Jena

Read

- Semantic Web Programming: chapters 3, 2.

This week's exercises will give you the first looks on the Jena API. Use the book to look for examples, but look also at the Jena Framework javadoc[1] , there might be better ways to do things than what is used in the book.

1 Creating RDF models with Jena

In this set of exercises we will learn how to create models with Jena. First, we will create the model "by hand", i.e., writing triples like in last week's exercises, then we will do the same using a Jena java program.

1.1 Exercise

Write an RDF representation of the graph in Figure 1. Add also some of the names of the the resources in the graphs in different languages, e.g., "Cat" spells "Katt" in Norwegian and "Katze" in German.

Use whatever namespace you prefer. The important thing is that you get a good idea of how the file should look like as you will need that in the next exercise.

1.1.1 Solution

I have translated the edges labelled "is a" to `rdfs:subClassOf` predicates. (Don't worry, we'll learn about RDFS later) and otherwise kept the names of the edges and nodes as they are. I have added "Bjørn" and "Bear" as names of the resource `:Bear` using the predicate `rdfs:label`.

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

:Fish   rdfs:subClassOf :Animal ;
 :livesIn        :Water .

:Mammal rdfs:subClassOf :Animal ;
 :has            :Vertebra .

:Whale rdfs:subClassOf :Mammal ;
```

[1] http://incubator.apache.org/jena/documentation/javadoc/jena/

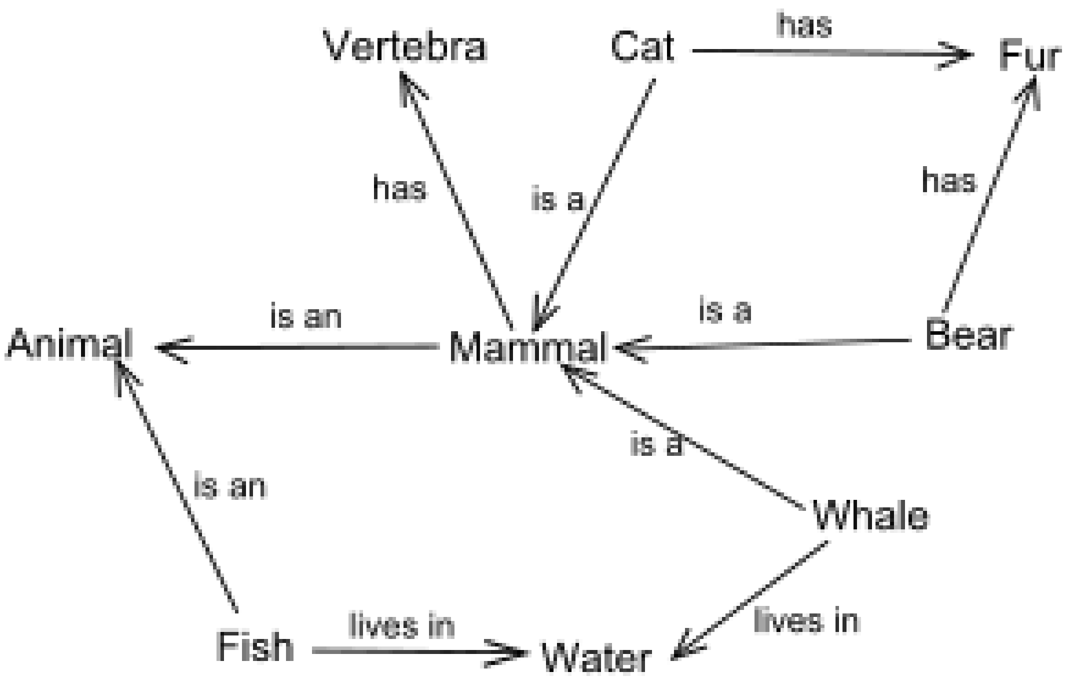

Figure 1: Graphs are suitable for encoding meaning.

```
       :livesIn        :Water .

:Cat   rdfs:subClassOf :Mammal ;
       :has            :Fur .

:Bear  rdfs:subClassOf :Mammal ;
       :has            :Fur ;
       rdfs:label      "Bjørn"@no ;
       rdfs:label      "Bear"@en .
```

1.2 Exercise

Write a java program which creates the same RDF model as you made in the previous exercise and writes the model to a file in the same RDF serialisation language as you used.

1.2.1 Solution

My program is called `RDFAnimals`. The package `com.hp.hpl.jena.vocabulary.RDFS` contains static variables for the RDFS vocabulary. The animal model will be assembled in the variable `model`.

```
import com.hp.hpl.jena.rdf.model.*;
import com.hp.hpl.jena.vocabulary.RDFS;
public class RDFAnimals{
private Model model;
```

`writeModel`. Writes `model` to `System.out` in Turtle (TTL) format.

```
public void writeModel(){
    model.write(System.out, "TTL");
}
```

addStatement takes three strings as arguments, which is assumed to be three URIs which constitute a triple, and creates respectively a subject, a predicate and an object of these strings. The subject, predicate and object is assembled to a statement (triple) and added to the model.

Note that this method assumes that all objects are resources and not literals, which I thought was OK for this example, since the program is custom built to create just this one RDF graph. A proper addStatement method should be more generic than my method.

```
public void addStatement(String s, String p, String o){
  Resource subject = model.createResource(s);
  Property predicate = model.createProperty(p);
  RDFNode object = model.createResource(o);
  Statement stmt = model.createStatement(subject, predicate, object);
  model.add(stmt);
}
```

createModel creates an empty model in the variable model, defines the two namespaces I will be using, and uses the addStatement method to add triples to the model. Finally, "Bjørn" and "Bear" are added as rdfs:label-s to :Bear.

