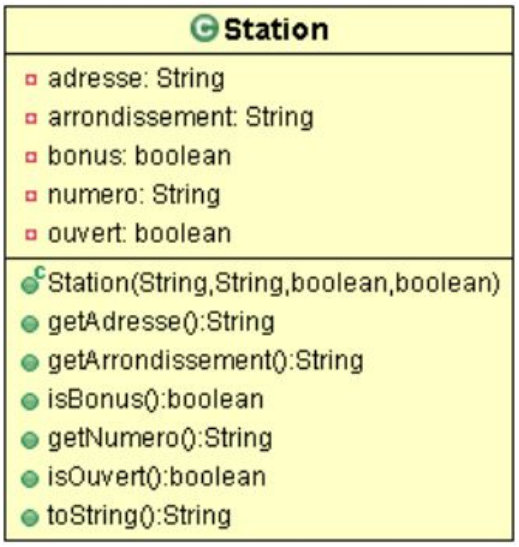


Ajouté à ces composants, se trouvera une classe nommée “JPasserelle”, qui servira de lien entre l’API et votre application.

Travail à faire :

- Créez un nouveau Projet nommé “Vélibs”, à partir du diagramme ci-dessus. Le détail des classes se fera au fur et à mesure de la suite du projet.
- N’oubliez pas d’y ajouter la classe JPasserelle et la classe Vélibs (qui contiendra la méthode Main).

La classe Station



Travail à faire :

- Créez la classe Station.

Remarque : Dans le constructeur, le champ arrondissement sera obtenu à partir du numéro de la station. En effet, observez cet attribut dans l'API, afin de comprendre la construction de ce dernier, et pouvoir repérer très aisément les arrondissements.

Effectuez les tests suivants dans la méthode Main :

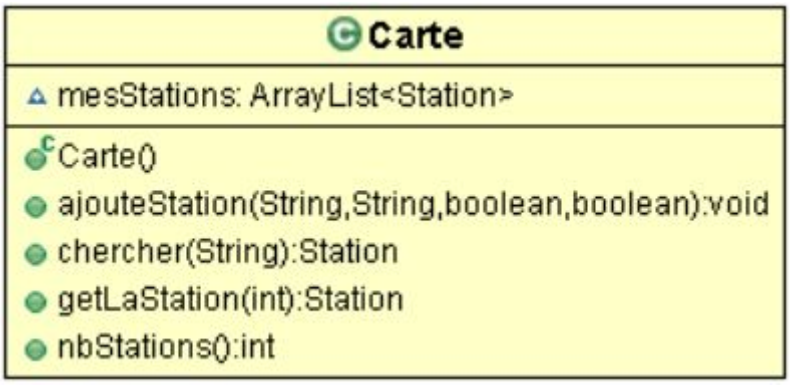
- Testez avec :

```
public class test {
  public static void main(String[] args) {
    Station s = new Station("20021", "15 rue petit", true, true);
    Station s1 = new Station("31023", "11 rue Blanche", true,
true);
    Station s2 = new Station("8567", "45 rue Noire", true, true);

    System.out.println(s.getArrondissement());
    System.out.println(s1.getArrondissement());
    System.out.println(s2.getArrondissement());
  }
}
```

Vous devez obtenir : 20, 93 et 8 comme numéro d'arrondissement.

La classe Carte



Travail à faire :

- Créez la classe Carte
- Le constructeur de cette classe est le suivant :

```
public Carte() {
  mesStations = new ArrayList<Station>();
}
```

Effectuez les tests suivants dans la méthode Main :

- Testez avec :

```
public class test {
  public static void main(String[] args) {
    Carte c = new Carte();
    c.ajouteStation("20021", "15 rue petit", true, true);
    c.ajouteStation("31023", "11 rue Blanche", true, true);
    c.ajouteStation("8567", "45 rue Noire", true, true);

    System.out.println(c.nbStations());
    System.out.println(c.getLaStation(1));
  }
}
```

Vérifiez que votre les stations sont bien ajoutées à votre carte.

La classe JPasserelle

Comme précisé précédemment, cette classe va permettre à l'application de pouvoir récupérer (parser) les informations fournies par l'API dans notre application. Ayant juste ce but précis, toutes les méthodes de cette classe seront à portée classe, de plus elle ne possédera pas de constructeur.

Cette classe contiendra un attribut statique URL :

```
public class JPasserelle {
    private static String url
```

Pour générer votre URL, il faudra passer par l'API : en effet, vous devez préciser les informations que vous souhaitez obtenir dans le fichier JSON généré.

Travail à faire :

- En vous basant sur l'API (et sa documentation), générez le lien du fichier JSON qui devra contenir :
  - La liste de toutes les stations Vélibs (Informations comprises),
  - Être trié soit par le numéro de station (méthode qui risque de ne pas fonctionner de manière optimale), soit par le nom de station,
  - Désactiver les "facets" (Options de classification en JSON, inutile dans notre cas)

