Université Abderrahmane Mira de Bejaia
Faculté des Sciences Exactes
Département d'informatique
Année Universitaire : 2023/2024

2ème Année Master SIA - GL (S3)

Module : **Services Web**
Ontologies et Services Web

TP 9 : Création et manipulation d'une ontologie

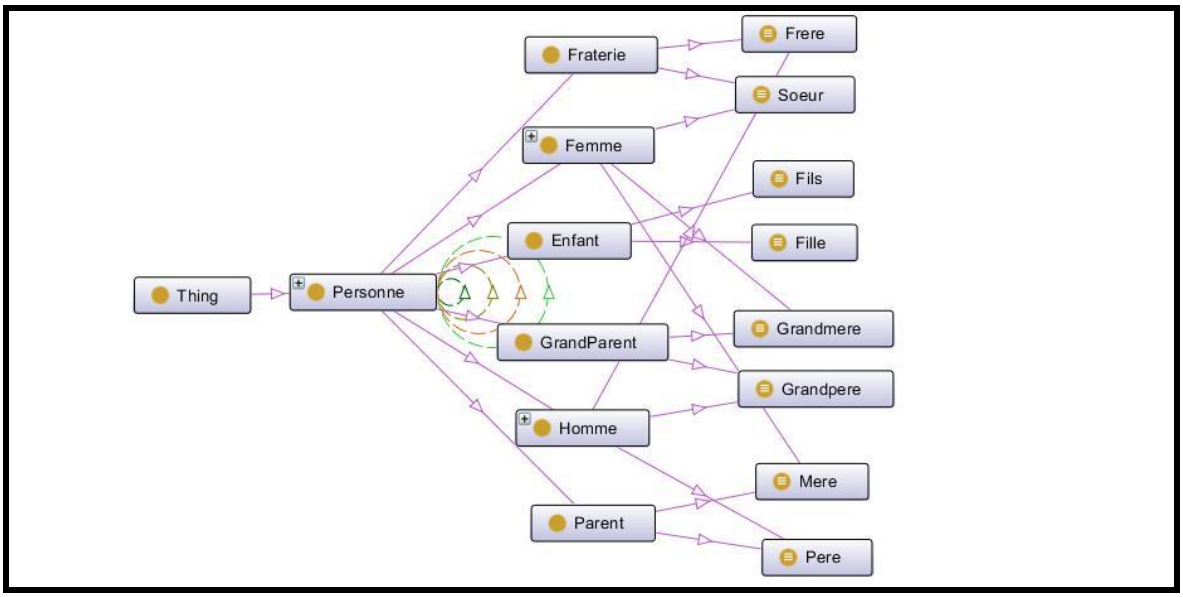
Objectifs

Implémentation et manipulation d'une ontologie et découverte des outils appropriés.

Outils : Protégé (https://protege.stanford.edu/)

Activité 1 : Création de l'ontologie

1. Création des classes : Créer les classes et sous-classes de l'ontologie 'Famille' selon la figure suivante :



2. Création des propriétés pour les classes

Une personne possède un nom, un âge et une nationalité :

a. Créer la propriété (type Datatype) *nom* avec le domaine **Personne** et le range xsd:String
b. Créer la propriété (type Datatype) *age* avec le domaine **Personne** et le range xsd:int
c. Créer la propriété (type Datatype) *nationalite* avec le domaine Personne et le range xsd:String

Deux personnes peuvent se marier

d. Créer la propriété (type Object) *se_marier_avec* avec le domaine **Personne** et le range **Personne**

Une personne est le parent d'une autre personne.

e. Créer la propriété (type Object) estParentDe avec le domaine **Personne** et le range **Personne**

Un homme est le père d'une personne

f. Créer la propriété (type Object) *estPereDe* qui est la sous propriété de estParentDe avec le domaine **Homme** et le range **Personne**

Une femme est la mère d'une personne

g. Créer la propriété (type Object) *estMereDe* qui est la sous propriété de estParentDe avec le domaine **Femme** et le range **Personne**

   Une personne appartient à la fraterie d'une autre personne

h. Créer la propriété (type Object) *estEnRelationDeFraterieAvec* avec le domaine **Personne** et le range **Personne**

   Un homme est le frère d'une personne

i. Créer la propriété (type Object) *estFrereDe* qui est la sous propriété estEnRelationDeFraterieAvec avec le domaine **Homme** et le range **Personne**

