**Code for Ontology Application**

**package** test;

**import** java.io.FileInputStream;

**import** tutorial.JenaTutorialExamples;

**import** com.franz.agraph.jena.AGGraphMaker;

**import** com.franz.agraph.jena.AGInfModel;

**import** com.franz.agraph.jena.AGModel;

**import** com.franz.agraph.jena.AGReasoner;

**import** com.franz.agraph.repository.AGCatalog;

**import** com.franz.agraph.repository.AGRepositoryConnection;

**import** com.franz.agraph.repository.AGServer;

**import** com.franz.agraph.repository.AGVirtualRepository;

**import** com.hp.hpl.jena.ontology.OntModel;

**import** com.hp.hpl.jena.ontology.OntModelSpec;

**import** com.hp.hpl.jena.rdf.model.InfModel;

**import** com.hp.hpl.jena.rdf.model.ModelFactory;

**import** com.hp.hpl.jena.rdf.model.Property;

**import** com.hp.hpl.jena.rdf.model.RDFNode;

**import** com.hp.hpl.jena.rdf.model.Resource;

**import** com.hp.hpl.jena.rdf.model.StmtIterator;

**import** com.hp.hpl.jena.vocabulary.OWL;

**import** com.hp.hpl.jena.vocabulary.RDF;

**public** **class** AGMoreJenaExamples **extends** JenaTutorialExamples {

**public** **static** **void** addModelToAGModel() **throws** Exception {

     AGGraphMaker maker = example1(**false**);

  AGModel agmodel = **new** AGModel(maker.createGraph());

  OntModel model = ModelFactory.createOntologyModel(**new** OntModelSpec(OntModelSpec.OWL\_DL\_MEM));

        model.read(**new** FileInputStream("src/tutorial/java-lesmis.rdf"), **null**);

        println("Read " + model.size() + " lesmis.rdf triples.");

  agmodel.add(model);

  println("Add yields " + agmodel.size() + " triples in agmodel.");

 }

**public** **static** AGGraphMaker exampleMakerForRestrictionReasoningRepository() **throws** Exception {

  AGServer server = **new** AGServer(SERVER\_URL,USERNAME,PASSWORD);

  AGCatalog catalog = server.getCatalog(CATALOG\_ID);

  catalog.deleteRepository(REPOSITORY\_ID);

  catalog.createRepository(REPOSITORY\_ID);

  String spec = AGVirtualRepository.reasoningSpec("<"+CATALOG\_ID+":"+REPOSITORY\_ID+">", "restriction");

  AGVirtualRepository repo = server.virtualRepository(spec);

  AGRepositoryConnection conn = repo.getConnection();

**return** **new** AGGraphMaker(conn);

 }

**public** **static** **void** exampleRestrictionReasoning() **throws** Exception {

  //AGGraphMaker maker = exampleMakerForRestrictionReasoningRepository();

  AGGraphMaker maker = example1(**false**);

  AGModel model = **new** AGModel(maker.getGraph());

  AGReasoner reasoner = AGReasoner.RESTRICTION;

  InfModel infmodel = **new** AGInfModel(reasoner, model);

  Resource a = model.createResource("http://a");

  Resource c = model.createResource("http://C");

  Property p = model.createProperty("http://p");

  Resource v = model.createResource("http://v");

  model.add(c,OWL.equivalentClass,c);

  model.add(c,RDF.type,OWL.Restriction);

  model.add(c,OWL.onProperty,p);

  model.add(c,OWL.hasValue,v);

  model.add(a,RDF.type,c);

  StmtIterator results = infmodel.listStatements();

**while** (results.hasNext()) {

   System.out.println(results.next());

  }

  System.out.println(infmodel.contains(a,RDF.type,c));

  System.out.println(infmodel.contains(a,p,v));

 }

**public** **static** **void** exampleBulkUpdate() **throws** Exception {

  AGGraphMaker maker = example6();

  AGModel model = **new** AGModel(maker.getGraph());

  AGModel model2 = **new** AGModel(maker.createGraph("http://example.org/foo"));

  StmtIterator statements = model.listStatements(**null**,RDF.type, (RDFNode)**null**);

  model2.add(statements);

  System.out.println("Size: "+model2.size());

  model2.write(System.out);

  statements = model.listStatements(**null**,RDF.type, (RDFNode)**null**);

  model2.remove(statements);

  System.out.println("Size: "+model2.size());

  model2.write(System.out);

 }

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

  addModelToAGModel();

  exampleBulkUpdate();

  exampleRestrictionReasoning();

  closeAll();

 }

}