

TP sur Jena

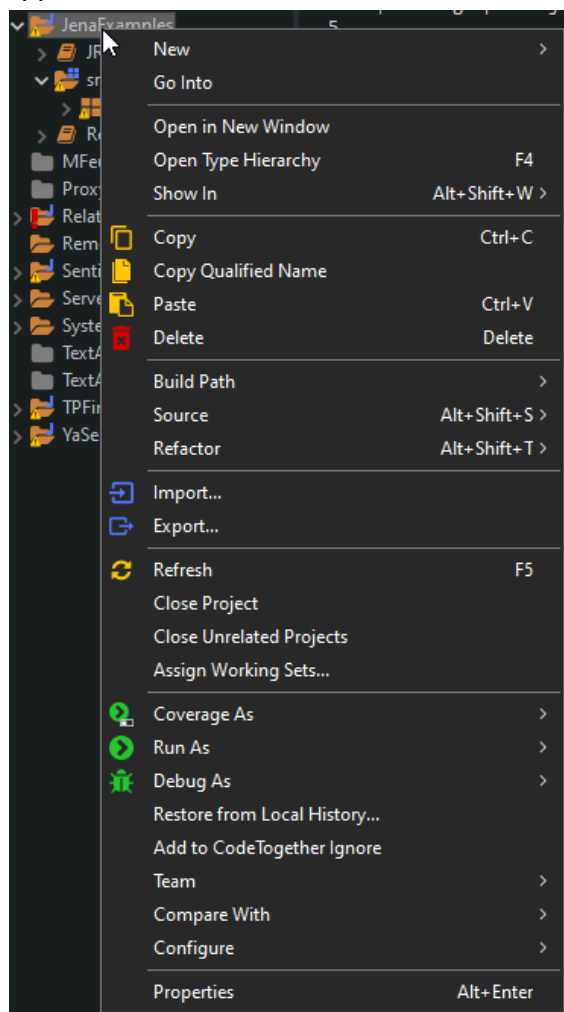
A. Préliminaires: installation et mise en route

Prérequis:

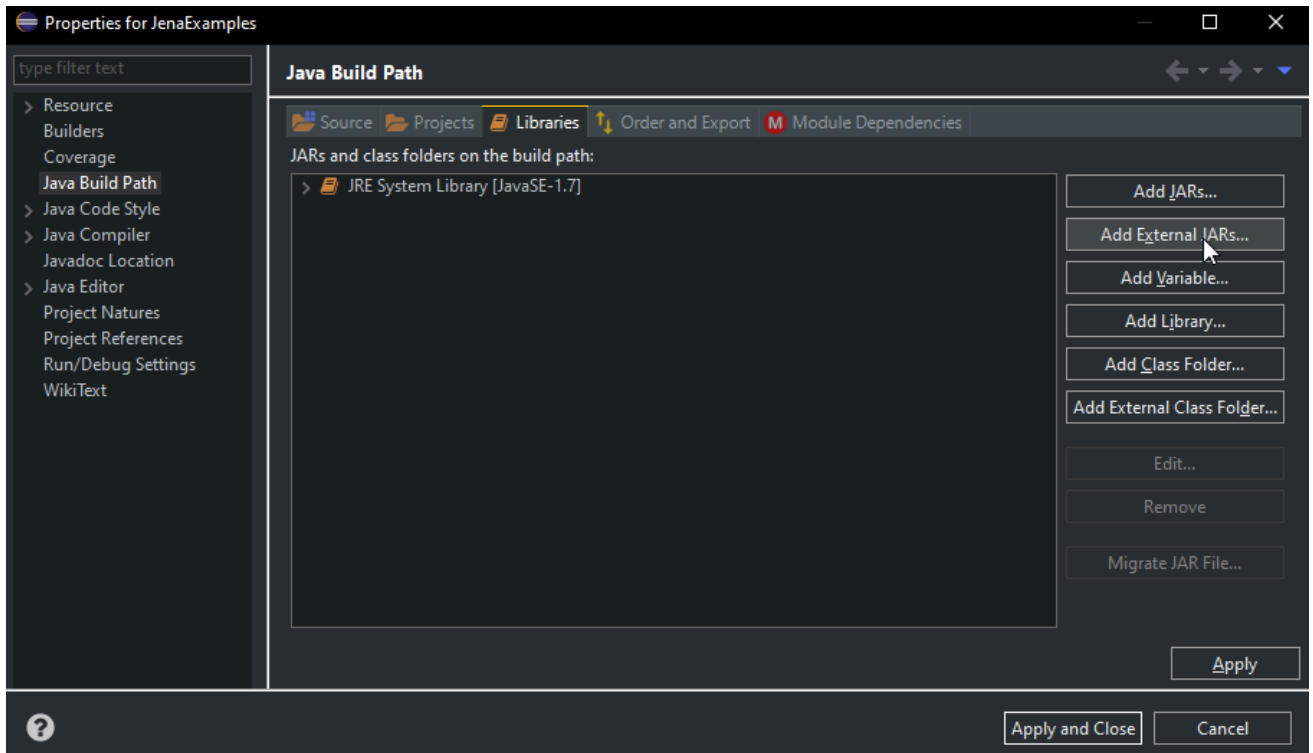
- Eclipse <https://www.eclipse.org/downloads/> (choisir Eclipse IDE pour développeurs Java)
- Java (au moins JSE 1.8) - pas nécessaire sauf si vous installez Eclipse sans JRE
- Jena

