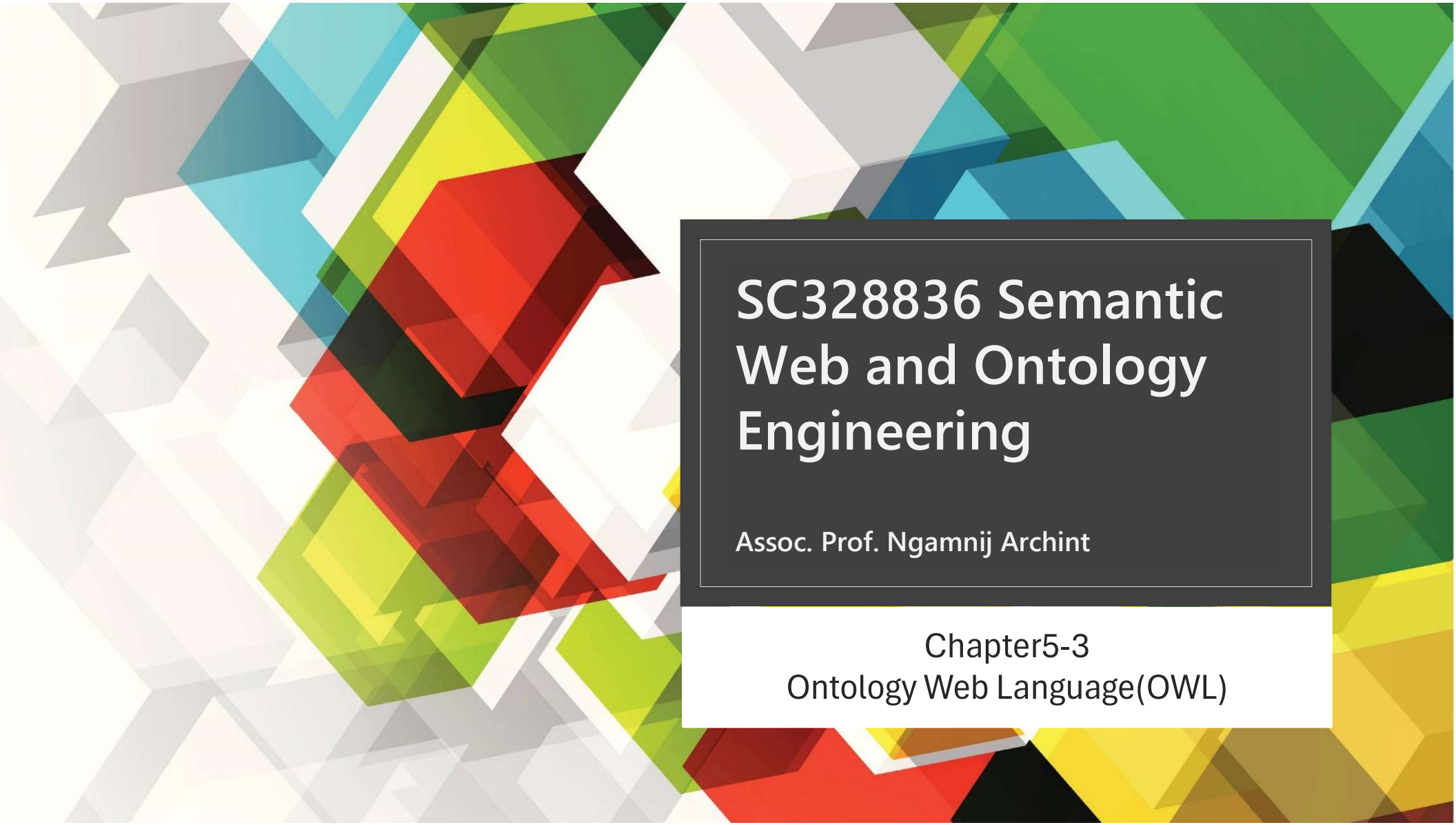


SC328836 Semantic Web and Ontology Engineering

Assoc. Prof. Ngamnij Archint

Chapter5-3
Ontology Web Language(OWL)

