

- JAVA program export
- OWL API (2012)
- 5 min idea about your appl.

Applying protege OWL A... cw.unik.no/images/UNIK... UNIK4710 - CWI

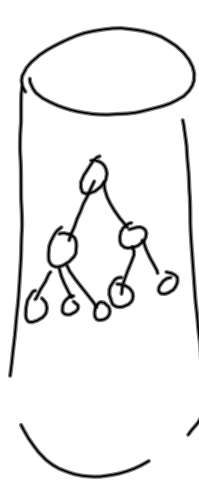
cwi.unik.no/wiki/UNIK4710

- This course is a master course (UNIK4710) - for selected topics in mobile service delivery - see UNIK9710
- List of Participants: UNIK4710V13Participants (2012: UNIK4710V12Participants), social media: [Google Plus UNIK4710 group](#)
- The course takes place on fridays, 0900-1200h at UNIK. A video communication is available to Ifi, Room Scheme@ifi.uio.no (room 1251).
- We'll have video streaming: <mms://lux.unik.no/301>
- we'll try to have a recorded session from every lecture, see description for the first two lectures
- you may also join from home through Polycom Local Presence, see [Video_conference](#)
- The course is built on your inputs. You will be asked to come up with a scenario, describe it and implement it.
- Evaluation is based on a presentation of topics and the implementation of your scenario.
- Implementations of Ontologies are available at: <https://github.com/unikdrift/UNIK4710-owl>

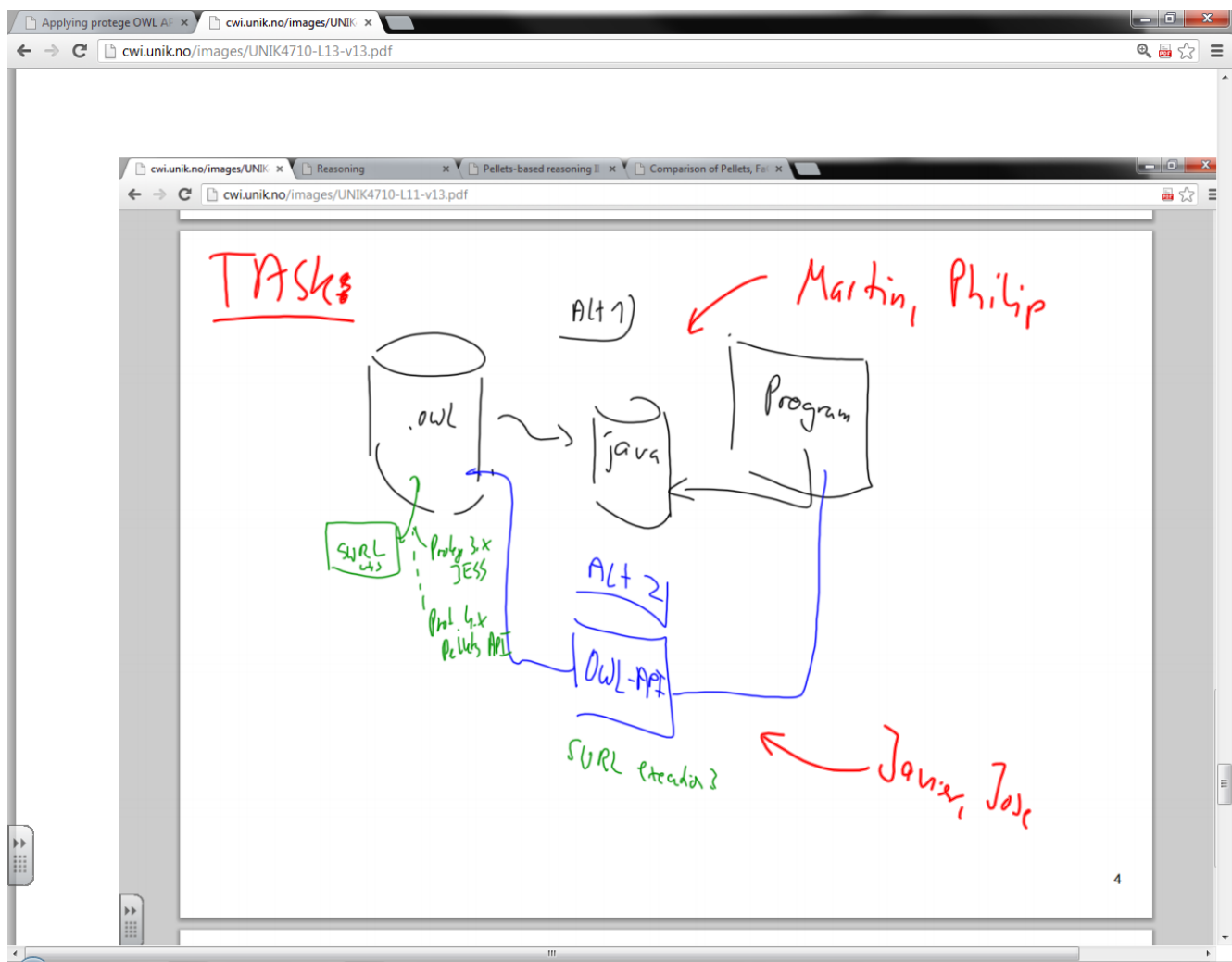
```

graph TD
    Thing --> SemanticTechnologies
    Thing --> Context
    SemanticTechnologies --> Protege
    SemanticTechnologies --> Reasoner
    SemanticTechnologies --> SemanticMediaWiki
    SemanticTechnologies --> SemanticFormats
    Protege --> Protege41[Protege4.1]
    Protege --> Protege345[Protege3.45]
    Reasoner --> ExternalRules
    Reasoner --> InternalRules
    Context --> DynamicMoving
    Context --> Indoor
    Indoor --> PublicPlace
    Indoor --> PrivatPlace
    PublicPlace --> Home
    PublicPlace --> PrivatPlace
  