Il existe la possibilité d'installer Jena via Maven, de toute façon la méthode que je conseille c'est à partir des librairies officielles:

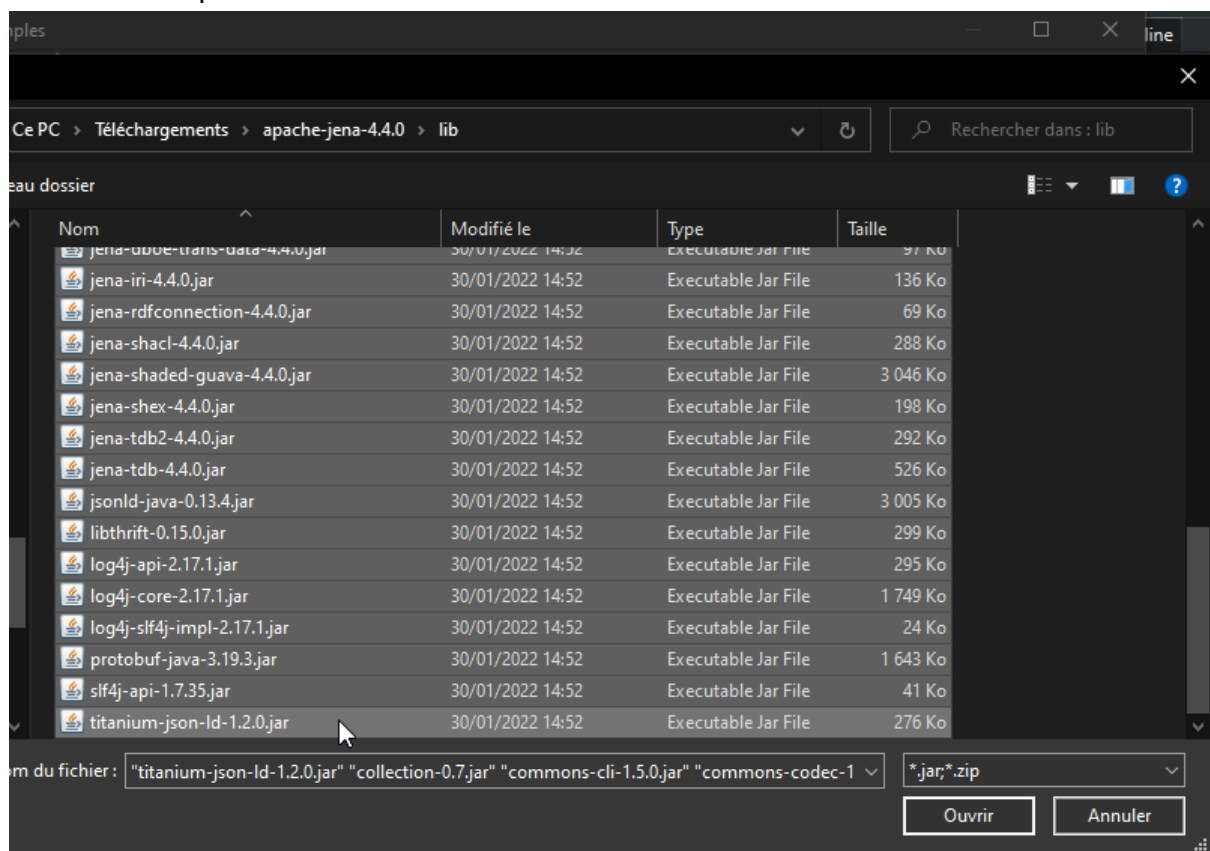
1. Aller sur <https://jena.apache.org/download/index.cgi>
2. Télécharger les binaires [apache-jena-4.5.0.zip](#) (ou .tar.gz)
3. Décompresser le .zip dans un répertoire de votre choix
4. Pour l'installation dans Eclipse, créer un nouveau projet Java avec Eclipse en sélectionnant *File -> New -> Java Project*
5. Donnez un nom de projet et complétez la création du projet (conseil: rester en compatibilité Java 1.8)
6. Sur le nom du projet, faire clic avec le bouton de droite. La fenêtre suivante devrait apparaître:



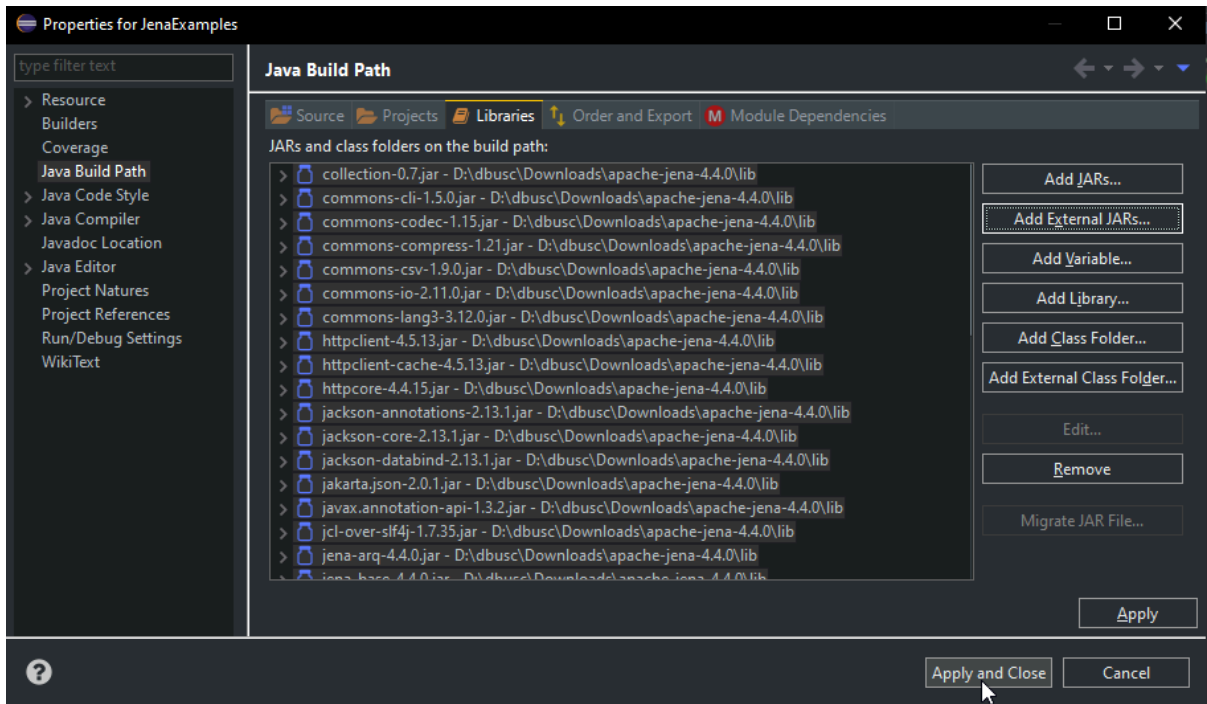
7. Choisir “Properties”, tout en bas. Allez dans l’onglet “libraries”:



8. Cliquer sur “Add External JARS” et sélectionnez tous les fichiers dans le sous-répertoire **lib** du dossier où vous avez versé les contenus du file Jena



9. Cliquer sur “Ouvrir”. Maintenant vous devriez avoir la situation suivante:

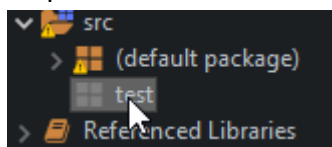


10. Cliquer sur “Apply and close”. Félicitations! Vous avez installé Jena et vous êtes prêts à créer votre premier programme Jena

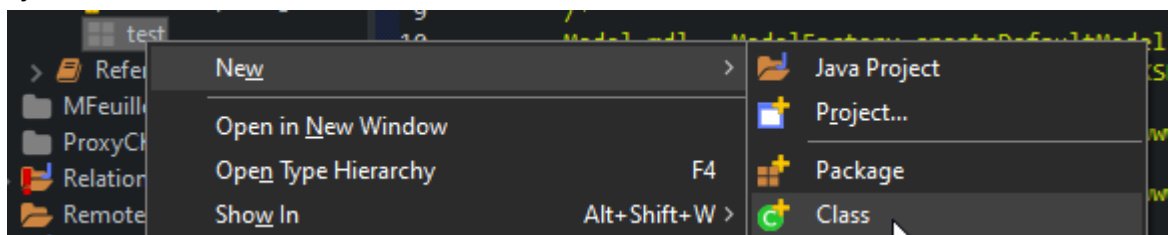
B. Tester le fonctionnement de l'API

Nous allons écrire un petit programme pour créer un graphe RDF et insérer un triplet.