Ontology Web Language (OWL)

- **Topics**
  - owl:equivalentProperty
  - Using OWL to manage classes
  - Using OWL for instance document
  - Importing other OWL documents

Using OWL to set equivalent Property

owl:equivalentProperty

Equivalent Properties

- เนื้อหาต่อไป จะเป็นการกล่าวถึงคุณสมบัติความเท่าเทียมกันของ Properties

Subject name is equivalent to the Title property in Dublin Core

Subject

Properties:
name: *Literal*

ถ้าต้องการกำหนดให้ property “name” มีความหมายเหมือนกับ property “dcterms:title” เราจะกำหนดได้อย่างไร?

Defining name to be equivalent to dcterms:title

```
@base <http://www.university.edu/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dcterms: <http://purl.org/dc/terms/> .

<name> a owl:DatatypeProperty;
    owl:equivalentProperty dcterms:title;
    rdfs:domain <Subject>;
    rdfs:range rdfs:Literal.

dcterms:title  a owl:DatatypeProperty.
```

Use Protégé to create this ontology too.

```
<CS10001>
  a <Subject>;
  <name> "Semantic Web".
```

ในตย.นี้จะกำหนดให้ instance <CS10001> ซึ่งเป็น Subject มีการใช้ property <name> อย่างไรก็ตาม เรายังสามารถถาม inference engine (โดยใช้ SPARQL) ว่า <CS10001> มี dcterms:title คืออะไรได้ด้วย เนื่องจากเรามีการกำหนดให้ <name> equivalent กับ <dcterms:title> นั่นเอง

Using OWL to manage Classes

owl:intersectionOf

owl:disjointWith

owl:unionOf

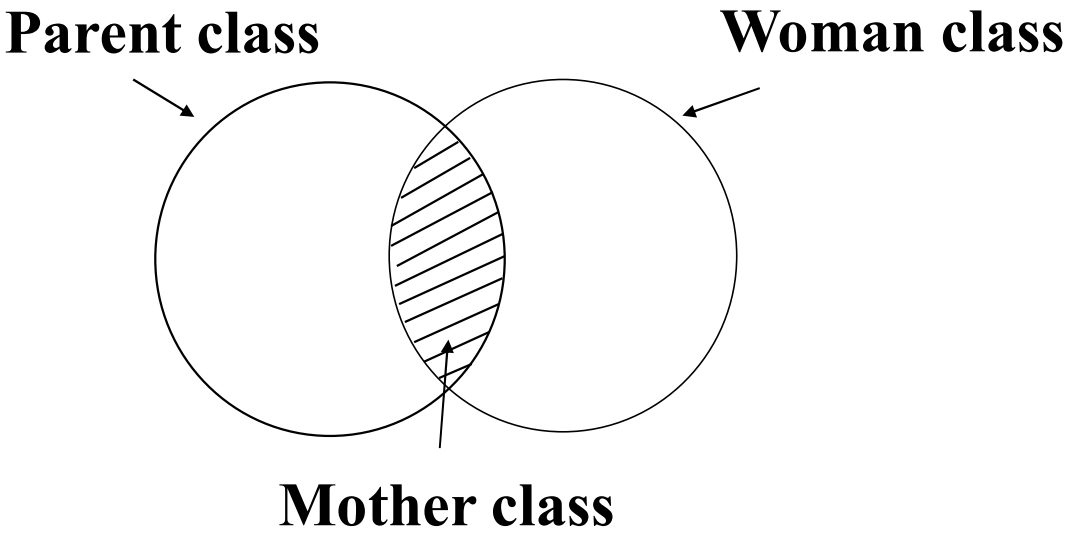
owl:oneOf

Constructing Classes using Set Operators

- OWL ยังช่วยให้เราสามารถกำหนดคุณสมบัติต่อไปนี้ให้กับ class ได้อีกด้วย
  - owl:intersectionOf
  - owl:disjointWith
  - owl:unionOf
  - owl:oneOf