   Une femme est la sœur d'une personne

j. Créer la propriété (type Object) *estSoeurDe* qui est la sous propriété *estEnRelationDeFraterieAvec* avec le domaine **Femme** et le range **Personne**

   Une personne est un enfant d'une autre personne

k. Créer la propriété (type Object) estEnfantDe avec le domaine **Personne** et le range **Personne**

   Un homme est le fils d'une personne

l. Créer la propriété (type Object) *estFilsDe* qui est la sous propriété *estEnfantDe* avec le domaine **Homme** et le range **Personne**

   Une femme est la fille d'une personne

m. Créer la propriété (type Object) *estFilleDe* qui est la sous propriété *estEnfantDe* avec le domaine **Femme** et le range **Personne**

3. Création des restrictions sur les classes et propriétés

- La classe Pere a la restriction : La valeur de la propriété *estPereDe* a au moins une instance
- La classe Mere a la restriction : La valeur de la propriété *estMereDe* a au moins une instance
- La classe Fils a la restriction : La valeur de la propriété *estFilsDe* a au moins une instance
- La classe Fille a la restriction : La valeur de la propriété *estFilleDe* a au moins une instance
- La classe Frere a la restriction : La valeur de la propriété *estFrereDe* a au moins une instance
- La classe Sœur a la restriction : La valeur de la propriété *estSoeurDe* a au moins une instance
- Homme et Femme sont disjointes
- Père et Mère sont disjointes
- Fils et Fille sont disjointes
- Grand père et Grand mère sont disjointes

4. Assigner les types pour les propriétés
   - La propriété estEnRelationDeFraterieAvec est transitive
   - La propriété estEnfantDe est la propriété inverse de la propriété estParentDe
   - La propriété nom, age, nationalite sont functionnelle

5. Assigner des instances

   Création des instances pour la classe Homme :

   a. Mohamed, 70, de nationalité tunisienne.
   b. Omar, 40, de nationalité tunisienne.
   c. Ali, 38
   d. Khaled, 45, de nationalité Algérienne.
   e. Nader, 10, de nationalité Algérienne.

f. Zied, 10.
g. Jamel, 5.

Création des instances pour la classe Femme :

a. Aycha, 69, de nationalité tunisienne.
b. Sonia, 30, de nationalité tunisienne.
c. Fatima, 18.
d. Ameni, 5, de nationalité tunisienne.
e. Manel, 25.

Réglage les instances pour les deux classes Homme et Femme :

a. Mohamed se_marier_avec Aycha.
b. Omar estFilsDe Mohamed.
c. Ali estFilsDe Mohamed.
d. Nader estFilsDe Khaled.
e. Zied estFilsDe Omar et Manel.
f. Jamel estFilsDe Omar et Manel.
g. Sonia estFilleDe Aycha.
h. Fatima estFilleDe Aycha et Mohamed.
i. Sonia se_marier_avec Khaled.
j. Ameni estFilleDe Sonia.
k. Manel se_marier_avec Omar.

Activité 2 : Le langage de requête SPARQL, le langage de règles en Jena (Exécute des requêtes avec OWL et crée les règles d'inférence)

1. Lister toutes les instances de la classe Homme, pour chaque instance afficher son nom.

```
PREFIX ns: <http://www.owl-ontologies.com/test_famille.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?homme ?nom
WHERE {
        ?homme rdf:type ns:Homme .
    ?homme ns:nom ?nom .
}
```

2. Lister toutes les instances de la classe Femme, pour chaque instance afficher son âge.

```
PREFIX ns: <http://www.owl-ontologies.com/test_famille.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?femme ?age
WHERE {
        ?femme rdf:type ns:Femme .
    ?femme ns:age ?age .
}
```

3. La liste des personnes (limitée à 3) a un âge supérieur a 20

```
PREFIX ns: <http://www.owl-ontologies.com/test_famille.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?personne ?age
```

```
WHERE{
        ?personne rdf:type ns:Personne .
        ?personne ns:age ?age .
        FILTER (?age >20 )
}
ORDER BY ( ?age)
LIMIT 3
```

4. Lister toutes les instances de la classe Père.

```
PREFIX ns: <http://www.owl-ontologies.com/test_famille.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?pere
WHERE {
        ?pere rdf:type ns:Pere .

}
```

- Règles d'inférence :