1. Faire clic avec le bouton de droite sur le nom de votre projet
2. Choisir “New package”, qu'on va nommer “test”
3. Cliquez à nouveau avec le bouton de droite sur le package test:



4. Ajouter une nouvelle classe



nommée TestTriple (cochez sur la case pour créer un **main**):

Name:

Modifiers: ☐ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?

☒ `public static void main(String[] args)`

☐ Constructors from superclass

☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

5. Et finalement cliquez sur "Finish". Vous êtes prêts à rentrer du code:

```
package test;

import org.apache.jena.rdf.model.*;
import org.apache.jena.datatypes.xsd.XSDDatatype;

public class TestTriple {

    public static void main(String[] args) {
        Model m = ModelFactory.createDefaultModel();
        String ns = "http://example.com/test#";
        Resource r = m.createResource(ns+"r");
        Property p = m.createProperty(ns+"p");
        r.addProperty(p, "HelloWorld", XSDDatatype.XSDstring);
        m.write(System.out, "turtle");
    }

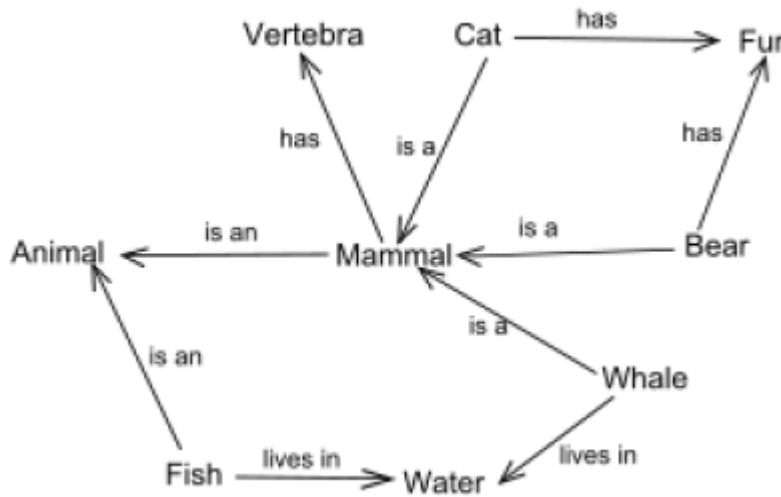
}
```

En exécutant le code, le résultat devrait être le suivant:

```
Problems Javadoc Declaration Console Progress
<terminated> TestTriple [Java Application] C:\Users\dbusc\p2\pool\p
<http://example.com/test#r>
  <http://example.com/test#p> "HelloWorld" .
```

C. Création d'un graphe de connaissances

Nous allons utiliser Jena pour créer un graphe de connaissances.
On considère le graphe suivant:



pour la relation is_a vous pouvez utiliser les predicate **rdfs:subClassOf**
pour le namespace vous pouvez choisir le namespace de votre choix ou comme dans
l'exemple suivant <http://www.example.org#>

@prefix rdfs: <<http://www.w3.org/2000/01/rdf-schema#>> .

@prefix : <<http://www.example.org#>> .

1. Ecrivez le code nécessaire pour créer toutes les ressources et propriétés du graphe
2. Créer les Statements nécessaires pour représenter le graphe
3. Afficher le code Turtle du graphe que vous avez créé
4. Sauvegarder le graphe dans un fichier nommé "animaux.ttl"

D. Navigation du graphe

1. Ecrivez un programme qui compte le nombre de triples (Statements) d'un graphe
2. Calculer ce résultat sur le graphe "animaux.ttl"
3. Lister toutes les triples où le sujet est "Fish"
4. En utilisant l'API SPARQL (ARQ), afficher les triples ayant comme relation "has" et objet "Fur"

E. RDFlib

1. Charger le fichier produit au point C.4
2. Implémenter les points D.1, D.2 et D.3 en utilisant l'API RDFlib en Python
3. Afficher toutes les triples avec relation "lives_in" et objet "Water"