Understanding owl:intersectionOf

- ถ้าเราต้องการกำหนดให้ class Mother เกิดจากการ intersection กันระหว่าง class Parent กับ class Woman เราจะสร้างด้วย OWL ได้อย่างไร



Understanding owl:intersectionOf

Use Protégé to create this ontology too.

```
@prefix : <http://example.com/owl/person/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

:Woman a owl:Class.
:Parent a owl:Class.

:Mother a owl:Class;
    owl:intersectionOf (:Woman :Parent) .
```

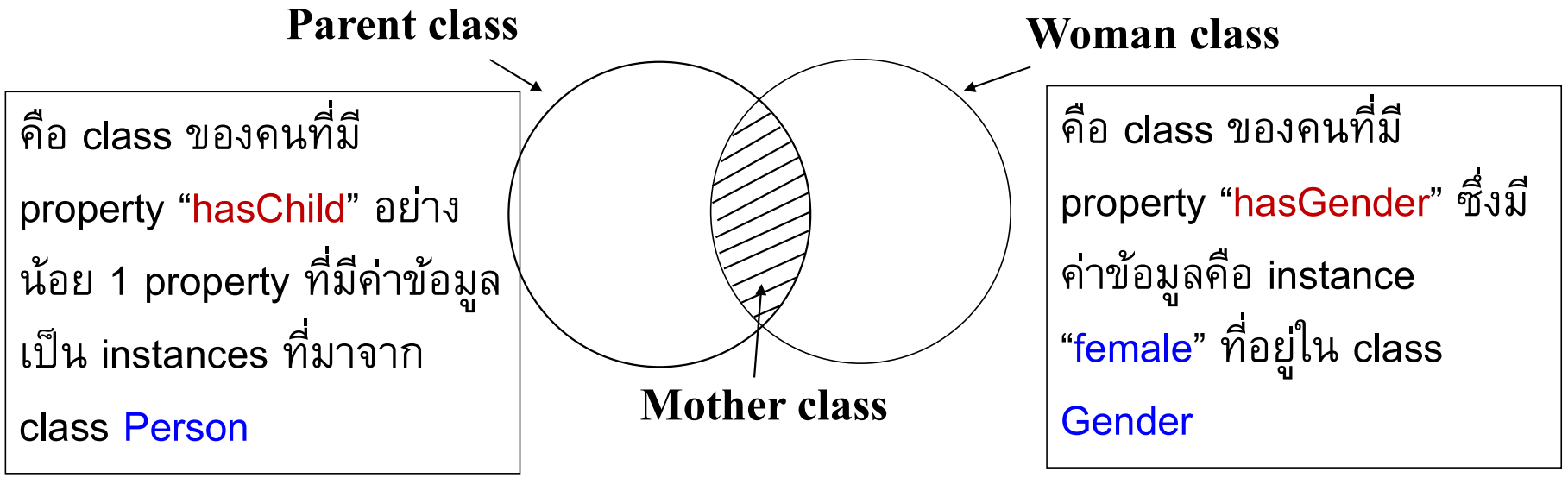
Same = mother

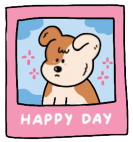
```
:3168812422
    a  :Woman.
:3132234433
    a  :Woman.
:3168812422
    a :Parent.
```

สำหรับตัวอย่างนี้ เมื่อเราสอบถาม inference engine (โดยใช้ SPARQL) เกี่ยวกับ rdf:type ของคนที่มีรหัสคือ :3168812422 ก็จะได้คำตอบคือ class “Woman”, “Parent” และ “Mother”

