ENSG Géomatique

ÉCOLE NATIONALE DES SCIENCES GÉOGRAPHIQUES

Université Gustave Eiffel

Projet en programmation java

Conception et développement d'une application informatique.

Marie-Dominique Van Damme

Cycle ingénieur 1ère année
2021-2022

Table des matières

1 Introduction

1.1 Objectifs pédagogiques

Ce projet informatique intervient pour vous permettre de faire une synthèse entre les différents cours suivi cette année en informatique, à savoir ; l'algorithmique, outils de gestion de projet, l'analyse informatique et la programmation orientée objet en Java.

Vous devrez en partie (dans le désordre) :

- formaliser une analyse du sujet avec les outils UML ;
- Écrire un ou plusieurs algorithmes ;
- Mettre en œuvre les concepts de l'orienté objet ;
- Charger, manipuler, visualiser des données géographiques ;
- Présenter les algorithmes utilisés devant un auditoire
- Rédiger un rapport technique, expliquer clairement un algorithme, son contexte.

1.2 Organisation Générale

Le projet informatique d'une durée de 42h s'étend sur 12 séances de 3h et concerne essentiellement les semaines du 11-15 avril et 18-22 avril et la semaine du 9-13 mai. Les soutenances auront lieu les jeudi 19 mai et vendredi 20 mai matin.

Vous travaillerez en trinôme. Chaque groupe sera composé de trois étudiants. La composition des groupes est laissée libre. Cette année il n'y aura qu'un seul sujet. Mais vous pourrez choisir la ville que vous voudrez pour récupérer les données. Les groupes doivent être précisés par mail à ***marie-dominique.van-damme@ensg.eu*** le 07 avril à 17H00 au plus tard.

Certaines séances seront encadrées, tandis que d'autres s'effectueront en autonomie. Les enseignants ne seront pas tous présents aux séances encadrées (roulement). Votre présence est bien entendu obligatoire durant l'ensemble des créneaux réservés.

La soutenance du projet comprend une présentation du sujet, du travail réalisé ainsi qu'une démonstration, tests en conditions réelles. La soutenance est de 15 minutes par groupe plus 10 minutes de questions.

1.3 Le déroulement du projet

- Présentation du sujet : 04/04/2022
- Constitution des groupes : du 04/04/2022 au 07/04/2022
- Démarrage du projet : vendredi 08/04/2022 matin
- Remise des livrables du projet : vendredi 13/05/2022
- Soutenances : 19/05/2022 et 20/05/2022

1.4 Livrables du projet

Il vous est demandé 2 rapports à rendre avant de faire votre soutenance. Les documents sont à rendre une semaine avant la fin du projet sur la plateforme Teams (les consignes seront données plus tard). Le projet se clôt par une présentation du travail oralement devant vos enseignants (ppt, vidéo de démonstration, etc.).

1.4.1 Rapport d'analyse

Le rapport d'analyse permet de formaliser une analyse du cas d'étude, mieux comprendre le fonctionnement demandé et modéliser l'application avant sa réalisation. Les modèles seront représentés par des diagrammes UML. Vous pouvez réaliser :

- ◎ un diagramme d'utilisations afin d'exprimer les grandes fonctionnalités de votre application
- ◎ un ou plusieurs diagrammes de classes suivant les composants identifiés. Les classes de l'interface graphique ne doivent pas forcément être modélisées.
- ◎ un diagramme d'activités pour modéliser votre processus de décision
- ◎ un diagramme de composants

Ce premier travail aboutit à un rapport d'analyse de minimum 5 pages (avec du texte). Ce rapport écrit peut être mis à jour tout au long du déroulement du projet. Il concerne les spécifications fonctionnelles de l'outil à développer :

- ◎ Exprimer le sujet avec votre compréhension ;
- ◎ Qui utilisera le logiciel et pour quoi faire ?
- ◎ Quels sont les différentes fonctionnalités à implémenter ?
- ◎ Comment les actions devront-elles se dérouler ?
- ◎ Quelles informations seront utilisées pour cela ?
- ◎ Poser des hypothèses ;
- ◎ Citez vos sources : vous avez parfaitement le droit de vous inspirer de solutions existantes, voire de les utiliser intégralement (on ne cherche pas à réinventer la roue), mais ayez l'honnêteté intellectuelle de citer les origines des contenus utilisés.
- ◎ Comment vous avez planifié votre projet (diagramme de GANTT par exemple).
- ◎ Conclusion.