-Régle 1 : A de type homme, alors A est un Personne.
-Régle 2 : A de type femme, alors A est un Personne.
-Régle 3 :(Si A de type femme, A estFilleDe B, B de type Homme) et (Si C de type homme, C estFilsDe D, D est Homme) alors B et D sont des pères.

```
 @include <OWLMicro>.
@prefix ns: <http://www.owl-ontologies.com/test_famille.owl#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.

[rule1: (?A rdf:type ns:Homme) -> (?A rdf:type ns:Personne)]
[rule2: (?A rdf:type ns:Femme) -> (?A rdf:type ns:Personne)]
[rule3: (?A rdf:type ns:Personne) (?A ns:estFilleDe ?B) (?B rdf:type ns:Homme) (?K rdf:type ns:Personne) (?k ns:estFilsDe ?F) (?F rdf:type ns:Homme) -> (?B rdf:type ns:Pere) (?F rdf:type ns:Pere)]
```

5. Lister toutes les instances de la classe Mère.

```
PREFIX ns: <http://www.owl-ontologies.com/test_famille.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?mere
WHERE {
        ?mere rdf:type ns:Mere .

}
```

6. Règles d'inférence :

-Régle4 : (Si A est une femme, A estFilleDe B, B de type Femme) et (Si C est un homme, C estFilsDe D, D est Femme) alors B et D sont des mères.

```
 [rule4: (?A rdf:type ns:Femme) (?A ns:estFilleDe ?B) (?B rdf:type ns:Femme) (?K rdf:type ns:Homme) (?k ns:estFilsDe ?F) (?F rdf:type ns:Femme) -> (?B rdf:type ns:Mere) (?F
```

```
rdf:type ns:Mere)]
```

7. Lister toutes les instances de la classe Grand Père.

```
PREFIX ns: <http://www.owl-ontologies.com/test_famille.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?grandpere
WHERE {
        ?grandpere rdf:type ns:GrandPere .
}
```

- Règles d'inférence :

-Régle 5 : A de type Personne, B estFilsDe A, alors B estEnfantDe A.

-Régle 6 : A de type Personne, B estFilleDe A, alors B estEnfantDe A.

-Régle 7 : A de type Personne, A estEnfantDe B, C de type Homme, B estEnfantDe C, alors C est un Grand Pére et l'un de Grand Parent.

-Régle 8 : A de type Personne, A estEnfantDe B, C de type Femme, B estEnfantDe C, alors C est un Grand Père et l'un de Grand Parent.

```
[rule5: (?A rdf:type ns:Personne) (?B ns:estFilsDe ?A) -> (?B ns:estEnfantDe ?A)]
[rule6: (?A rdf:type ns:Personne) (?B ns:estFilleDe ?A) -> (?B ns:estEnfantDe ?A)]
[rule7: (?A rdf:type ns:Personne) (?A ns:estEnfantDe ?B) (?C rdf:type ns:Homme) (?B
ns:estEnfantDe ?C) -> (?C rdf:type ns:GrandPere) (?C rdf:type ns:GrandParent)]
[rule8: (?A rdf:type ns:Personne) (?A ns:estEnfantDe ?B) (?C rdf:type ns:Femme) (?B
ns:estEnfantDe ?C) -> (?C rdf:type ns:GrandMere) (?C rdf:type ns:GrandParent)]
```

8. Lister toutes les instances de la classe Grand Père.

```
PREFIX ns: <http://www.owl-ontologies.com/test_famille.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?grandmere
WHERE {
        ?grandmere rdf:type ns:GrandMere .
}
```

- Règle d'inférence :

-Régle 8 : A de type Personne, A estEnfantDe B, C de type Femme, B estEnfantDe C, alors C est un Grand Père et l'un de Grand Parent.

```
[rule8: (?A rdf:type ns:Personne) (?A ns:estEnfantDe ?B) (?C rdf:type ns:Femme) (?B
ns:estEnfantDe ?C) -> (?C rdf:type ns:GrandMere) (?C rdf:type ns:GrandParent)]
```

9. Lister toutes les instances de la classe Frere.

```
PREFIX ns: <http://www.owl-ontologies.com/test_famille.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?frere
WHERE {
        ?frere rdf:type ns:Frere .
}
```

- Règle d'inférence :

-Régle 9 : A estEnfantDe P, B estEnfantDe P, A est différent de B, A de type Homme, alors A est frère B et B est un frère.

```
[rule9: (?A ns:estEnfantDe ?P) (?B ns:estEnfantDe ?P) notEqual(?A, ?B) (?A rdf:type ns:Homme)-> (?A ns:estFrereDe ?B) (?A rdf:type ns:Frere)]
```

