```
package aminePlatform.samples.prologPlusCG;

/*************************************************************************
Amine - Artificial Intelligence platform for the development of Intelligent
Systems.
Copyright (C) 2004 AmineGroup.

GNU Lesser General Public License

This library is free software; you can redistribute it and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation,
version 2.1 of the License.

This library is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public
License along with this library; if not, write to the
Free Software Foundation, Inc., 59 Temple Place - Suite 330,
Boston, MA  02111-1307, USA.
*************************************************************************/

import java.util.*;
import java.io.*;
import javax.swing.JTextArea;

import aminePlatform.guis.util.AmineFrame;
import aminePlatform.util.*;
import aminePlatform.util.cg.*;

import aminePlatform.kernel.ontology.Ontology;
import aminePlatform.kernel.lexicons.Lexicon;
import aminePlatform.engines.prologPlusCG.interpreter.*;
import aminePlatform.engines.prologPlusCG.util.*;
import aminePlatform.engines.prologPlusCG.parser.*;
import
aminePlatform.guis.prologPlusCGGUIs.prologPlusCGGUI.PrologPlusCGGUIFrame;

/**
 * <p>Title : PrologPlusCGFromJava class</p>
 * <p>Description : PrologPlusCGFromJava is a simple example that illustrates
 * how Prolog+CG can be called from a Java class and how requests and reponses
 * can be treated.</p>
 * Copyright:    Copyright (c) Adil KABBAJ, 2004 <p>
 * @author Adil KABBAJ
 * @version 3
 */
public class PrologPlusCGFromJava {
  JTextArea txtArea = new JTextArea();
  Interpreter interpreter;
  Lexicon lexicon;

  public PrologPlusCGFromJava() {
  }
```

```
// We present here an example of a call of Prolog+CG from a Java class.
// The basic steps are illustrated.
void resolve() {
  try {
    // The File Path of the Ontology to use in this example.
    String ontologyFilePath = AmineFrame.ontologyDirPath +
                                                        "ManOntology2.xml";

    // Specify the List of Prolog+CG source files that will be used.
    // In this example, we will use only one file :
                                  /CGPrograms/citizenExple.prlg,
    // So, we construct the full path for the file and we create an array of
    // String with this filePath (as the only element).
    String filePath = PrologPlusCGGUIFrame.ppcgDirPath +
                         System.getProperty("file.separator") +
                         "CGPrograms" + System.getProperty("file.separator")
                          + "citizenExple.prlg";
    String[] ppcgFilePaths = new String[1];
    ppcgFilePaths[0] = filePath;

    // Create an Instance of Prolog+CG Interpreter, with the ontology file
    // path and the array of Prolog+CG file paths. This creation will be
    // concerned by the parsing of all the specified files, in order to be
    // ready for answering requests.
    interpreter = new Interpreter(ontologyFilePath, ppcgFilePaths);

    lexicon = interpreter.getLexicon();
    // Step 7 : Ask the interpreter to resolve the desired request and
    // to return all the solutions. The request is specified in a String.

    // First request
    ArrayList allSolutions =
          interpreter.findAllSolutions(
                      "[CITIZEN : x]<-memberOf-[COUNTRY : Oz].");

    // Do whatever with the solutions. Here we print the result.
    txtArea.setText("");
    HashMap aSolution;
    if (allSolutions != null) {
      for (Iterator i = allSolutions.iterator(); i.hasNext();) {
        aSolution = (HashMap) i.next();
        writeASolution(aSolution);
        txtArea.append("\n");
      }
      System.out.println(txtArea.getText());
    }

    // Second request
    allSolutions = interpreter.findAllSolutions("analogy(s, t1, t2).");
    // Do whatever with the solutions. Here we print the result.
    txtArea.setText("");
    if (allSolutions != null) {
      for (Iterator i = allSolutions.iterator(); i.hasNext();) {
        aSolution = (HashMap) i.next();
        writeASolution(aSolution);
        txtArea.append("\n");
```

```
        }
        System.out.println(txtArea.getText());
      }
    }
    catch (Exception ex) {
      System.out.println("Problem during the call of Prolog+CG from Java : " +
                         ex.getMessage());
    }
  }

  public void write(String message) {
    txtArea.append(message);
  }

  void writeASolution(HashMap aSolution) {
    txtArea.append("{");

    int nbreOfVar = aSolution.size();
    int indxOfLastElem = nbreOfVar - 1;
    int j = 0;
    Variable variable;
    String valOfVar = null;
    Object obj = null;
    for (Iterator i=interpreter.getVarsInRequest().iterator();i.hasNext();) {
      variable = (Variable) i.next();

      try {
        obj = aSolution.get(variable);
        if (obj != null)
          valOfVar = AmineObjects.toString(obj, lexicon);
        else valOfVar = "_FREE";
      }
      catch (Exception ex) {
        ex.printStackTrace();
      }
      if (j != 0 && (CG.isCG(obj) || Concept.isConcept(obj)))
        txtArea.append("\n");
      txtArea.append(variable.toString());
      txtArea.append(" = ");
      txtArea.append(valOfVar);
      if (j != indxOfLastElem) {
        txtArea.append(", ");
        j++;
      }
    }
    write("}");
  }

  public static void main(String[] args) {
    PrologPlusCGFromJava prologPlusCGFromJava1 = new PrologPlusCGFromJava();
    prologPlusCGFromJava1.resolve();
  }
}
```