```
public void createModel(){
  model = ModelFactory.createDefaultModel();
  String ns = "http://www.example.org#";
  String nsRDFS = "http://www.w3.org/2000/01/rdf-schema#";

  addStatement(ns+"Fish", nsRDFS+"subClassOf", ns+"Animal");
  addStatement(ns+"Fish", ns+"livesIn", ns+"Water");
  addStatement(ns+"Mammal", nsRDFS+"subClassOf", ns+"Animal");
  addStatement(ns+"Mammal", ns+"has", ns+"Vertebra");
  addStatement(ns+"Whale", nsRDFS+"subClassOf", ns+"Mammal");
  addStatement(ns+"Whale", ns+"livesIn", ns+"Water");
  addStatement(ns+"Cat", nsRDFS+"subClassOf", ns+"Mammal");
  addStatement(ns+"Cat", ns+"has", ns+"Fur");
  addStatement(ns+"Bear", nsRDFS+"subClassOf", ns+"Mammal");
  addStatement(ns+"Bear", ns+"has", ns+"Fur");

  Resource bear = model.getResource(ns+"Bear");
  bear.addProperty(RDFS.label, "Bjørn", "no");
  bear.addProperty(RDFS.label, "Bear", "en");
}
```

main method. Creates the model and writes the model to standard out.

```
  public static void main(String[] args){
    RDFAnimals dave = new RDFAnimals();
    dave.createModel();
    dave.writeModel();
  }
} // end RDFAnimals
```

1.2.2 Result

The result of running my program:

```
<http://www.example.org#Bear>
```

```
      <http://www.w3.org/2000/01/rdf-schema#label>
              "Bear"@en , "Bjørn"@no ;
      <http://www.w3.org/2000/01/rdf-schema#subClassOf>
              <http://www.example.org#Mammal> ;
      <http://www.example.org#has>
              <http://www.example.org#Fur> .

<http://www.example.org#Cat>
      <http://www.w3.org/2000/01/rdf-schema#subClassOf>
              <http://www.example.org#Mammal> ;
      <http://www.example.org#has>
              <http://www.example.org#Fur> .

<http://www.example.org#Whale>
      <http://www.w3.org/2000/01/rdf-schema#subClassOf>
              <http://www.example.org#Mammal> ;
      <http://www.example.org#livesIn>
              <http://www.example.org#Water> .

<http://www.example.org#Fish>
      <http://www.w3.org/2000/01/rdf-schema#subClassOf>
              <http://www.example.org#Animal> ;
      <http://www.example.org#livesIn>
              <http://www.example.org#Water> .

<http://www.example.org#Mammal>
      <http://www.w3.org/2000/01/rdf-schema#subClassOf>
              <http://www.example.org#Animal> ;
      <http://www.example.org#has>
              <http://www.example.org#Vertebra> .
```

2 RDF serialisation converter

RDF may be written in different ways, in different serialisations. The common serialisations are RDF/XML, Turtle and N3.

2.1 Exercise

Write a Java program which can convert between four serialisations of RDF: RDF/XML, Turtle, N3 and N-TRIPLE. The program should take two arguments, input file and output format, where the file extension of the input file indicate which RDF syntax the file has and the output format specifies the format to convert to. The converted output shall be written to standard output. The common file extensions for RDF/XML, Turtle, N3 and N-TRIPLE are `.rdf`, `.ttl`, `.n3` and `.nt`, respectively.

Running

```
java your_java_program test.n3 RDF/XML > test.rdf
```

should convert the file `test.n3`, which is assumed written in N3, to RDF/XML format and write to the file `test.rdf`.

2.1.1 Solution

A similar program for this is given in the book on page 82. See also `http://jena.sourceforge.net/IO/iohowto.html`.

My program is called `RDFConverter` and has three methods. `readModel` creates a model from a file. `writeModel` writes a model to file. The last method is `main`.

We start by importing the necessary libraries. `jena.rdf.model` is needed to make and write models, `jena.util` is used to create a model from file.

```
import com.hp.hpl.jena.rdf.model.*;
import com.hp.hpl.jena.util.*;
public class RDFConverter{
```

`readModel`. Creates a model from `file`.

```
public Model readModel(String file){
  Model model = FileManager.get().loadModel(file);
  return model;
}
```

`writeModel`. Writes `model` to `System.out` in the RDF syntax prescribed by `format`. If the RDF serialisation is specified as RDF/XML we use an RDFWriter to output the XML file using XML entities, thus increasing its readability.

```
public void writeModel(Model model, String format){
  if(format.equals("RDF/XML")){
    RDFWriter writer = model.getWriter(format);
    writer.setProperty("showDoctypeDeclaration","true");
    writer.write(model, System.out, null);
  } else{
    model.write(System.out, format);
  }
}
```

`main`. Gets arguments from `args` and sends to other methods.

```
  public static void main(String[] args){
    String infile = args[0];
    String outformat = args[1];

    RDFConverter dave = new RDFConverter();
    Model model = dave.readModel(infile);
    dave.writeModel(model, outformat.toUpperCase());
  }
} // end RDFConverter
```

2.2 Exercise

Using your RDF syntax converter program, convert the Simpsons RDF graph you wrote in last week's exercise to **RDF/XML**.

2.2.1 Results