10. Lister toutes les instances de la classe Soeur.

```
PREFIX ns: <http://www.owl-ontologies.com/test_famille.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?soeur
WHERE {
        ?soeur rdf:type ns:Soeur .
}
```

- Règle d'inférence :

-Régle 10: A estEnfantDe P, B estEnfantDe P, A est différent de B, A de type Femme, alors A est soeur B et B est un soeur.

```
[rule10: (?A ns:estEnfantDe ?P) (?B ns:estEnfantDe ?P) notEqual(?A, ?B) (?A rdf:type ns:Femme)-> (?A ns:estSoeurDe ?B) (?A rdf:type ns:Soeur)]
```

Activité 3 : Programmation avec Jena (Ecrire un programme en Java permettant de faire de l'inférence et d'écrire des requêtes avec OWL)

1. Lancer Eclipse
2. Créer un projet « Java Project »
3. Créer un dossier **data** dans le projet (Right click -> New -> Folder)
4. Créer le fichier **rules.txt** dans le répertoire **data** avec le contenu décrit ci-dessous
5. Créer le fichier **query.txt** dans le répertoire **data** avec le contenu décrit ci-dessous
6. Copier puis coller le fichier **test_famille.owl** que vous avez crée dans le répertoire **data**
7. Créer un package **itsudparis.tools** dans le répertoire **src** dans le projet
8. Ouvrir le menu **Projet/Properties/Java Build Path.** Dans la partie **Libraries**, Cliquer sur **Add External JARS**. Ajouter *toutes les librairies de Jen*a (les librairies de Jena se trouvent dans le lib de Jena).
9. Créer une classe **FileTool** avec le contenu décrit ci-dessous dans le package **itsudparis.tools**
10. Exécuter une classe **JenaEngine** avec le contenu décrit ci-dessous dans le package **itsudparis.tools**
11. Créer un package **itsudparis.application**
12. Créer une class **Main** avec le contenu décrit ci-dessous dans le package **itsudparis.application**
13. Exécuter la classe **Main**

Fichier FileTool.java

```
package itsudparis.tools;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
public class FileTool {
   static public String getContents(File aFile) {
      //...checks on aFile are elided
      StringBuilder contents = new StringBuilder();
      try {
```

```
        //use buffering, reading one line at a time
        //FileReader always assumes default encoding is OK!
        BufferedReader input = new BufferedReader(new FileReader(aFile));
        try {
          String line = null; //not declared within while loop
                /*
           * readLine is a bit quirky :
           * it returns the content of a line MINUS the newline.
           * it returns null only for the END of the stream.
           * it returns an empty String if two newlines appear in a row.
           */
          while ((line = input.readLine()) != null) {
            contents.append(line);
            contents.append(System.getProperty("line.separator"));
          }
        } finally {
          input.close();
        }
      } catch (IOException ex) {
        ex.printStackTrace();
      }
      return contents.toString();
    } }
```