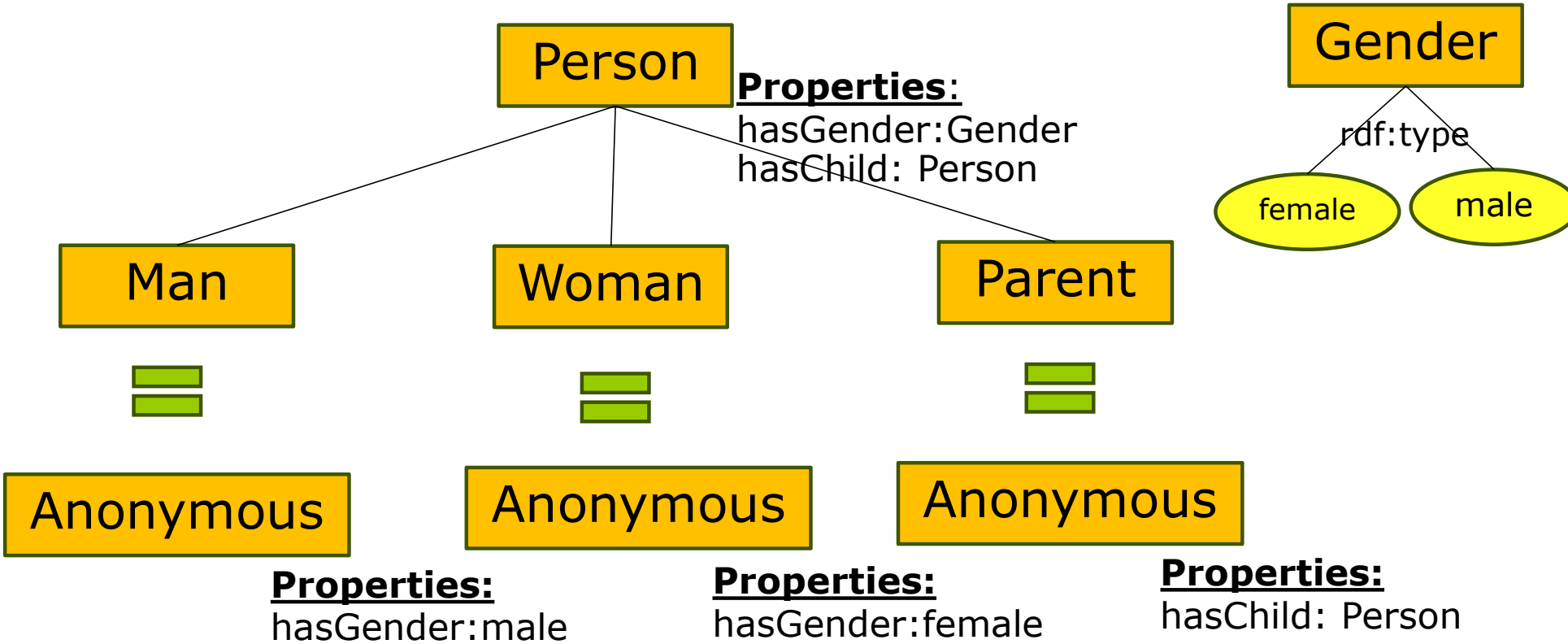
Another example of owl:intersectionOf

- ถ้าต้องการอธิบายว่า class Mother เกิดจากการ intersection ของ class Parent และ class “Woman” โดย class Parent ก็คือคนทุกคนที่มี property “hasChild” อย่างน้อย 1 property ซึ่งค่าข้อมูลของ property นี้ก็จะเป็น instance ที่เป็นคนที่อยู่ใน class Person นั่นเอง และclass Woman ก็จะได้แก่คนทุกคนที่มี property “gender” ซึ่งมีค่าข้อมูลคือ “female”





Another example of owl:intersectionOf



Understanding owl:intersectionOf

```
@prefix : <http://example.com/owl/person/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

:Person a owl:Class.
:Gender a owl:Class.
:Woman
   rdfs:subClassOf :Person;
   owl:equivalentClass _:x1.
_:x1
      a owl:Restriction;
      owl:onProperty :hasGender;
      owl:hasValue :female.
:Man
   rdfs:subClassOf :Person;
   owl:equivalentClass _:x2.
_:x2
      a owl:Restriction;
      owl:onProperty :hasGender;
      owl:hasValue :male.
```