```
<!DOCTYPE rdf:RDF [
  <!ENTITY foaf 'http://xmlns.com/foaf/0.1/'>
  <!ENTITY xsd 'http://www.w3.org/2001/XMLSchema#'>
  <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
  <!ENTITY sim 'http://www.ifi.uio.no/INF3580/simpsons#'>
  <!ENTITY fam 'http://www.ifi.uio.no/INF3580/family#'>]>
<rdf:RDF
    xmlns:rdf="&rdf;"
    xmlns:foaf="&foaf;"
    xmlns:fam="&fam;"
    xmlns:xsd="&xsd;"
    xmlns:sim="&sim;" >
  <rdf:Description rdf:about="&sim;Herb">
    <rdf:type rdf:resource="&foaf;Person"/>
  </rdf:Description>
  <rdf:Description rdf:about="&sim;Marge">
    <fam:hasSpouse rdf:nodeID="A0"/>
    <fam:hasSpouse rdf:resource="&sim;Homer"/>
    <foaf:name>Marge Simpson</foaf:name>
    <foaf:age rdf:datatype="&xsd;int">34</foaf:age>
    <rdf:type rdf:resource="&foaf;Person"/>
  </rdf:Description>
  <rdf:Description rdf:about="&sim;Lisa">
    <fam:hasParent rdf:nodeID="A1"/>
    <fam:hasParent rdf:nodeID="A2"/>
    <fam:hasMother rdf:resource="&sim;Marge"/>
    <fam:hasFather rdf:resource="&sim;Homer"/>
    <foaf:name>Lisa Simpson</foaf:name>
    <foaf:age rdf:datatype="&xsd;int">8</foaf:age>
    <rdf:type rdf:resource="&foaf;Person"/>
  </rdf:Description>
  <rdf:Description rdf:about="&sim;Patty">
    <fam:hasSister rdf:resource="&sim;Selma"/>
    <rdf:type rdf:resource="&foaf;Person"/>
  </rdf:Description>
  <rdf:Description rdf:about="&sim;Mona">
    <rdf:type rdf:resource="&foaf;Person"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A2">
    <fam:hasSister rdf:resource="&sim;Selma"/>
    <fam:hasSister rdf:resource="&sim;Patty"/>
    <rdf:type rdf:resource="&foaf;Person"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A3">
    <fam:hasMother rdf:resource="&sim;Mona"/>
    <rdf:type rdf:resource="&foaf;Person"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A4">
    <fam:hasFather rdf:resource="&sim;Abraham"/>
    <rdf:type rdf:resource="&foaf;Person"/>
  </rdf:Description>
  <rdf:Description rdf:about="&sim;Simpsons">
    <fam:hasFamilyMember rdf:resource="&sim;Marge"/>
    <fam:hasFamilyMember rdf:resource="&sim;Bart"/>
    <fam:hasFamilyMember rdf:resource="&sim;Lisa"/>
    <fam:hasFamilyMember rdf:resource="&sim;Homer"/>
    <fam:hasFamilyMember rdf:resource="&sim;Maggie"/>
    <rdf:type rdf:resource="&fam;Family"/>
  </rdf:Description>
  <rdf:Description rdf:about="&sim;Maggie">
    <fam:hasParent rdf:nodeID="A3"/>
    <fam:hasParent rdf:nodeID="A4"/>
    <fam:hasMother rdf:resource="&sim;Marge"/>
    <fam:hasFather rdf:resource="&sim;Homer"/>
    <foaf:name>Maggie Simpson</foaf:name>
    <foaf:age rdf:datatype="&xsd;int">1</foaf:age>
    <rdf:type rdf:resource="&foaf;Person"/>
  </rdf:Description>
  <rdf:Description rdf:about="&sim;Selma">
    <fam:hasSister rdf:resource="&sim;Patty"/>
    <rdf:type rdf:resource="&foaf;Person"/>
  </rdf:Description>
  <rdf:Description rdf:about="&sim;Bart">
    <fam:hasMother rdf:resource="&sim;Marge"/>
    <fam:hasFather rdf:resource="&sim;Homer"/>
    <foaf:name>Bart Simpson</foaf:name>
    <foaf:age rdf:datatype="&xsd;int">10</foaf:age>
    <rdf:type rdf:resource="&foaf;Person"/>
  </rdf:Description>
  <rdf:Description rdf:about="&sim;Homer">
    <fam:hasSpouse rdf:resource="&sim;Marge"/>
    <foaf:name>Homer Simpson</foaf:name>
    <foaf:age rdf:datatype="&xsd;int">36</foaf:age>
    <rdf:type rdf:resource="&foaf;Person"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A1">
    <fam:hasBrother rdf:resource="&sim;Herb"/>
    <rdf:type rdf:resource="&foaf;Person"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A0">
    <fam:hasFather rdf:resource="&sim;Abraham"/>
    <rdf:type rdf:resource="&foaf;Person"/>
  </rdf:Description>
  <rdf:Description rdf:about="&sim;Abraham">
    <rdf:type rdf:resource="&foaf;Person"/>
  </rdf:Description>
</rdf:RDF>
```

2.3 Exercise

Using your RDF syntax converter program, convert the Simpsons RDF graph you wrote in last week's exercise to **Turtle**.

2.3.1 Results

```
@prefix foaf:    <http://xmlns.com/foaf/0.1/> .
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix sim:     <http://www.ifi.uio.no/INF3580/simpsons#> .
@prefix fam:     <http://www.ifi.uio.no/INF3580/family#> .
```

```
sim:Herb
      a       foaf:Person .

sim:Marge
      a       foaf:Person ;
      fam:hasSpouse sim:Homer ;
      fam:hasSpouse
              [ a       foaf:Person ;
                fam:hasFather sim:Abraham
              ] ;
      foaf:age "34"^^xsd:int ;
      foaf:name "Marge Simpson" .

sim:Lisa
      a       foaf:Person ;
      fam:hasFather sim:Homer ;
      fam:hasMother sim:Marge ;
      fam:hasParent
              [ a       foaf:Person ;
                fam:hasSister sim:Selma , sim:Patty
              ] ;
      fam:hasParent
              [ a       foaf:Person ;
                fam:hasBrother sim:Herb
              ] ;
      foaf:age "8"^^xsd:int ;
      foaf:name "Lisa Simpson" .