Fichier JenaEngine.java

```
package itsudparis.tools;

import java.io.File;
import java.io.InputStream;
import java.util.List;

import com.hp.hpl.jena.query.Query;
import com.hp.hpl.jena.query.QueryExecution;
import com.hp.hpl.jena.query.QueryExecutionFactory;
import com.hp.hpl.jena.query.QueryFactory;
import com.hp.hpl.jena.query.ResultSet;
import com.hp.hpl.jena.query.ResultSetFormatter;
import com.hp.hpl.jena.rdf.model.*;
import com.hp.hpl.jena.reasoner.rulesys.GenericRuleReasoner;
import com.hp.hpl.jena.reasoner.rulesys.Rule;
import com.hp.hpl.jena.util.FileManager;
import java.io.IOException;
import java.io.OutputStream;
public class JenaEngine {
   static private String RDF = "http://www.w3.org/1999/02/22-rdf-syntax-ns#";

   static public Model readModel(String inputDataFile) {
//                  create an empty model
      Model model = ModelFactory.createDefaultModel();
      // use the FileManager to find the input file
      InputStream in = FileManager.get().open(inputDataFile);
      if (in == null) {
         System.out.println("Ontology file: " + inputDataFile + " not found");
         return null;
```

```
        }
        // read the RDF/XML file
        model.read(in, "");
        try {
            in.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            return null;
        }
        return model;
    }
    /**
     * Faire l'inference
     * @param args
     * + Entree: l'objet model Jena avec le chemin du fichier de regles
     * + Sortie: l'objet model infere Jena
     */
    static public Model readInferencedModelFromRuleFile(Model model, String
inputRuleFile) {
        InputStream in = FileManager.get().open(inputRuleFile);
        if (in == null) {
            System.out.println("Rule File: " + inputRuleFile + " not found");
            return null;
        } else {
            try {
                in.close();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                return null;
            }
        }

        List rules = Rule.rulesFromURL(inputRuleFile);
        GenericRuleReasoner reasoner = new GenericRuleReasoner(rules);
        reasoner.setDerivationLogging(true);
        reasoner.setOWLTranslation(true);            // not needed in RDFS case
        reasoner.setTransitiveClosureCaching(true);
        InfModel inf = ModelFactory.createInfModel(reasoner, model);
        return inf;
    }


    /**
     * Executer une requete
     * @param args
     * + Entree: l'objet model Jena avec une chaine des caracteres SparQL
     * + Sortie: le resultat de la requete en String
     */
    static public String executeQuery(Model model, String queryString) {
        Query query = QueryFactory.create(queryString);
        // No reasoning
        // Execute the query and obtain results
        QueryExecution qe = QueryExecutionFactory.create(query, model);
        ResultSet results = qe.execSelect();
```

```
        OutputStream output = new OutputStream() {

            private StringBuilder string = new StringBuilder();

            @Override
            public void write(int b) throws IOException {
                this.string.append((char) b);
            }

            //Netbeans IDE automatically overrides this toString()
            public String toString() {
                return this.string.toString();
            }
        };

        ResultSetFormatter.out(output, results, query);
        return output.toString();
    }
    /**
     * Executer un fichier d'une requete
     * @param args
     * + Entree: l'objet model Jena avec une chaine des caracteres SparQL
     * + Sortie: le resultat de la requete en String
     */
    static public String executeQueryFile(Model model, String filepath) {
        File queryFile = new File(filepath);
        // use the FileManager to find the input file
        InputStream in = FileManager.get().open(filepath);
        if (in == null) {
            System.out.println("Query file: " + filepath + " not found");
            return null;
        } else {
            try {
                in.close();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                return null;
            }
        }
        String queryString = FileTool.getContents(queryFile);
        return executeQuery(model, queryString);
    }
    /**
     * Executer un fichier d'une requete avec le parametre
     * @param args
     * + Entree: l'objet model Jena avec une chaine des caracteres SparQL
     * + Sortie: le resultat de la requete en String
     */
    static public String executeQueryFileWithParameter(Model model, String filepath, String
parameter) {
        File queryFile = new File(filepath);
        // use the FileManager to find the input file
        InputStream in = FileManager.get().open(filepath);
```

```
        if (in == null) {
            System.out.println("Query file: " + filepath + " not found");
            return null;
        } else {
            try {
                in.close();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                return null;
            }
        }
        String queryString = FileTool.getContents(queryFile);
        queryString = queryString.replace("%PARAMETER%", parameter);
        return executeQuery(model, queryString);
    }
     /**
     * Creer une Instance
     * @param args
     * + Entree:
     *      - l'objet model Jena
     *      - Namespace de l'ontologie
     *      - Le nom de le classe
     *      - Le nom de l'instance
     * + Sortie: le resultat de la requete en String
     */
    static public boolean createInstanceOfClass(Model model, String namespace, String
className, String instanceName) {
        Resource rs = model.getResource(namespace + instanceName);
        if (rs == null)
            rs = model.createResource(namespace+instanceName);