Testez directement en observant le fichier JSON affiché sur votre navigateur. Ensuite, valorisez l'attribut URL avec ce lien généré.
Maintenant que nous avons notre fichier JSON, revenons à notre classe JPasserelle. Tout d'abord, nous avons besoin d'une bibliothèque externe Java, qui va nous simplifier la tâche pour parser le fichier. Ce dernier vous est fourni en annexe, vous le trouverez aussi via le lien suivant : https://github.com/stleary/JSON-java
Une fois ce fichier importé, vous allez créer une méthode nommée "TestCarte", afin de vérifier le fonctionnement de notre fonction :

```
public static void testCarte() {
    try {
        // Récupération du fichier JSON en ligne
        InputStream lien = new URL(url).openStream();
        BufferedReader fichier = new BufferedReader(new InputStreamReader(lien, Charset.forName("UTF-8")));
        // Conversion en format JSON Object
        JSONObject json = new JSONObject(fichier.readLine());
        fichier.close();
        for(int i=0;i<json.getInt("nhits");i++) { //Boucle par rapport au nb de stations
            System.out.println(json.getJSONArray("records") // Récupération des stations sous format JSON Array
                               .getJSONObject(i)  // Récupération de la station indiquée en paramètre
                               .getJSONObject("fields") // Récupération des informations de la station sélectionnée
                               .get("name") // Affichage information donnée en paramètre (ici le nom)
                               );
        }

    }
    catch (FileNotFoundException err) {
        System.out.println( "Erreur : le fichier n'existe pas !\n" + err);
    }
    catch (IOException err) {
        System.out.println( "Erreur: \n" + err );
    }
}
```

Dans cette fonction, nous allons récupérer notre fichier JSON via le lien généré précédemment, le parser afin de pouvoir l'utiliser, et enfin afficher les différentes données contenus (dans cet exemple, nous prendrons que le nom de la station).

Travail à faire :

- Codez cette méthode, puis testez la en faisant un appel via la méthode main : JPasserelle.testCarte();
- Vérifier que vous obtenez bien dans l'affichage console la liste de toutes les stations :

```
<terminated> Velibs [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java
8 Mai 1945 - 10 Juillet 1940
11 Novembre 1918 - 8 Mai 1945
18 juin 1940 - Buzenval
Abbé Carton - Plantes
Abbé Groult - Convention
Abbeville - Faubourg Poissonnière
Aboukir - Alexandrie
Adolphe Lalyre - Armand Silvestre
Adolphe Pinard - Victor Hugo
Alésia - Suisses
Alexander Fleming - Belvédère
Alexandre Dumas - Place de la Réunion
Alexandre Dumas - Voltaire
Alexandre Parodi - Quai de Valmy
Alfred de Vigny - Place du Général Brocard
Alfred Roll - Berthier
Alger - Rivoli
Alibert - Jemmapes
```

- En vous inspirant de la méthode ci-dessus, codez désormais une fonction static getCarte(), qui devra retourner une Carte complétée via le fichier JSON en ligne.
- Tester dans la méthode main :

- Testez avec :

```
public class test {
  public static void main(String[] args) {
    Carte c = new Carte();
    c = Passerelle.getCarte();

    System.out.println(c.nbStations());
    System.out.println(c.getLaStation(0));
    System.out.println(c.chercher("4017"));
  }
}
```

La classe FRMStations

Vous pouvez désormais vous occuper de la trame graphique de l'application.

Travail à faire :

- Créez la fenêtre de l'application via l'éditeur, en ajoutant les composants comme dans la capture en début de TP (Fenêtre en 800x600),
- La liste des stations est affichée grâce à une JTable, qui sera chargée avec MyTableModel :

```
res = new JTable(maTable);
res.setPreferredScrollableViewportSize(new Dimension(700,
300));
res.setFillsViewportHeight(true);
res.setRowSelectionAllowed(true);
res.setColumnSelectionAllowed(false);
JScrollPane scrollPane = new JScrollPane(res);
```

La classe MyTableModel

Travail à faire :

- Complétez la classe MyTableModel donnée en annexe. Elle permet de remplir la JTable avec les stations voulues.
- Testez dans la méthode main :

- Testez avec :

```
public class test {
  public static void main(String[] args) {
    Carte c = Passerelle.getCarte();
    System.out.println(c.nbStations()+" stations
chargées\n");

    MyTableModel maTable = new MyTableModel();
    System.out.println(maTable.getValueAt(1,0));
    System.out.println(maTable.getValueAt(1,1));
    System.out.println(maTable.getValueAt(1,2));
    System.out.println(maTable.getValueAt(1,3));
  }
}
```

- Testez enfin votre votre fenêtre :

```
public class test {
  public static void main(String[] args) {
    Carte c = Passerelle.getCarte();
    System.out.println(c.nbStations()+" stations
chargées");
    new FrmStations();
  }
}
```

FrmStations - Evenementiel

Travail à faire :

- Écrivez le gestionnaire d'événement suivant. Il devra être déclenché lors du choix d'un arrondissement ou département (choix d'un des JRadioButton) :

```
public void actionPerformed(ActionEvent e) {
  maTable.setLesStations(e.getActionCommand());
  this.changerLabels(false, null);
  res.revalidate();
  res.clearSelection();
  this.repaint();
}
```

- Écrivez le gestionnaire d'événement suivant (Attention, la variable info doit être réadaptée) :

```
public void valueChanged(ListSelectionEvent e) {
  int numLigne = res.getSelectedRow();
  if(numLigne != -1) {
    String numStation = maTable.getValueAt(numLigne,0).toString()
    String adresse = maTable.getValueAt(numLigne,1).toString();
    HashMap<String,String> info = Passerelle.getDispo(adresse,
numStation);
    changerLabels(true, info);
    this.repaint();
  }
}
```

- Il sera déclenché lors du choix d'une des stations dans la JTable grâce à :

```
res.getSelectionModel().addListSelectionListener(this);
```

- Écrivez la méthode changerLabels qui est utilisée dans les deux gestionnaires d'événements précédents et qui permet de mettre à jour les labels d'informations d'une station. Attention, lorsqu'on clique sur un des JRadioButton, les labels redeviennent invisibles.
- Testez le bon fonctionnement de l'application finale.