sim:Patty
      a       foaf:Person ;
      fam:hasSister sim:Selma .

sim:Mona
      a       foaf:Person .

sim:Simpsons
      a       fam:Family ;
      fam:hasFamilyMember sim:Marge , sim:Maggie , sim:Lisa , sim:Bart , sim:Homer .

sim:Maggie
      a       foaf:Person ;
      fam:hasFather sim:Homer ;
      fam:hasMother sim:Marge ;
      fam:hasParent
              [ a       foaf:Person ;
                fam:hasMother sim:Mona
              ] ;
      fam:hasParent
              [ a       foaf:Person ;
                fam:hasFather sim:Abraham
              ] ;
      foaf:age "1"^^xsd:int ;
      foaf:name "Maggie Simpson" .

sim:Selma
      a       foaf:Person ;
      fam:hasSister sim:Patty .

sim:Bart
      a       foaf:Person ;
      fam:hasFather sim:Homer ;
      fam:hasMother sim:Marge ;
      foaf:age "10"^^xsd:int ;
      foaf:name "Bart Simpson" .

sim:Homer
      a       foaf:Person ;
      fam:hasSpouse sim:Marge ;
      foaf:age "36"^^xsd:int ;
      foaf:name "Homer Simpson" .

sim:Abraham
      a       foaf:Person .
```

2.4 Exercise

Using your RDF syntax converter program, convert the Simpsons RDF graph you wrote in last week's exercise to **N3**.

2.4.1 Results

```
@prefix foaf:    <http://xmlns.com/foaf/0.1/> .
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix sim:     <http://www.ifi.uio.no/INF3580/simpsons#> .
@prefix fam:     <http://www.ifi.uio.no/INF3580/family#> .

sim:Herb
      a       foaf:Person .

sim:Marge
      a       foaf:Person ;
      fam:hasSpouse sim:Homer ;
      fam:hasSpouse
              [ a       foaf:Person ;
                fam:hasFather sim:Abraham
              ] ;
      foaf:age "34"^^xsd:int ;
      foaf:name "Marge Simpson" .

sim:Lisa
      a       foaf:Person ;
      fam:hasFather sim:Homer ;
      fam:hasMother sim:Marge ;
      fam:hasParent
              [ a       foaf:Person ;
                fam:hasSister sim:Selma , sim:Patty
              ] ;
      fam:hasParent
              [ a       foaf:Person ;
```

```
                fam:hasBrother sim:Herb
              ] ;
      foaf:age "8"^^xsd:int ;
      foaf:name "Lisa Simpson" .

sim:Patty
      a       foaf:Person ;
      fam:hasSister sim:Selma .

sim:Mona
      a       foaf:Person .

sim:Simpsons
      a       fam:Family ;
      fam:hasFamilyMember sim:Marge , sim:Maggie , sim:Lisa , sim:Bart , sim:Homer .

sim:Maggie
      a       foaf:Person ;
      fam:hasFather sim:Homer ;
      fam:hasMother sim:Marge ;
      fam:hasParent
              [ a       foaf:Person ;
                fam:hasMother sim:Mona
              ] ;
      fam:hasParent
              [ a       foaf:Person ;
                fam:hasFather sim:Abraham
              ] ;
      foaf:age "1"^^xsd:int ;
      foaf:name "Maggie Simpson" .

sim:Selma
      a       foaf:Person ;
      fam:hasSister sim:Patty .

sim:Bart
      a       foaf:Person ;
      fam:hasFather sim:Homer ;
      fam:hasMother sim:Marge ;
      foaf:age "10"^^xsd:int ;
      foaf:name "Bart Simpson" .

sim:Homer
      a       foaf:Person ;
      fam:hasSpouse sim:Marge ;
      foaf:age "36"^^xsd:int ;
      foaf:name "Homer Simpson" .

sim:Abraham
      a       foaf:Person .
```

2.5 Exercise

Using your RDF syntax converter program, convert the Simpsons RDF graph you wrote in last week's exercise to **N-Triples**.

2.5.1 Results

These results are very verbose, but we include them for completeness.