Pour construire les diagrammes UML, le logiciel *StarUML* est installé sur les machines *Programmation*. Pour d'autres outils, n'hésitez pas à consulter le cours d'analyse sur la FAD.

1.4.2 Code source et rapport de synthèse

Dans la deuxième partie de ce projet, vous allez réaliser le code de votre application. Vous êtes amené à mettre en œuvre les concepts de l'orientée objet, en programmant avec le langage Java. Ainsi, il faut :

- ◎ Le code, documenté et testé sur **GIT**
- ◎ produire une documentation de votre code ;
- ◎ rédiger un rapport de synthèse (3 à 5 pages) ; ce rapport doit mettre en évidence « Comment s'est passé votre projet ? Si vous abandonnez une solution/ une hypothèse en cours de route (par exemple parce que vous aurez trouvé une meilleure méthode). Expliquer votre cheminement. Citez toujours vos sources : d'où vient votre code ou une partie du code ? ;
- ◎ rédiger un manuel d'utilisation de votre application dans le fichier README de votre repository sur github ;
- ◎ rédiger une documentation pour installer votre application dans le fichier README de votre repository sur github.

1.5 Conseils

- ◎ Fixez vous des objectifs à court terme et évaluez les : « ce matin, je développe telle fonctionnalité » / « j'ai perdu du temps à cause de problèmes d'imports et finalement, je n'ai pas encore complètement implémenté cette fonctionnalité ».
- ◎ Organisez des points d'avancement rapides et réguliers (5min max) pour répondre aux questions suivantes : qu'est ce que j'ai terminé depuis la dernière réunion ? Qu'est ce que j'aurai terminé d'ici la prochaine réunion ? Quels obstacles me retardent ?

- ◎ Mettez à l'écrit le bilan de vos points d'avancement.
- ◎ En matière de développement, vous allez probablement écrire beaucoup de code. Respectez des règles simples d'organisation : 20 lignes max par méthode, 1 classe par fichier, découpage en package, sous-package, etc.. Un code bien structuré sera plus facile à comprendre et à debugger.
- ◎ Pour vos rapports :
  1. Si vous abandonnez une solution en cours de route (parce que, par exemple, vous avez trouvé une meilleure méthode), ne l'abandonnez pas complètement. Expliquer votre cheminement et montrer pourquoi vous avez privilégié une solution plutôt qu'une autre, c'est un vrai plus.
  2. Illustrez vos résultats et expliquez vos choix : une phrase comme « on choisit finalement la première solution, mais avec les bons paramètres » n'explique pas quels sont les paramètres choisis, pourquoi ils le sont, etc.
  3. Citez vos sources : vous avez parfaitement le droit de vous inspirer de solutions existantes, voir de les utiliser intégralement (on ne cherche pas à réinventer la roue), mais ayez l'honnêteté intellectuelle de citer les origines des contenus utilisés.
- ◎ Pour la présentation :
  1. Préparez un scénario pour votre démonstration et faites une vidéo (plus sûr). Ouvrir l'application pour montrer qu'il se passe quelque chose lorsque l'on clique sur un bouton n'apporte rien.
  2. Ne projetez pas l'intégralité de votre code lors des soutenances.
  3. Si vous projetez des diagrammes UML, rendez les lisible ou zoomez sur une petite partie.

Attention : il n'est pas interdit que les groupes s'entraident ou s'inspirent de solutions existantes, mais cela ne signifie pas que vos fichiers doivent être identiques à ce que vous avez trouvé (sur internet par exemple, ou sur d'anciens projets d'élèves). L'ENSG dispose d'un outil qui permet de vérifier toutes similitudes entre projets et sources internet.

1.6 Évaluation

La livraison d'une application fonctionnelle n'est pas la seule fin du projet. La qualité du code produit (clarté, réutilisabilité, ...), la documentation et les tests comptent pour une part importante de la note finale. L'analyse du sujet et la pertinence de la modélisation UML interviennent également dans la note finale. La grille d'évaluation est présentée en tableau 1.1.

