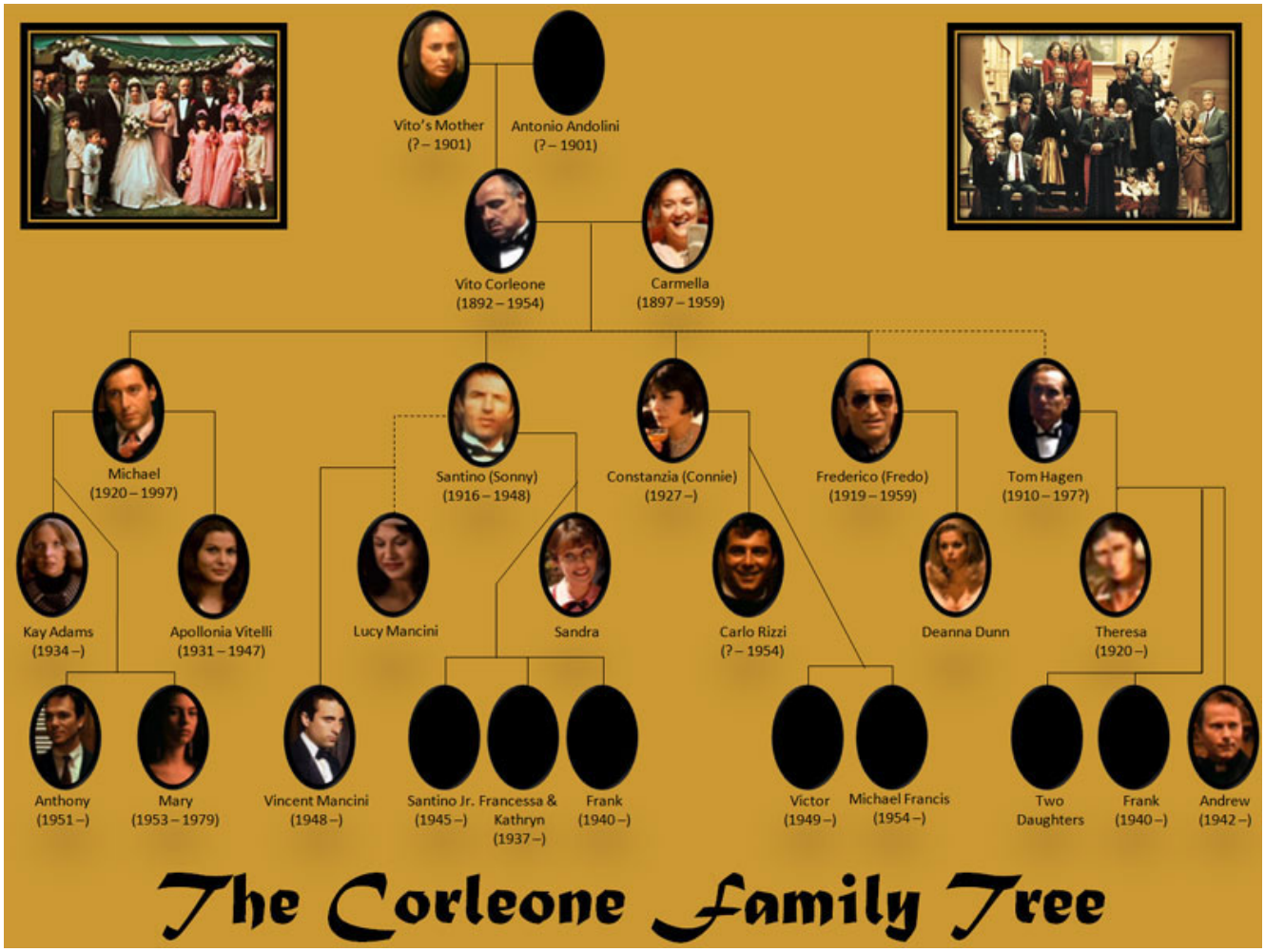
Práctica 3:
Ontologías

Introducción y Objetivos

El objetivo de la práctica es la gestión de datos utilizando ontologías, elemento básico en Sistemas de Información Distribuidos. Se pretende

- Utilizar un lenguaje de ontologías para representar conocimiento
- Utilizar un razonador para realizar consultas a una ontología
- Saber utilizar un editor de ontologías
- Saber utilizar una API para la gestión de ontologías desde aplicaciones

Como ejemplo concreto de aplicación, se propone el dominio de las relaciones familiares y los personajes de “El padrino”, sin perjuicio de que los alumnos puedan proponer aplicaciones alternativas de su interés personal.