```
_:AX2dX64cc37b2X3aX14b1609940cX3aXX2dX7ffd <http://www.ifi.uio.no/INF3580/family#hasSister> <http://www.ifi.uio.no/INF3580/simpsons#Selma> .
_:AX2dX64cc37b2X3aX14b1609940cX3aXX2dX7ffd <http://www.ifi.uio.no/INF3580/family#hasSister> <http://www.ifi.uio.no/INF3580/simpsons#Patty> .
_:AX2dX64cc37b2X3aX14b1609940cX3aXX2dX7ffd <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> .
<http://www.ifi.uio.no/INF3580/simpsons#Patty> <http://www.ifi.uio.no/INF3580/family#hasSister> <http://www.ifi.uio.no/INF3580/simpsons#Selma> .
<http://www.ifi.uio.no/INF3580/simpsons#Patty> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> .
<http://www.ifi.uio.no/INF3580/simpsons#Homer> <http://www.ifi.uio.no/INF3580/family#hasSpouse> <http://www.ifi.uio.no/INF3580/simpsons#Marge> .
<http://www.ifi.uio.no/INF3580/simpsons#Homer> <http://xmlns.com/foaf/0.1/name> "Homer Simpson" .
<http://www.ifi.uio.no/INF3580/simpsons#Homer> <http://xmlns.com/foaf/0.1/age> "36"^^<http://www.w3.org/2001/XMLSchema#int> .
<http://www.ifi.uio.no/INF3580/simpsons#Homer> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> .
_:AX2dX64cc37b2X3aX14b1609940cX3aXX2dX7ffb <http://www.ifi.uio.no/INF3580/family#hasFather> <http://www.ifi.uio.no/INF3580/simpsons#Abraham> .
_:AX2dX64cc37b2X3aX14b1609940cX3aXX2dX7ffb <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> .
<http://www.ifi.uio.no/INF3580/simpsons#Abraham> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> .
<http://www.ifi.uio.no/INF3580/simpsons#Herb> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> .
_:AX2dX64cc37b2X3aX14b1609940cX3aXX2dX7ffe <http://www.ifi.uio.no/INF3580/family#hasMother> <http://www.ifi.uio.no/INF3580/simpsons#Mona> .
_:AX2dX64cc37b2X3aX14b1609940cX3aXX2dX7ffe <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> .
<http://www.ifi.uio.no/INF3580/simpsons#Selma> <http://www.ifi.uio.no/INF3580/family#hasSister> <http://www.ifi.uio.no/INF3580/simpsons#Patty> .
<http://www.ifi.uio.no/INF3580/simpsons#Selma> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> .
_:AX2dX64cc37b2X3aX14b1609940cX3aXX2dX7ffc <http://www.ifi.uio.no/INF3580/family#hasBrother> <http://www.ifi.uio.no/INF3580/simpsons#Herb> .
_:AX2dX64cc37b2X3aX14b1609940cX3aXX2dX7ffc <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> .
<http://www.ifi.uio.no/INF3580/simpsons#Lisa> <http://www.ifi.uio.no/INF3580/family#hasParent> _:AX2dX64cc37b2X3aX14b1609940cX3aXX2dX7ffc .
<http://www.ifi.uio.no/INF3580/simpsons#Lisa> <http://www.ifi.uio.no/INF3580/family#hasParent> _:AX2dX64cc37b2X3aX14b1609940cX3aXX2dX7ffd .
<http://www.ifi.uio.no/INF3580/simpsons#Lisa> <http://www.ifi.uio.no/INF3580/family#hasMother> <http://www.ifi.uio.no/INF3580/simpsons#Marge> .
<http://www.ifi.uio.no/INF3580/simpsons#Lisa> <http://www.ifi.uio.no/INF3580/family#hasFather> <http://www.ifi.uio.no/INF3580/simpsons#Homer> .
<http://www.ifi.uio.no/INF3580/simpsons#Lisa> <http://xmlns.com/foaf/0.1/name> "Lisa Simpson" .
<http://www.ifi.uio.no/INF3580/simpsons#Lisa> <http://xmlns.com/foaf/0.1/age> "8"^^<http://www.w3.org/2001/XMLSchema#int> .
<http://www.ifi.uio.no/INF3580/simpsons#Lisa> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> .
<http://www.ifi.uio.no/INF3580/simpsons#Bart> <http://www.ifi.uio.no/INF3580/family#hasMother> <http://www.ifi.uio.no/INF3580/simpsons#Marge> .
<http://www.ifi.uio.no/INF3580/simpsons#Bart> <http://www.ifi.uio.no/INF3580/family#hasFather> <http://www.ifi.uio.no/INF3580/simpsons#Homer> .
<http://www.ifi.uio.no/INF3580/simpsons#Bart> <http://xmlns.com/foaf/0.1/name> "Bart Simpson" .
<http://www.ifi.uio.no/INF3580/simpsons#Bart> <http://xmlns.com/foaf/0.1/age> "10"^^<http://www.w3.org/2001/XMLSchema#int> .
<http://www.ifi.uio.no/INF3580/simpsons#Bart> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> .
<http://www.ifi.uio.no/INF3580/simpsons#Maggie> <http://www.ifi.uio.no/INF3580/family#hasParent> _:AX2dX64cc37b2X3aX14b1609940cX3aXX2dX7ffe .
<http://www.ifi.uio.no/INF3580/simpsons#Maggie> <http://www.ifi.uio.no/INF3580/family#hasParent> _:AX2dX64cc37b2X3aX14b1609940cX3aXX2dX7fff .
<http://www.ifi.uio.no/INF3580/simpsons#Maggie> <http://www.ifi.uio.no/INF3580/family#hasMother> <http://www.ifi.uio.no/INF3580/simpsons#Marge> .
<http://www.ifi.uio.no/INF3580/simpsons#Maggie> <http://www.ifi.uio.no/INF3580/family#hasFather> <http://www.ifi.uio.no/INF3580/simpsons#Homer> .
<http://www.ifi.uio.no/INF3580/simpsons#Maggie> <http://xmlns.com/foaf/0.1/name> "Maggie Simpson" .
<http://www.ifi.uio.no/INF3580/simpsons#Maggie> <http://xmlns.com/foaf/0.1/age> "1"^^<http://www.w3.org/2001/XMLSchema#int> .
<http://www.ifi.uio.no/INF3580/simpsons#Maggie> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> .
<http://www.ifi.uio.no/INF3580/simpsons#Simpsons> <http://www.ifi.uio.no/INF3580/family#hasFamilyMember> <http://www.ifi.uio.no/INF3580/simpsons#Marge> .
<http://www.ifi.uio.no/INF3580/simpsons#Simpsons> <http://www.ifi.uio.no/INF3580/family#hasFamilyMember> <http://www.ifi.uio.no/INF3580/simpsons#Bart> .
<http://www.ifi.uio.no/INF3580/simpsons#Simpsons> <http://www.ifi.uio.no/INF3580/family#hasFamilyMember> <http://www.ifi.uio.no/INF3580/simpsons#Lisa> .
<http://www.ifi.uio.no/INF3580/simpsons#Simpsons> <http://www.ifi.uio.no/INF3580/family#hasFamilyMember> <http://www.ifi.uio.no/INF3580/simpsons#Homer> .
<http://www.ifi.uio.no/INF3580/simpsons#Simpsons> <http://www.ifi.uio.no/INF3580/family#hasFamilyMember> <http://www.ifi.uio.no/INF3580/simpsons#Maggie> .
<http://www.ifi.uio.no/INF3580/simpsons#Simpsons> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.ifi.uio.no/INF3580/family#Family> .
<http://www.ifi.uio.no/INF3580/simpsons#Marge> <http://www.ifi.uio.no/INF3580/family#hasSpouse> _:AX2dX64cc37b2X3aX14b1609940cX3aXX2dX7ffb .
<http://www.ifi.uio.no/INF3580/simpsons#Marge> <http://www.ifi.uio.no/INF3580/family#hasSpouse> <http://www.ifi.uio.no/INF3580/simpsons#Homer> .
<http://www.ifi.uio.no/INF3580/simpsons#Marge> <http://xmlns.com/foaf/0.1/name> "Marge Simpson" .
<http://www.ifi.uio.no/INF3580/simpsons#Marge> <http://xmlns.com/foaf/0.1/age> "34"^^<http://www.w3.org/2001/XMLSchema#int> .
<http://www.ifi.uio.no/INF3580/simpsons#Marge> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> .
<http://www.ifi.uio.no/INF3580/simpsons#Mona> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> .
_:AX2dX64cc37b2X3aX14b1609940cX3aXX2dX7fff <http://www.ifi.uio.no/INF3580/family#hasFather> <http://www.ifi.uio.no/INF3580/simpsons#Abraham> .
_:AX2dX64cc37b2X3aX14b1609940cX3aXX2dX7fff <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> .
```

2.6 Exercise

Explain the differences between the four RDF serialisations. Illustrate your points with examples from the RDF conversions done in these exercises.

See also `http://www.w3.org/TR/turtle/` and `http://www.w3.org/DesignIssues/Notation3.html`.

2.6.1 Solution

All languages are able to express RDF 1.0 according to the standard, so in that sense it does not matter which one we choose.

The smallest language of the four is N-Triples. It has no "syntactic sugar" to express similar triples, only "raw" triples. It is therefore very verbose, thus time consuming for humans to read or write, but very easy for computers to parse.

The full N3 language is more expressive than what is necessary for representing RDF. It has constructs for rules and for quantifiers, i.e., `@forAll` and `@forSome`. There is also a N3-rdf sublanguage of N3, which is meant to have the expressiveness of RDF, but the status of this language is not clear to me.

Turtle lies in the sweet spot between N-Triples and N3, at least when for the purpose of expressing RDF. See http://www.w3.org/TeamSubmission/turtle/ chapters 8 and 9 for a comparison of the three. Comparing the output of the N3 simpsons file against the Turtle version the only relevant difference I could find between the two was that in N3 `rdf:type` is abbreviated to `a`, while Turtle does not use this abbreviation. This is probably due to the implementation in Jena, since the documentation of Turtle I have found allows this abbreviation.

RDF/XML is currently the only W3C recommended RDF language, which means that software claiming to support RDF must support RDF expressed in RDF/XML. An important feature of RDF/XML is that it is XML. This makes it quite verbose and forces a hierarchical structure on the graph nature of RDF, but allows for namespace declarations, XML entities and other space saving features.

3 Browsing RDF using Jena

In these exercises we will learn more about Jena by making a program that browses FOAF RDF graphs for friends.

3.1 Exercise

Make a java program which reads a FOAF file and lists the names of all the persons the FOAF person knows and the names of who they again know, i.e., all friends and the friends of these friends. The program should not use queries. You can assume that the name of a person is located in a `foaf:name` element and the foaf file of a person is located in a `rdfs:seeAlso` element—just like how your own FOAF file should look like if generated by the FOAF-a-Matic.

3.1.1 Tip

Running your program on Martin G. Skjæveland's FOAF file should give you something like this