Use Protégé to create this ontology too.

```
:Parent
    rdfs:subClassOf :Person;
    owl:equivalentClass _:x3.
_:x3 a owl:Restriction;
     owl:onProperty :hasChild;
     owl:someValuesFrom :Person.
:Mother a owl:Class;
   owl:intersectionOf (:Parent :Woman) .

:hasGender a owl:ObjectProperty;
    rdfs:domain :Person;
    rdfs:range :Gender.
:hasChild a owl:ObjectProperty;
    rdfs:domain :Person;
    rdfs:range :Person.
```

Understanding owl:intersectionOf

Use Protégé to create this ontology too.

 # Define Instances

```
:female a :Gender.
:male a :Gender.

:David a :Person.
:Willy a :Person.
:Mary a :Person.
:Jane a :Person.
:Katty  a :Person.
:Rose a :Person.

:Mary :hasChild :David.
:Jane :hasGender :female.
:Mary :hasGender :female.
:Katty :hasGender :female.
:Rose :hasChild :Katty.
:David :hasChild :Jane.
:Rose :hasGender :female.
```

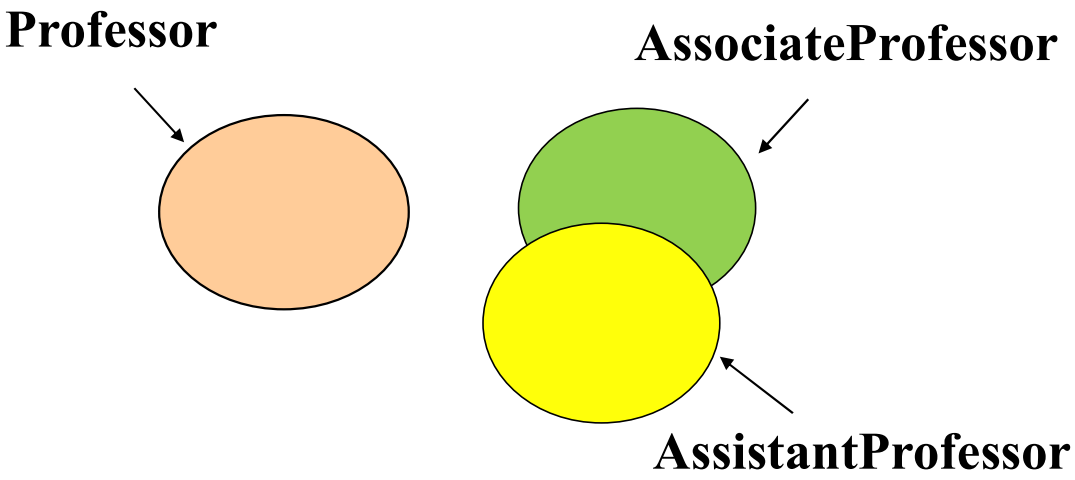
ให้ตอบว่ามีใครบ้างที่มี

rdf:type เป็น Woman,

Man, Parent, หรือ Mother

Understanding owl:disjointWith

- ถ้าต้องการอธิบายว่า class Professor จะ disjoints กับ class AssociateProfessor และ AssistantProfessor เราจะเขียน owl ได้อย่างไร