Objet	**Critères**	**Barème**
Rapports analyse	Reformulation du sujet, Hypothèses de travail, Modélisation UML, Nombre de fonctionnalités, Forme du rapport.	5
Rapport de synthèse	Retour d'expérience, Explication des algorithmes et leurs choix, Vos apports, Forme du rapport.	3
Code	Mise en œuvre des concepts objets, Clarté des notations, respect des conventions, Organisation modulaire, Tests, Manuel d'utilisation et d'installation.	7
Présentation	Qualité de l'expression orale, Qualité du support, Respect des contraintes de temps, Réponses aux questions, Démonstration dynamisme, clarté.	5

Table 1.1 – Grille évaluation

2 Sujet : Instructions de navigation pour un déplacement à pied à partir des données OSM

Nombre d'étudiants : 3 ou 4.

2.1 Sujet

Les besoins de services de navigation pour des itinéraires à pied sont de plus en plus nécessaire dû à l'avènement des applications mobiles. Les points de repères utilisés pour la description de l'itinéraire dans ce contexte ne sont pas les mêmes que ceux utilisés pour définir les instructions de navigation en voiture. Donner des noms de rue n'est pas la meilleure solution car il faut chercher les panneaux qui sont souvent cachés. En revanche proposer une instruction du type "Tourner à droit après le MacDo" est bien plus pratique car l'enseigne est visible de loin. Des travaux de recherche existent pour déterminer les points de repères les plus appropriés pour les piétons [1].

Le but de ce projet est de construire de manière automatique les instructions de navigation pour un déplacement à pied en s'inspirant de la méthode détaillée dans [2] en employant les données d'OpenStreetMap dans une interface graphique.

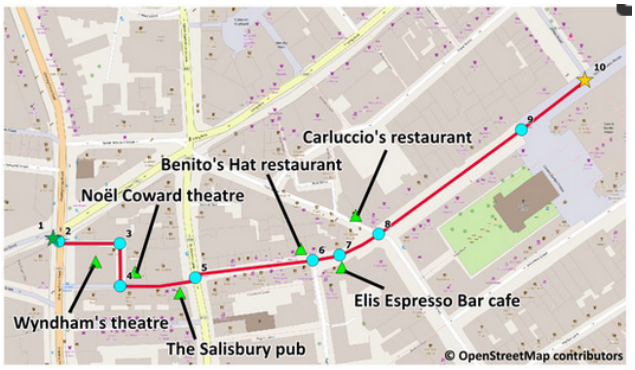

FIGURE 2.1 – Exemple d'instructions calculées avec les points de repères employés (source [1])

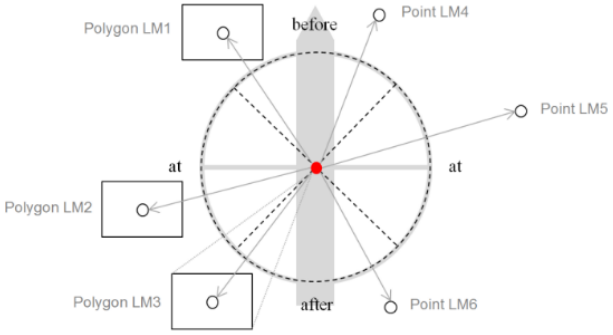

FIGURE 2.2 – Calcul des prépositions des points de repères par rapport aux points de décision (source [2])

A l'issu du service du calcul d'un itinéraire de l'IGN, une suite de morceau de tronçon d'itinéraire (*step*) est identifiée et représente les changements de direction ou noms de rue. Chaque *step* constitue potentiellement un point de décision où une instruction doit alors être calculée : "tourner à droite après le MacDo".

Pour calculer les instructions :

- Il faut ensuite identifier le meilleur point de repère pour chaque morceau d'itinéraire depuis la base des toponymes d'OSM parmi des candidats dont vous devrez définir les caractéristiques.
- (Partie optionnelle) Parmi les candidats retenus, il faudra calculer ceux qui sont visibles depuis le point de décision en prenant en compte le thème Bati d'OSM.
- Une fois le point de repère identifié, il faut définir quelle préposition prendre suivant son emplacement spatial par rapport au tronçon d'itinéraire suivant : "avant", "après", "à".
- Enfin, il reste à générer les instructions calculées par cette approche. Si vous avez le temps, comparez-les aux instructions d'un calcul d'itinéraire en ligne.

Libre à votre groupe de choisir la zone d'étude en France métropolitaine (le service de calcul d'itinéraires de l'IGN ne fonctionne dans cette zone uniquement, il me semble). Assurez-vous que votre zone d'étude a beaucoup de données dans OSM pour donner des bons candidats aux calculs d'instruction.