```
Martin G. Skjæveland’s friends:
   Tim Berners-Lee
   Martin Giese
      Kjetil Kjernsmo
      Arild Waaler
      Bruno Buchberger
      Reiner Hähnle
      David Norheim
      Renate Schmidt
      Roar Fjellheim
   Robert Engels
```

3.1.2 Solution

This solution uses recursion to traverse FOAF files looking for friends, their names and their FOAF files.

We start by importing the necessary libraries. `jena.vocabulary` is imported as it has the standard RDFS vocabulary represented. It does not have the FOAF vocabulary represented, so we create the properties we need as `public static final`.

```
 1  import com.hp.hpl.jena.rdf.model.*;
 2  import com.hp.hpl.jena.util.*;
 3  import com.hp.hpl.jena.vocabulary.*;
 4
 5  public class FOAFFriendFinder {
 6    public static final String NS_FOAF = "http://xmlns.com/foaf/0.1/";
 7    public static final Property FOAF_knows =
 8      ResourceFactory.createProperty(NS_FOAF, "knows");
 9    public static final Property FOAF_name =
10      ResourceFactory.createProperty(NS_FOAF, "name");
11    public static final String INDENT = "   ";
```

The "read model" method is the same as in the converter program.

```
12  public Model readModel(String file) {
13    Model model = FileManager.get().loadModel(file);
14    return model;
15  }
```

The following method writes the name of `# me` in the FOAF file given in the parameter `FOAFURL` and prints all the "transitive" FOAF friends in the FOAF file of depth `depth` by using the method `wFOAFFriends`.