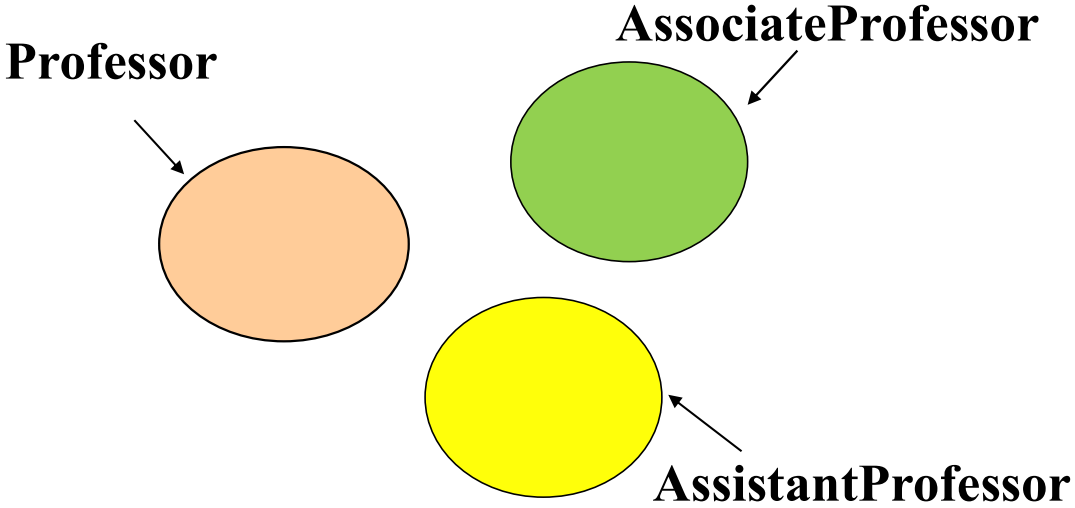


จากรูปจะสามารถกล่าวอีกนัยหนึ่งก็คือ จะต้องไม่มี instances ใดๆของ class Professor ไปเป็นสมาชิกอยู่ใน class AssociateProfessor หรือ AssistantProfessor อย่างไรก็ตาม ก็ไม่ได้หมายความว่าทั้งสาม classes นี้จะ disjoint กันทั้งหมด เพราะ class AssociateProfessor ก็อาจมีสมาชิกที่เป็น instances ที่อยู่ทั้งใน AssociateProfessor และ AssistantProfessor ด้วยกันทั้งคู่ก็ได้

```
:Professor owl:disjointWith :AssociateProfessor,
                            :AssistantProfessor.
```

Understanding owl:disjointWith

- ดังนั้น ถ้าต้องการบังคับให้ทั้งสาม classes นี้มีการ disjoint กันทั้งหมด คือต้องไม่มีสมาชิกร่วมกันเลย เราก็จะต้องอธิบายว่าทั้งสามนี้ต่างก็เป็น disjoint กันทุก classes

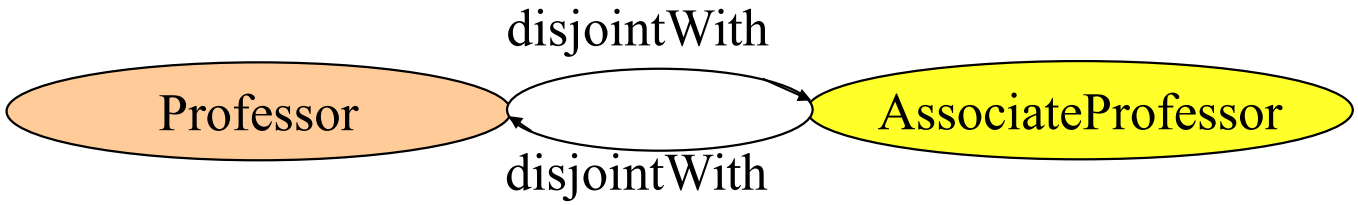


Inference engine จะไม่ยอมให้มีการกำหนดให้ instance หนึ่งๆ (:Mary) เป็นสมาชิกอยู่ใน class มากกว่า 1 class

```
:Professor a owl:Class.
:AssociateProfessor a owl:Class;
   owl:disjointWith :Professor.
:AsistantProfessor a owl:Class;
   owl:disjointWith :Professor,
                    :AssociateProfessor.
:David a :Professor.
:Willy a :Professor.
:Mary a :AssociateProfessor.
:Jane a :AssistantProfessor.
:Mary a :Professor.
```