2.2 Biblio sur le sujet

⊚ [1] Rousell, A. ; Zipf, A. Towards a Landmark-Based Pedestrian Navigation Service Using OSM Data. ISPRS Int. J. Geo-Inf. 2017, 6, 64. https ://doi.org/10.3390/ijgi6030064

⊚ [2] Graser, A. (2017). Towards landmark-based instructions for pedestrian navigation systems using OpenStreetMap. AGILE 2017, Wageningen, Netherlands.

3 Annexes

3.1 Ressources internet

Des liens sur la programmation d'interface en java :

`https://www.youtube.com/watch?v=YhpzUubGg4Q`
`https://www.youtube.com/watch?v=SQae_yJBK1c`
`https://www.youtube.com/watch?v=-UvxwleNA20`

3.2 Service de calcul d'itinéraire

Le service de calcul d'itinéraires de la plateforme du Géoportail de l'IGN (`https://geoservices.ign.fr/documentation/services/api-et-services-ogc/itineraires/documentation-du-service-du-calcul`) permet d'obtenir un itinéraire et les étapes intermédiaires entre deux points selon un profil de parcours (piéton).

Les résultats de l'appel au service de calcul d'itinéraire de l'IGN sont retournés dans un format JSON. Pour cela, nous vous conseillons d'utiliser la librairie GSON (fourni dans le code source). Cette librairie permet de transformer un format JSON en classes JAVA (désérialisation). Pour info, l'opération inverse est la sérialisation et permet de transformer des objets Java dans un format JSON. La désérialisation consiste à mapper la structure du fichier JSON en classe JAVA, c'est à dire on cherche à associer à chaque clé-valeur du fichier JSON soit un attribut (cas simple) ou une classe (structure complexe). Voici une première ébauche :