```
16  public void wFOAFFriends(String FOAFURL, int depth){
17    Model model = readModel(FOAFURL);
18    Resource me = model.createResource(FOAFURL + "#me");
19    NodeIterator mynames = model.listObjectsOfProperty(me, FOAF_name);
20    if(mynames.hasNext()){
21      System.out.println(mynames.next() + "'s friends:");
22    } else{
23      System.out.println(FOAFURL + "#me's friends:");
24    }
25    wFOAFFriends(FOAFURL, depth, 0);
26  }
```

The next method does most of the work. It prints the FOAF friends of the FOAF file `FOAFURL` at depth `depth` and indents the results by `indent` number of indents by `INDENT`. By indenting the printed friends we can easily see who is who's immediate friend. Now we will go through the method in some detail. The numbers in parenthesis refer to line numbers in the code.

For each call to this method the depth is reduced by one (28) and the indentation size is increased by one (29). The model at `FOAFURL` is loaded (30) and if the model is successfully loaded (31), we list the friends in the model. First we find `# me` (32) and all its related objects by the `foaf:knows` property (33)—which will be `# me`'s friends. For each friend (34), get the name(s) (36,37) of the friend and print them (41) indented (38, 39). If there are deeper depths of friends to explore (43), find the FOAF files of the friend (44), and repeat the process on him/her (46, 47).

```
27  public void wFOAFFriends(String FOAFURL, int depth, int indent){
28    depth--;
29    indent++;
30    Model model = readModel(FOAFURL);
31    if(model != null){
32      Resource me = model.createResource(FOAFURL + "#me");
```

```
    NodeIterator friends = model.listObjectsOfProperty(me, FOAF_knows);
    while(friends.hasNext()){
      Resource friend = (Resource)friends.next();
      NodeIterator friend_names = model.listObjectsOfProperty(friend, FOAF_name);
      while(friend_names.hasNext()){
 for(int i = 0; i < indent; i++){
   System.out.print(INDENT);
 }
 System.out.println(friend_names.next());
      }
      if(depth != 0){
 NodeIterator friend_seeAlso = model.listObjectsOfProperty(friend, RDFS.seeAlso);
 while(friend_seeAlso.hasNext()){
   String friend_FOAFURL = friend_seeAlso.next().toString();
   wFOAFFriends(friend_FOAFURL, depth, indent);
 }
      }
    }
  }
}
```

`main` method. Takes FOAF file and recursion depth as input and passes on to other method.

```
  public static void main(String args[]){
    FOAFFriendFinder dave = new FOAFFriendFinder();
    dave.wFOAFFriends(args[0], Integer.parseInt(args[1]));
  }
} // end class
```

3.2 Exercise

Run your program on your own FOAF file.

3.2.1 Solution

According to my FOAF file, these are my friends:

```
Martin G. Skjæveland’s friends:
   Tim Berners-Lee
   Martin Giese
      Kjetil Kjernsmo
      Arild Waaler
      Bruno Buchberger
      Reiner Hähnle
      David Norheim
      Renate Schmidt
      Roar Fjellheim
   Robert Engels
```

4 RDF summary program

When opening OWL ontologies in ontology editors it is common to be presented with some metrics about the ontology, e.g., how many classes and relationships there are and what kinds of axioms are used. In this exercise we will create a similar program, but for RDF graphs.

4.1 Exercise

Write a program which reads an RDF file and outputs

- the total number of triples in the file
- the number of distinct subjects, predicates and objects in the graph
- all *types*, e.g., all objects in a triple where `rdf:type` is the predicate, and the distinct members of these types

4.1.1 Tip

Running your program with the following graph as input

```
@prefix :    <http://example.com/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

 :m rdf:type :A .
 :n rdf:type :A .
 :m :r       :o .
 :q :r       :n .
```

should give a result equivalent of the following:

```
RDF metrics: execs/summary_test.n3
No. of triples: 4
No. of subject: 3
No. of predicates: 2
No. of objects: 3

TYPES:
http://example.com/A
   http://example.com/n
   http://example.com/m
```

4.1.2 Solution

My class is called `RDFMetrics`. I include the necessary libraries (I'll come back to the ones we have not seen before) and make a constant for indenting the output.

```
import com.hp.hpl.jena.rdf.model.*;
import com.hp.hpl.jena.util.FileManager;
import com.hp.hpl.jena.vocabulary.RDF;
import com.hp.hpl.jena.graph.Node;
import com.hp.hpl.jena.util.iterator.ExtendedIterator;
import java.util.Iterator;

public class RDFMetrics {
  protected static Model modelRDF;
  public static final String INDENT = "    ";
```

readModel is the same as we have seen before.

The next method gets the size of an iterator. All of the things that this program will count is kept in iterators.

```
public void readModel(String file) {
  modelRDF = FileManager.get().loadModel(file);
}
public int getIteratorSize(Iterator i){
  int count = 0;
  while(i.hasNext()){
    count++;
    i.next();
  }
  return count;
}
```

There is no `Model.listPredicates()` in the Jena API (while there are `listSubjects()` and `listObjects()`, so with help from the jena-dev mailinglist[2] I added one. This is an example of diving into the low-level interface of Jena, its service provider interface (SPI), see lecture 3, slide 21. The definition of this method is the reason why `jena.graph.Node` and `jena.util.iterator.ExtendedIterator` is imported.

```
private ExtendedIterator<Node> listPredicates(Model model){
  return model.getGraph().queryHandler().predicatesFor(Node.ANY, Node.ANY);
}
```

Then follows a block of counting methods. All methods retrieves an iterator containing the things we want to counts and passes it to the iterator counter method. Special cases are made for null values. Notice the funny cast of null to an RDFNode in `countSubjects` and `countPredicates`. This is to make the call unambiguous, there is also a `listStatements` method defined for superinterface of Model.