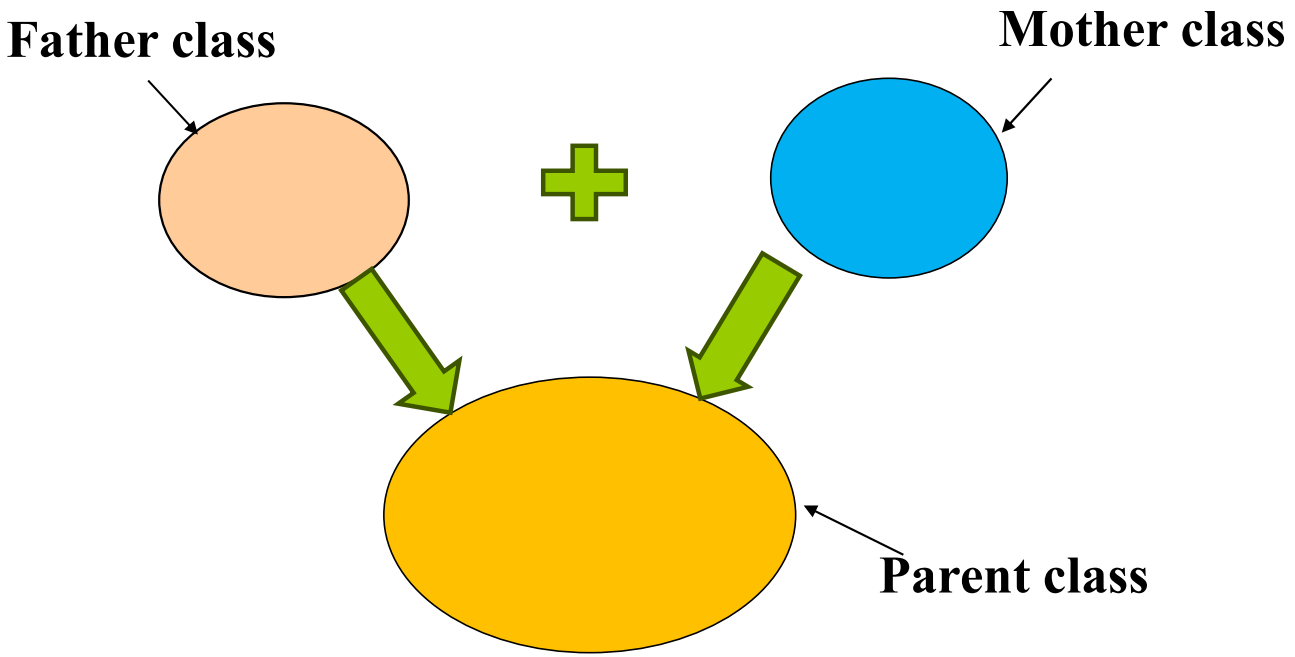
Note: disjointWith is a SymmetricProperty!

- Example: if Professor is disjointWith AssociateProfessor, then AssociateProfessor is disjointWith Professor.



Understanding owl:unionOf

- ถ้าต้องการอธิบายว่า class Parent เกิดจากการ union กันของ class Father และ class Mother โดย class Father นั้นจะต้อง disjoint กับ class Mother ด้วย



Understanding owl:unionOf

Use Protégé to create this ontology too.

```
@prefix : <http://example.com/owl/person/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

:Father a owl:Class.
:Mother a owl:Class;
   owl:disjointWith :Father.

:Parent a owl:Class;
   owl:unionOf (:Father :Mother).
```

เอาคนทั้งหมดมาเป็นพ่อ

ให้เดาว่ามีใครที่มี rdf:type เป็น Parent บ้าง

```
:David a :Father.
:Willy a :Father.
:Mary a :Mother.
:Jane a :Mother.
```

Understanding owl:oneOf

- จุดประสงค์ของการใช้ owl:oneOf คือการระบุว่ามี instances ตัวไหนบ้างที่เป็นสมาชิกของ class ที่เรากำหนด
- คำสั่งนี้จะมีประโยชน์เมื่อเราต้องการที่จะกำหนดให้ class ใด class หนึ่งมีจำนวนสมาชิกตามที่เรากำหนดไว้เท่านั้น โดยมีการกำหนดจำนวน instances ที่ชัดเจนไปเลย (fixed set of members) และจะต้องไม่มี instances อื่นที่นอกเหนือจากนี้อยู่ใน class นั้น