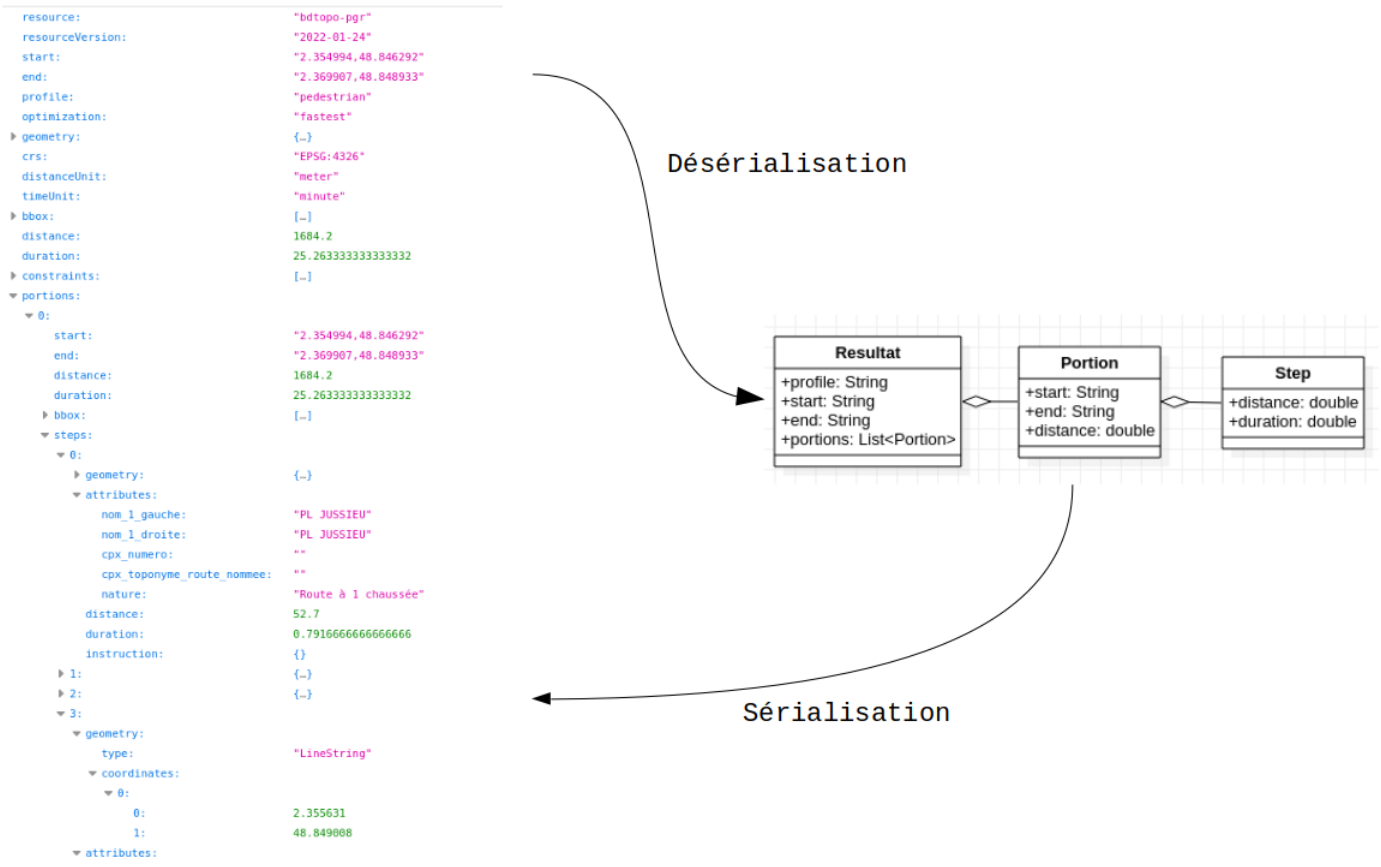

FIGURE 3.1 – Désérialisation des résultats du calcul d'itinéraire

Ci-dessous, vous avez un exemple d'appel au service de calcul d'itinéraire de l'IGN dans le langage Java. Cet exemple est à intégrer dans votre modélisation. Pour qu'il fonctionne, il faut qu'il soit accompagner des classes *Resultat* et *Portion* (à vous de continuer pour les autres classes).

```
...

  public class MainExempleItineraireIgn {

    public static void main(String[] args) {

          String url = "https://wxs.ign.fr/calcul/geoportail/itineraire/rest/1.0.0/route?"
                   + "resource=bdtopo-pgr&profile=pedestrian&optimization=fastest"
                   + "&start=2.3547965564258226,48.84615526548436"
                   + "&end=2.3698762993655165,48.84911418041841"
                   + "&intermediates="
                   + "&constraints={\"constraintType\":\"banned\",\"key\":\"wayType\","
                   + "               \"operator\":\"=\",\"value\":\"tunnel\"}"
                   + "&geometryFormat=geojson&crs=EPSG:4326"
                   + "&getSteps=true&getBbox=true&waysAttributes=nature";
          String txtJson = HttpClientIgn.request(url);
          System.out.println(txtJson);

          Gson gson = new GsonBuilder().create();
          Resultat itineraire = gson.fromJson(txtJson, Resultat.class);
          for (Portion portion : itineraire.getPortions()) {
                System.out.println(portion.getDistance());
          }
  }

}
```

Les coordonnées des points de départ et d'arrivée de l'itinéraire sont à initialisés dans l'URL (*&start=* et *&end=*).

Pour plus de précision sur le principe de mapping, n'hésitez pas lire la documentation du paragraphe **50.3. La désérialisation** de ce site internet `https://www.jmdoudoux.fr/java/dej/chap-gson.htm`.

3.3 Données d'OpenStreetMap

Le but de cette section est de vous donner les briques de code afin d'interroger la base de données d'OpenStreetMap.

L'API *Overpass* vous permet d'exécuter des requêtes depuis les serveurs OpenStreetMap afin d'extraire leurs données comme par exemple les pistes cyclables, les banques, les fontaines d'eau, les arrêts de bus et beaucoup plus. Précisez toujours dans la requête, l'emprise de votre recherche.

Le wiki OpenStreetMap `https://wiki.openstreetmap.org/wiki/Tag:highway=bus_stop` contient une documentation complète de tous les objets (Features) suivant leur Tag.