```
public int countStatements(Resource s, Property p, RDFNode o, Model m){
  if(s == null && p == null && m == null){
    return getIteratorSize(m.listStatements());
  } else{
    return getIteratorSize(m.listStatements(s, p, o));
  }
}
public int countSubjects(Resource s, Model m){
  if(s == null){
    return getIteratorSize(m.listSubjects());
  } else{
    return getIteratorSize(m.listStatements(s, null, (RDFNode)null));
  }
}
public int countPredicates(Property p, Model m){
  if(p == null){
    return getIteratorSize(listPredicates(m));
  } else{
    return getIteratorSize(m.listStatements(null, p, (RDFNode)null));
  }
}
public int countObjects(RDFNode o, Model m){
  if(o == null){
```

[2] http://tech.groups.yahoo.com/group/jena-dev/message/42236

```
    return getIteratorSize(m.listObjects());
  } else{
    return getIteratorSize(m.listStatements(null, null, o));
  }
}
```

The following method prints the types in the model and all the distinct members of the type. First, it gets all objects of the property `rdf:type`, i.e., all objects in triples where `rdf:type` is the predicate, and iterates through them. Notice that Jena has a vocabulary class for RDF. For each object, it writes the URI to `System.out`, and gets and writes all the subjects in triples where the predicate is `rdf:type` and `o` is the object.

```
public void printType(Model model){
  NodeIterator ni = model.listObjectsOfProperty(RDF.type);
  while(ni.hasNext()){
    RDFNode o = ni.next();
    System.out.println(o.toString());
    ResIterator ri = model.listResourcesWithProperty(RDF.type, o);
    while(ri.hasNext()){
 System.out.println(INDENT + ri.next());
    }
  }
}
```

`printTripleData` prints data about triples by using the counting method above. Since we want to count everything we set no restrictions, i.e., pass null to all methods.

```
public void printTripleData(Model model){
  System.out.println("No. of triples: " +
        countStatements(null, null, null, model));
  System.out.println("No. of subject: " +
        countSubjects(null, model));
  System.out.println("No. of predicates: " +
        countPredicates(null, model));
  System.out.println("No. of objects: " +
        countObjects(null, model));
}
```

`main` reads arguments and calls the two print data methods.

```
public static void main(String[] args){
  RDFMetrics dave = new RDFMetrics();
  dave.readModel(args[0]);
  System.out.println("RDF metrics: " + args[0]);
  dave.printTripleData(modelRDF);
  System.out.println("\nTYPES:");
  dave.printType(modelRDF);
  }
} // end class
```

4.2 Exercise

Use your program to analyse your FOAF file.

4.2.1 Results

```
RDF metrics: execs/foaf.rdf
No. of triples: 40
No. of subject: 7
No. of predicates: 24
No. of objects: 30

TYPES:
http://xmlns.com/foaf/0.1/Person
   http://folk.uio.no/martige/foaf.rdf#me
   http://www.w3.org/People/Berners-Lee/card#i
   http://folk.uio.no/martingi/foaf.rdf#me
   http://www.esis.no/data/robert.engels/foaf.rdf#me
http://xmlns.com/foaf/0.1/Organization
   http://folk.uio.no/martige/foaf.rdf#ifi
http://xmlns.com/foaf/0.1/PersonalProfileDocument
   http://folk.uio.no/martige/foaf.rdf
http://www.w3.org/2003/01/geo/wgs84_pos#Point
   142bdc5b:14b1609a24e:-7fff
```

4.3 Exercise

Use your program to analyse Martin Giese's FOAF file.

4.3.1 Results

```
RDF metrics: execs/martingi.foaf.rdf
No. of triples: 65
No. of subject: 11
No. of predicates: 26
No. of objects: 53

TYPES:
http://xmlns.com/foaf/0.1/Person
   file:execs/martingi.foaf.rdf#arild
   406f41ec:14b1609a690:-7ffe
   http://www.kjetil.kjernsmo.net/foaf#me
   406f41ec:14b1609a690:-7ffd
   http://dbpedia.org/resource/Bruno_Buchberger
   406f41ec:14b1609a690:-7ffc
   file:execs/martingi.foaf.rdf#me
   http://data.semanticweb.org/person/renate-schmidt
http://xmlns.com/foaf/0.1/Organization
   file:execs/martingi.foaf.rdf#uio
http://xmlns.com/foaf/0.1/PersonalProfileDocument
   file:execs/martingi.foaf.rdf
http://www.w3.org/2003/01/geo/wgs84_pos#Point
   406f41ec:14b1609a690:-7fff
```

4.4 Exercise

Use your program to analyse your Simpsons RDF file.

4.4.1 Results

```
RDF metrics: execs/simpsons.ttl
No. of triples: 52
No. of subject: 16
No. of predicates: 10
No. of objects: 27

TYPES:
http://www.ifi.uio.no/INF3580/family#Family
   http://www.ifi.uio.no/INF3580/simpsons#Simpsons
http://xmlns.com/foaf/0.1/Person
   http://www.ifi.uio.no/INF3580/simpsons#Herb
   http://www.ifi.uio.no/INF3580/simpsons#Marge
   http://www.ifi.uio.no/INF3580/simpsons#Lisa
   http://www.ifi.uio.no/INF3580/simpsons#Patty
   http://www.ifi.uio.no/INF3580/simpsons#Mona
   5060bfbb:14b1609aa67:-7ffb
   5060bfbb:14b1609aa67:-7ffc
   http://www.ifi.uio.no/INF3580/simpsons#Maggie
   5060bfbb:14b1609aa67:-7ffe
   http://www.ifi.uio.no/INF3580/simpsons#Bart
   5060bfbb:14b1609aa67:-7ffd
   http://www.ifi.uio.no/INF3580/simpsons#Selma
   http://www.ifi.uio.no/INF3580/simpsons#Homer
   5060bfbb:14b1609aa67:-7fff
   http://www.ifi.uio.no/INF3580/simpsons#Abraham
```