        Property p = model.getProperty(RDF + "type");
        Resource rs2 = model.getResource(namespace + className);
        if ((rs2 != null)&&(rs != null) && (p != null)) {
            //add new value
            rs.addProperty(p,rs2);
            return true;
        }
        return false;
    }
    /**
     * Mettre a jour la valeur d'une propriete objet d'une instance
     * @param args
     * + Entree:
     *      - l'objet model Jena
     *      - Namespace de l'ontologie
     *      - Le nom de la première Instance
     *      - Le nom de la propriete
     *      - Le nom de la deuxieme Instance
     * + Sortie: le resultat de la requete en String
     */
    static public boolean updateValueOfObjectProperty(Model model, String namespace,
String object1Name, String propertyName, String object2Name) {
        Resource rs1 = model.getResource(namespace + object1Name);
        Resource rs2 = model.getResource(namespace + object2Name);
```

```
        Property p = model.getProperty(namespace + propertyName);
        if ((rs1 != null) && (rs2 != null) && (p != null)) {
            //remove all old values of property p
            rs1.removeAll(p);
            //add new value
            rs1.addProperty(p,rs2);
            return true;
        }
        return false;
    }
    /**
     * Mettre a jour la valeur d'une propriete objet d'une Instance
     * @param args
     * + Entree:
     *      - l'objet model Jena
     *      - Namespace de l'ontologie
     *      - Le nom de la premiere Instance
     *      - Le nom de la propriete
     *      - Le nom de le deuxieme Instance
     * + Sortie: le resultat de la requete en String
     */
    static public boolean addValueOfObjectProperty(Model model, String namespace, String
instance1Name, String propertyName, String instance2Name) {
        Resource rs1 = model.getResource(namespace + instance1Name);
        Resource rs2 = model.getResource(namespace + instance2Name);
        Property p = model.getProperty(namespace + propertyName);
        if ((rs1 != null) && (rs2 != null) && (p != null)) {
            //add new value
            rs1.addProperty(p,rs2);
            return true;
        }
        return false;
    }

    /**
     * Mettre a jour la valeur d'une propriete datatype d'une Instance
     * @param args
     * + Entree:
     *      - l'objet model Jena
     *      - Namespace de l'ontologie
     *      - Le nom de l'Instance
     *      - Le nom de la propriete
     *      - La nouvelle valeur
     * + Sortie: le resultat de la requete en String
     */
    static public boolean updateValueOfDataTypeProperty(Model model, String namespace,
String instanceName, String propertyName, Object value) {
        Resource rs = model.getResource(namespace + instanceName);
        Property p = model.getProperty(namespace + propertyName);
        if ((rs != null) && (p != null)) {
            //remove all old values of property p
            rs.removeAll(p);
            //add new value
            rs.addLiteral(p, value);
```

```
            return true;
        }
        return false;
    }
    /**
     * Ajouter la valeur d'une propriete datatype d'une Instance
     * @param args
     * + Entree:
     *      - l'objet model Jena
     *      - Namespace de l'ontologie
     *      - Le nom de l'Instance
     *      - Le nom de la propriete
     *      - La valeur
     * + Sortie: le resultat de la requete en String
     */
    static public boolean addValueOfDataTypeProperty(Model model, String namespace,
String instanceName, String propertyName, Object value) {
        Resource rs = model.getResource(namespace + instanceName);
        Property p = model.getProperty(namespace + propertyName);
        if ((rs != null) && (p != null)) {
            //add new value
            rs.addLiteral(p, value);
            return true;
        }
        return false;
    }
    /**
     * Supprimer toutes les valeurs d'une propriete d'une Instance
     * @param args
     * + Entree:
     *      - l'objet model Jena
     *      - Namespace de l'ontologie
     *      - Le nom de l'Instance
     *      - Le nom de la propriete
     * + Sortie: le resultat de la requete en String
     */
    static public boolean removeAllValuesOfProperty(Model model, String namespace,
String objectName, String propertyName) {
        Resource rs = model.getResource(namespace + objectName);
        Property p = model.getProperty(namespace + propertyName);
        if ((rs != null) && (p != null)) {
            //remove all old values of property p
            rs.removeAll(p);
            //add new value
            return true;
        }
        return false;
    }

}
```

Fichier Main.java

```
package itsudparis.application;
```

```
import com.hp.hpl.jena.rdf.model.Model;
import itsudparis.tools.JenaEngine;
public class Main {
	/**
	 * @param args
	 *		the command line arguments
	 */
	public static void main(String[] args) {
		String NS = "";
		// lire le model a partir d'une ontologie
		Model model = JenaEngine.readModel("data/test_famille.owl");
		if (model != null) {
			//lire le Namespace de l'ontologie
			NS = model.getNsPrefixURI("");

			// apply our rules on the owlInferencedModel
			Model inferedModel =
JenaEngine.readInferencedModelFromRuleFile(model, "data/rules.txt");
			// query on the model after inference
		   System.out.println(JenaEngine.executeQueryFile(inferedModel,
				"data/query.txt"));
		} else {
			System.out.println("Error when reading model from ontology");
		}
	}
}
```

Fichier rules.txt

```
@include <OWLMicro>.
```

Fichier query.txt

```
PREFIX ns: <http://www.owl-ontologies.com/test_famille.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?personne
WHERE {
	?personne rdf:type ns:Personne .
}
```