Les données d'OSM sont composées de géométries (*node* et *way*) et de tags qui permettent de définir les attributs :

Listing 3.1 – Extrait XML qui décrit le point "Centre documentation" de l'ENSG

```
<node id="2039003175" lat="48.8409389" lon="2.5874552" version="8"
        timestamp="2021-09-07T06:07:30Z"
        changeset="110833704" uid="12243582" user="Aubin Bettiol">
    <tag k="access" v="restricted"/>
    <tag k="amenity" v="library"/>
    <tag k="name" v="Centre de documentation scientifique"/>
    <tag k="opening_hours" v="Mo-Fr 09:30-17:30"/>
    <tag k="operator" v="ENSG Géomatique"/>
    <tag k="operator:type" v="university"/>
</node>
```

Les requêtes à l'API doivent contenir à minima une bouding box et éventuellement des filtres sur les attributs (*tag*). Pour en savoir plus sur le système de requête :
`https://wiki.openstreetmap.org/wiki/Overpass_API/Language_Guide`.

Si vous voulez tester vos requêtes, n'hésitez pas à utiliser le plugIn dans QGIS : *QuickOSM*.

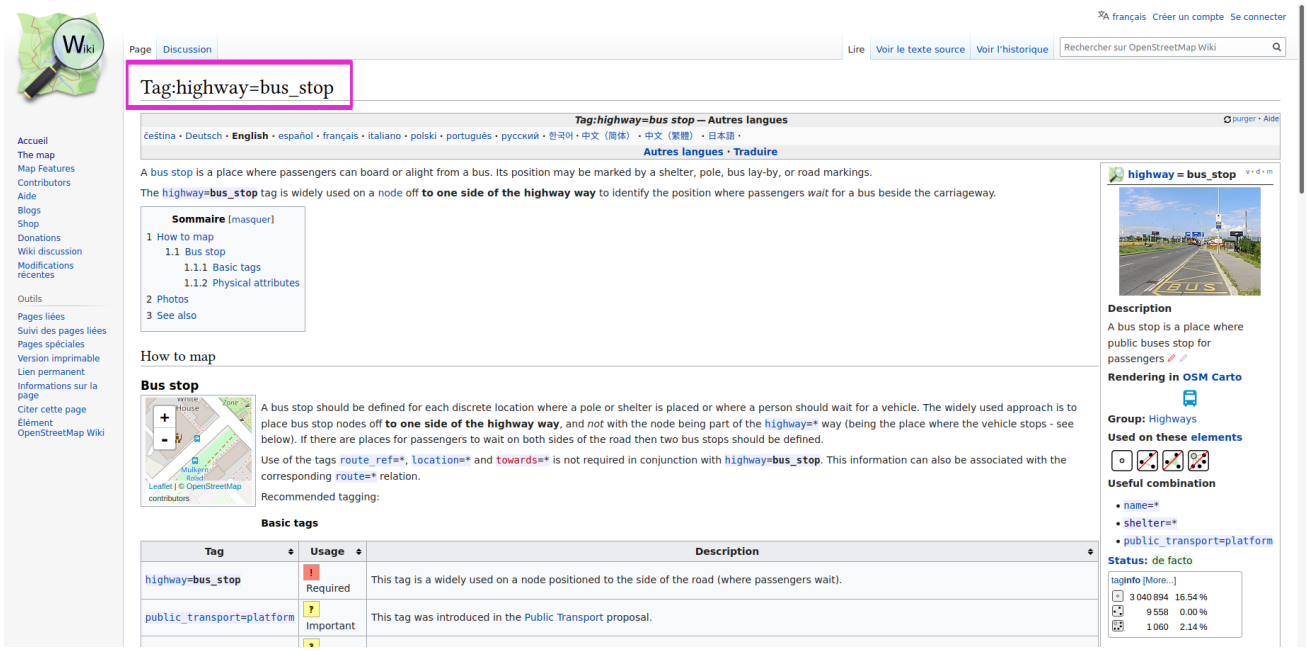

FIGURE 3.2 – Documentation des arrêts de bus dans OSM

Ci-dessous, vous avez un exemple d'appel au service d'OSM pour récupérer des données sur l'emprise (N,E,S,O). Le résultat du service est dans un format XML. Il n'est pas nécessaire de faire un désérialisation. Ici on parse le résultat de la requête *HTTP* (voir le Listing 3.1) en précisant les tags XML comme par exemple : *osm*, *node*, *lat*, *lon*, *tag*, etc.

```
...

public class MainExampleOpenStreetMap {

    private static double E = 2.59042;
    private static double O = 2.58640;
    private static double S = 48.84051;
    private static double N = 48.84202;

    public static void main(String[] args) throws ParserConfigurationException {

        String dataRequest = "<osm-script>"
                 + "<union>"
                 + "<query type=\"node\">"
                 + "<bbox-query e=\"" + E + "\" n=\"" + N + "\" s=\"" + S + "\" w=\"" + O + "\" />"
                 + "</query>"
                 + "</union>"
                 + "<print mode=\"meta\"/>"
                 + "</osm-script>";
        String xmldata = OsmXmlHttpClient.getOsmXML(dataRequest);
        // System.out.println(xml);

        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        factory.setNamespaceAware(true);
        DocumentBuilder builder = factory.newDocumentBuilder();
        try {
            Document doc = builder.parse(new ByteArrayInputStream(xmldata.getBytes()));
            doc.getDocumentElement().normalize();
            Element root = (Element) doc.getElementsByTagName("osm").item(0);

            int nbNoeuds = root.getElementsByTagName("node").getLength();
            for (int i = 0; i < nbNoeuds; i++) {

                Element elem = (Element) root.getElementsByTagName("node").item(i);

                // On récupère son ID
                long id = Long.valueOf(elem.getAttribute("id"));

                // on récupère sa géométrie
                double lat = Double.valueOf(elem.getAttribute("lat"));
                double lon = Double.valueOf(elem.getAttribute("lon"));

                for (int j = 0; j < elem.getElementsByTagName("tag").getLength(); j++) {