```

Challenges



- SQL - mathem. operations on
- SURL - count $\rightarrow$ average Owl
- Writing back knowledge into owl
- Can do
- Query
- in Part 3.1
- Java pyth as interface



Presentation_Java_Export.pdf - Adobe Acrobat Pro Extended

File Edit View Document Comments Forms Tools Advanced Window Help

Create Combine Collaborate Secure Sign Forms Multimedia Comment

5 / 7 100% Find

Bind individual to helper class

```
//Get all male objects
Set<OWLEntity> entity = ontology.getEntitiesInSignature(
    IRI.create(URI.create("http://www.semanticweb.org/ontologies/2013/2/Travel#Male"))
));

DefaultMale individualMartin = null;
//Iterate male objects
for(OWLEntity e: entity){
    for(OWLIndividual i: e.asOWLClass().getIndividuals(ontology)){
        //System.out.println(i.toStringID());
        IRI userIRI = IRI.create(i.toStringID());
        DefaultMale anonymousMale = new DefaultMale(ontology, userIRI);

        //Locate correct user
        if(anonymousMale.getHasUserId().contains(martinClient.getUserId())){
            individualMartin = anonymousMale;
            break;
        }
    }
}
```

Travel

male

Property

userid

Martin

userid

Presentation_Java_Export.pdf - Adobe Acrobat Pro Extended

File Edit View Document Comments Forms Tools Advanced Window Help

Create Combine Collaborate Secure Sign Forms Multimedia Comment

6 / 7 100% Find

Use the helper class

```
if(individualMartin != null){
    System.out.println("Martin was found!");

    //Print individual info
    System.out.println(individualMartin.toString());
    System.out.println(individualMartin.getHasFirstName().toArray()[0] + " " +
individualMartin.getHasLastName().toArray()[0] + ", with UserID: " +
individualMartin.getHasUserId().toArray()[0]);

    //Print individual reference
    if(individualMartin.hasHasResidence()){
        Collection<Place> residence = (Collection<Place>) individualMartin.getHasResidence();
        System.out.print("Individual lives in: ");
        for(Place pl: residence){
            System.out.println(pl.getHasPlaceName().toArray()[0]);
        }
    }
}
```

Person

Resides

Martin

Oslo

home address

13,33 x 7,50 in

Presentation_Java_Export.pdf - Adobe Acrobat Pro Extended

File Edit View Document Comments Forms Tools Advanced Window Help

Create Combine Collaborate Secure Sign Forms Multimedia Comment

7 / 7 100% Find

Advantages/ Outcome:

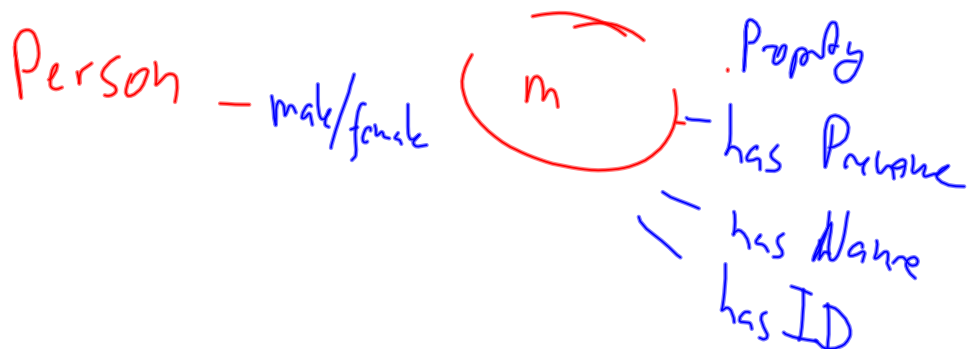
- easy class retrieval
- individual handling
Write back?
- handling classes
- hierarchy
- guaranteed code exchange

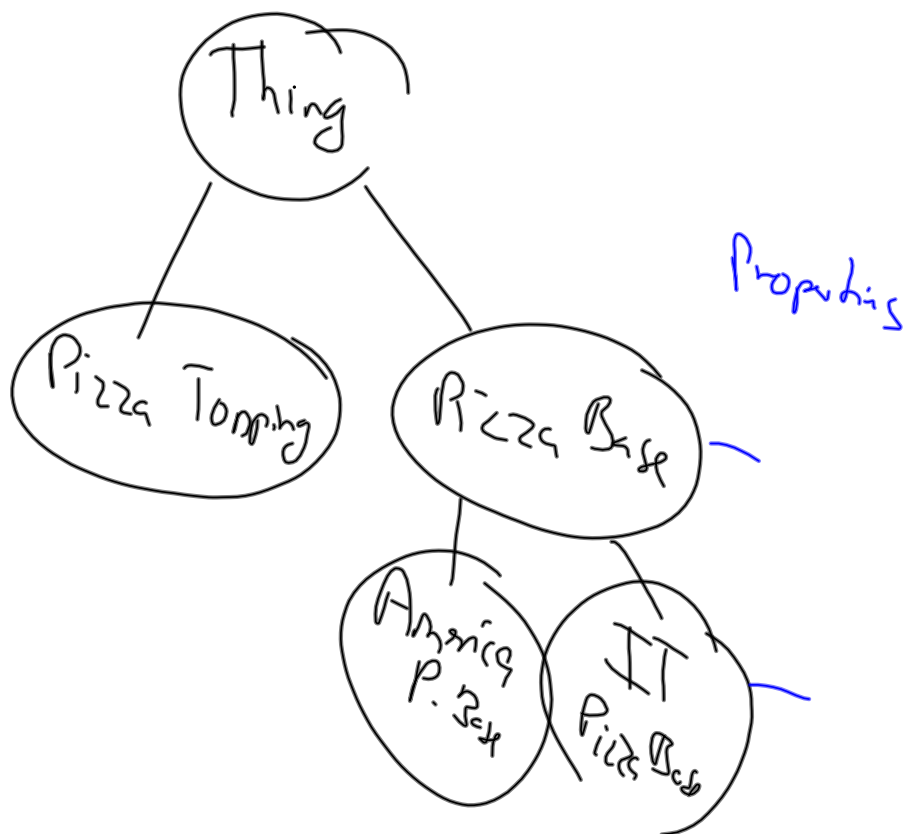
```

Male(
  hasDrinksTypePreference: Americano;
  hasFoodTypePreference: Cheeseburger, Maki, ThaiWok;
  hasResidence: Oslo;
  hasTrip: TripMartinToWashington;
  hasFirstName: Martin;
  hasGPSLocation: 14;
  hasLastName: Folkeseth;
  hasUserId: 101;
)
  
```

Martin Folkeseth, with UserID: 101

Individual lives in: Oslo





RDF - Indiv. $\leftrightarrow$ properties
 $\downarrow$
 "Classes"

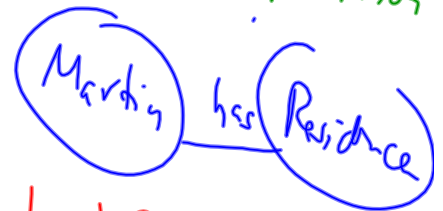
DWL

Class representation

$\Downarrow$ java
 relations & class names

SWRL, SQWRL

Individual representation



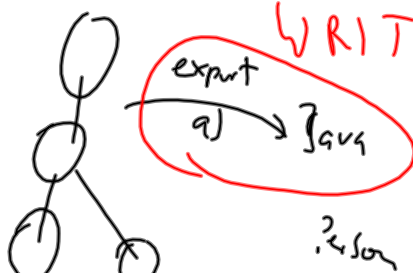
how to Reason?

not exported

WRITE back?