Ejercicios

1. Desarrollar una ontología en el lenguaje OWL 2, sintaxis Manchester, utilizando el editor **Protégé** (http://protege.stanford.edu). El IRI de la ontología será `http://sid.cps.unizar.es`. Por simplicidad, supondremos que abuelo, hijo, nieto y sobrino incluyen tanto a personas de género masculino como de femenino. La ontología debe codificar la siguiente información:

a) Todo mafioso es una persona

b) Toda persona es un hombre o una mujer

c) Hombre y mujer son disjuntos

d) Don Corleone es un mafioso

e) Mary, Carmela y Connie son mujeres

f) Vito, Fredo, Michael, Santino y Vincent son hombres

g) Don Corleone y Vito son en el mismo individuo

h) Los demás individuos son diferentes entre sí

i) Vito es marido de Carmela

j) Santino, Connie, Fredo y Michael son hijos de Carmela y Vito

k) Vincent es hijo de Santino

l) Mary es hija de Michael

m) `maridoDe` y `esposaDe` son inversos

n) `maridoDe` y `esposaDe` son subpropiedades de `cónyugeDe`

o) `cónyugeDe` es simétrica

p) Los maridos son hombres

q) Las esposas son mujeres

r) `hijoDe` y `tieneHijo` son inversos

s) La clase progenitor se define como aquellas personas que tienen algún hijo

t) Un padre es un progenitor que es hombre

u) Una madre es un progenitor que es mujer

v) `abueloDe` es la composición de `tieneHijo` y `tieneHijo`, usando la composición de propiedades (con *property chains*)

w) `nietoDe` es inverso de `abueloDe`

x) `nietoDe` e `hijoDe` son subpropiedades de `descendienteDe`

y) Un abuelo es `abueloDe` alguien

2. Desarrollar una aplicación Java que utilice la **OWL API** para modificar la ontología y razonar con ella usando el razonador **HermiT**. Para instalar OWL API, descargar `owlapi-distribution` y `owlapi-osgidistribution` de https://search.maven.org/search? q=g:net.sourceforge.owlapi, y de este último fichero incluir los ficheros `.jar` que hay en su carpeta `lib` como dependencias de nuestro proyecto. HermiT se puede descagar de http://www.hermit-reasoner.com/download.html, basta escomprimir y usar la biblioteca `HermiT.jar`.

El motivo es que este razonador es directamente posible desde Protégé, de manera que es posible comparar los resultados obtenidos llamándolo desde Protégé (a través de la interfaz gráfica, que usará de manera transparente un plug-in para Protégé) y desde nuestra aplicación Java (usando la biblioteca `HermiT.jar`). Puede ser necesario descarga la biblioteca **Guava** y añadirla al proyecto (https://mvnrepository.com/artifact/com.google.guava/guava).

La aplicación permitirá añadir nuevos axiomas y resolverá ciertas consultas. En concreto, se podrá:

a) Añadir individuos a la clase `Mafioso`

b) Mostrar todos los hombres

c) Mostrar todos los mafiosos

d) Mostrar la esposa de Vito

e) Mostrar todos los descendientes de Vito

f) Mostrar el padre de Michael

g) Mostrar la abuela de Vincent

Entrega

La entrega se realizará a través de Moodle hasta las 12 horas del **22 de marzo** y consistirá en un fichero `.zip` que incluya al menos

- La ontología `.owl`,
- las clases `.java` desarrolladas por el alumno,
- un ejecutable `.jar` y
- una brevísima documentación en `pdf` con los nombres de los autores y los aspectos no triviales del trabajo realizado, incluyendo la instalación de las herramientas necesarias.

Apéndice. Plantilla para la OWL API

```
import java.io.*;
import java.util.*;
import org.semanticweb.owlapi.apibinding.*;
import org.semanticweb.HermiT.*;
import org.semanticweb.owlapi.model.*;
import org.semanticweb.owlapi.reasoner.*;

public class P3 {

  public static void main(String[] args) {

    try {
      // Carga ontología
      IRI ontologyIRI = IRI.create("http://sid.cps.unizar.es");
      File file = new File("SIDpractica3.owl");
      OWLOntologyManager manager = OWLManager.createOWLOntologyManager();
      OWLOntology ont = manager.loadOntologyFromOntologyDocument(file);
      OWLDataFactory factory = manager.getOWLDataFactory();


      // Añadimos axiomas de la práctica
      // ...


      // Crear el razonador
      OWLReasoner reasoner = new Reasoner.ReasonerFactory().createReasoner(ont);
      reasoner.precomputeInferences();

      // Ejemplo de consulta: consistencia
      System.out.println(reasoner.isConsistent());


      // Lanzamos consultas de la práctica
      // ...


      // Salvamos la ontología
      manager.saveOntology(ont, IRI.create(file.toURI()));
      System.out.println("Ontología salvada correctamente");
    }
    catch (Exception e) {
      System.err.println("Exception: " + e);
    }
  }
}
```