                    Element tagElem = (Element) elem.getElementsByTagName("tag").item(j);
                    String cle = tagElem.getAttribute("k");
                    String val = tagElem.getAttribute("v");

                    if (cle.equals("name")) {
                        System.out.println(cle + "--" + val);
                        System.out.println(lon + "," + lat);
                    }
                }

            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

3.4 Projet Java et Interface Graphique

3.4.1 Environnement de développement

Afin de faciliter l'import de librairies externes (JTS, GSON, GEOTOOLS, etc.) pour les manipulations des géométries, les projections et la désérialisation, le squelette du code source vous est donné sous la forme d'un projet Maven.

Pour accéder au code et le modifier :

1. Copier le code source (i.e. projet) sur le disque local de votre ordinateur (surtout pas sur une machine distante ou sur votre répertoire partagé)
2. Ouvrir Eclipse
3. Cliquer sur File > Import
4. Chercher Maven dans la barre de recherche de la boite de dialogue
5. Choisissez : "Select Existing Maven Projects"
6. Cliquer sur "Next"
7. Cliquer sur "Browse" et choisissez le répertoire du projet
8. Cliquer sur "Next"
9. Cliquer sur "Finish"

Ca y est, vous êtes fin prêt pour commencer à coder.

Pour tester l'installation, vous pouvez exécuter le programme en lançant la méthode *main* qui se trouve dans une des classes du package *exemple MainExempleXXXX*. Vous devrez voir apparaître une interface graphique ou des résultats dans la console.

3.4.2 Interface Graphique

Si vous voulez, vous pouvez utiliser l'interface graphique qui vous est proposée. Elle est constituée d'une fenêtre (définie dans la classe *MainExempleAppli*). La fenêtre utilise la classe *MapPanel* qui gère l'affichage des images aériennes.

Pour dessiner sur la carte, vous pouvez ajouter une méthode *paintAppliInstruction* que vous appellerez dans la méthode *paintComponents* dans la classe *MapPanel* (vers la ligne 480) :

```
  protected void paintComponent(Graphics gOrig) {
      super.paintComponent(gOrig);
      Graphics2D g = (Graphics2D) gOrig.create();
      try {
          paintInternal(g);
          paintAppliInstruction(g);
      } finally {
          g.dispose();
      }
  }


private void paintAppliInstruction(Graphics2D g2d) {
    // Si besoin !
    // Ecrire le code de l'appli qui dessine sur la carte
}
```

Pour ajouter des évènements sur la carte ou ajouter de nouveaux panneaux (JPanel), ajouter le code dans la classe *MainExempleAppli* :

```
...

  mapPanel.addMouseListener(new MouseListener() {
      public void mouseReleased(MouseEvent e) {
          }

          public void mousePressed(MouseEvent e) {
                  Point p = e.getPoint();
                  System.out.println("Les coordonnées graphiques de la souris: " + p.x + "," + p.y);
                  // mapPanel.repaint();
          }

          public void mouseExited(MouseEvent e) {
          }

          public void mouseEntered(MouseEvent e) {
          }

          public void mouseClicked(MouseEvent e) {
          }
  });
```