$\rightarrow$ OWL API

$\rightarrow$ write back file



also possible

.owl

Reason through Java

- ^{- consistency}
^{- inherit knowledge} OWL API → write back into OWL
→ execute SWRL

- "forget about SWRL"

Instead: Reason through Java Expressions

- might be inefficient

+ negotiation, "not"
count, or

+ time

OWL2-API-Arne-Song.pdf - Adobe Acrobat Pro Extended

File Edit View Document Comments Forms Tools Advanced Window Help

Create Combine Collaborate Secure Sign Forms Multimedia Comment

2 / 8 105% Find

The OWL API includes the following components:

- An API for OWL 2 and an efficient in-memory reference implementation;
- RDF/XML parser and writer.
- OWL/XML parser and writer.
- OWL Functional Syntax parser and writer.
- ? Turtle parser and writer.
- ? KRSS parser.
- OBO Flat file format parser.
- Reasoner interfaces for working with reasoners such as FaCT++, HermiT, Pellet and Racer.

Hand-drawn diagram showing a tree structure of nodes and edges, with a red circle around the root node labeled 'OWL' and a green circle around the word 'reasoners' in the list. A blue arrow points from the 'OWL' circle to the 'Reasoner' circle.

OWL2-API-Arne-Song.pdf - Adobe Acrobat Pro Extended

File Edit View Document Comments Forms Tools Advanced Window Help

Create Combine Collaborate Secure Sign Forms Multimedia Comment

5 / 8 105% Find

What to do with OWL API?

- **A Code examples**

ng,
API

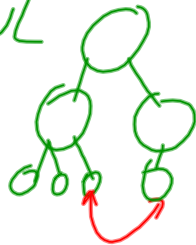
- [Loading Ontologies](#) - Shows how to **load an** an ontology
- [Saving Ontologies](#) - Shows how to **save an** an ontology
- [Entities](#) - Shows how to obtain references to **entities** (classes, properties, individuals etc.)
- [Data Ranges 1](#) - Shows how to work with **data types and other data ranges**
- [Data Ranges 2](#) - Shows how to work with **user defined data ranges** (e.g. **int > 10**) ?
- [Literals](#) - Shows how to work with **string, data values and language tags**
- [Adding Axioms](#) - Shows how to create an empty ontology, **add axioms** and save
- [Classes and Instances](#) - Shows how to specify that an **individual is an instance of a class**
- [Property Assertions 1](#) - Shows how to specify that two individuals are **related** to each other.
- [Property Assertions 2](#) - Shows how to add an **object property assertion (triple)** to an ontology
- [Deleting Entities](#) - Shows how to **delete entities** (classes, properties and individuals) from an ontology
- [Restrictions](#) - Shows how to create **restrictions** and **add them to classes** as superclasses
- [SWRL Rules](#) - Shows how to create an ontology and add some **rules**
- [Reasoning](#) - Shows how to interact with a **reasoner**
- [Visitors](#) - Shows how to collect the **properties** that are used in **restrictions** on a given class
- [Annotations](#) - Shows how to work with **annotations** such as **labels** and **comments**
- [Saving Inferred Axioms](#) - Shows how to save **inferred axioms** into a new ontology, or back into an existing ontology
- [Merging Ontologies](#) - Shows how two (or more) ontologies can be **merged** in a simple way
- [Walking Asserted Structure](#) - Shows how to **'walk'** over the **asserted structure** of an ontology.
- [Using Ontology IRI Mappers](#) - Shows how to use OWLOntologyIRIMappers to redirect loading and loading of imports.
- [Module Extraction](#) - Shows how to extract a locality based module from an ontology.

OWL API

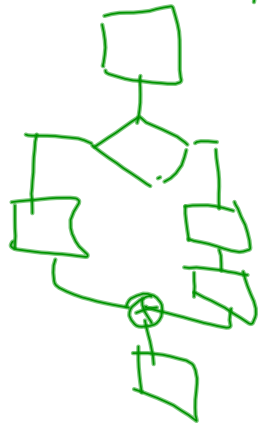
Facts

- code example
- communicate with owl
- thinks owl

Content



- more logic // business process oriented



Evaluation Recommendations

Java export

- easy export // user friendly
- creates structure for java prog
- allows to "think" java
- just classes and relations
- don't need to think owl
- implicit knowledge
- programming, object oriented
- does not need owl knowledge

Javier: → Android App → Java

→ Java API

José

—— // ——

Philip

"at home with Java" →

consider Python with own rules
 ↳ with Sem. Web

Martin

.net OWL API, ~~also~~ Java

old smviki

new quito