Understanding owl:oneOf

- ตัวอย่างเช่น กำหนดให้ class Day ประกอบด้วย instances ที่เป็นวัน 7 วันดังนี้

```
:Day a owl:Class;
      owl:oneOf (:Monday :Tuesday :Wednesday :Thursday :Friday :Saturday
:Sunday).
```

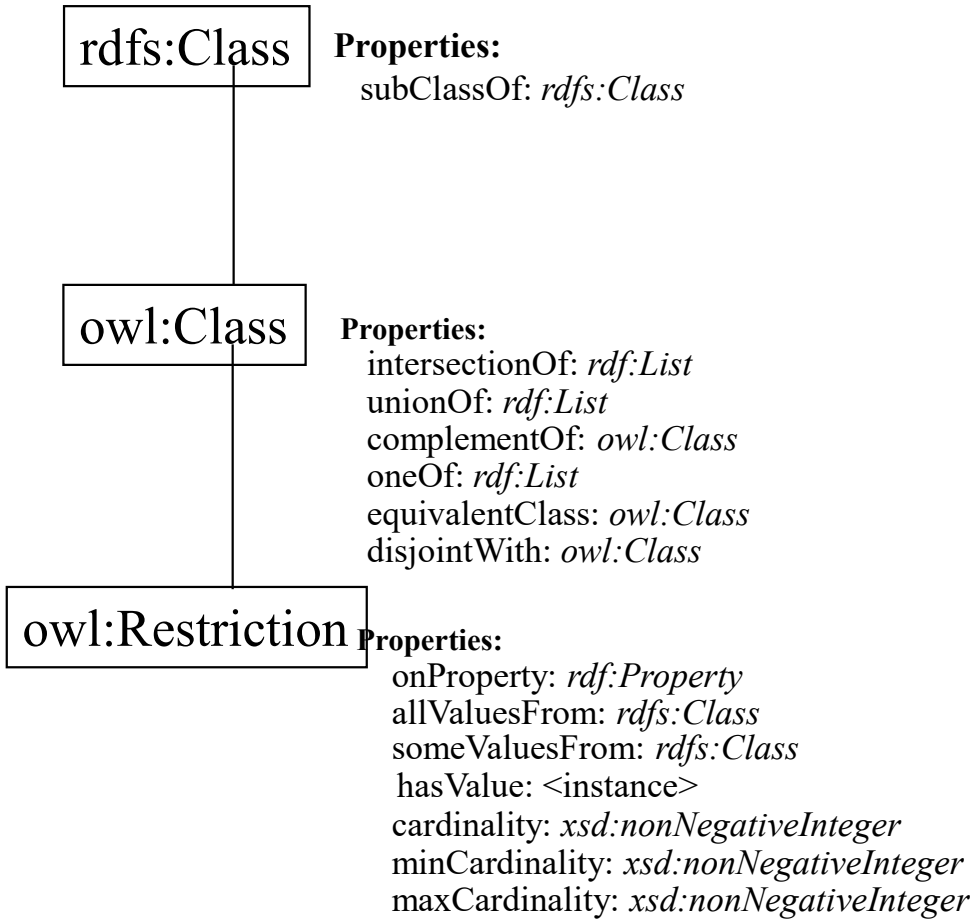
- หรือ class BobsChildren จะเป็น subclass of class Person และจะมีสมาชิกคือคน 3 คนต่อไปนี้เท่านั้น (ซึ่งเป็นลูกของ Bobs)

```
:Person a owl:Class.
:BobsChildren a owl:Class;
   rdfs:subClassOf :Person;
   owl:oneOf (:Bill :John :Mary).
```

ถ้ามีการถาม inference engine ว่า :Bill (หรือ :John หรือ :Mary) มี rdf:type เป็นอะไร ก็จะตอบว่า :BobsChildren นั่นเอง

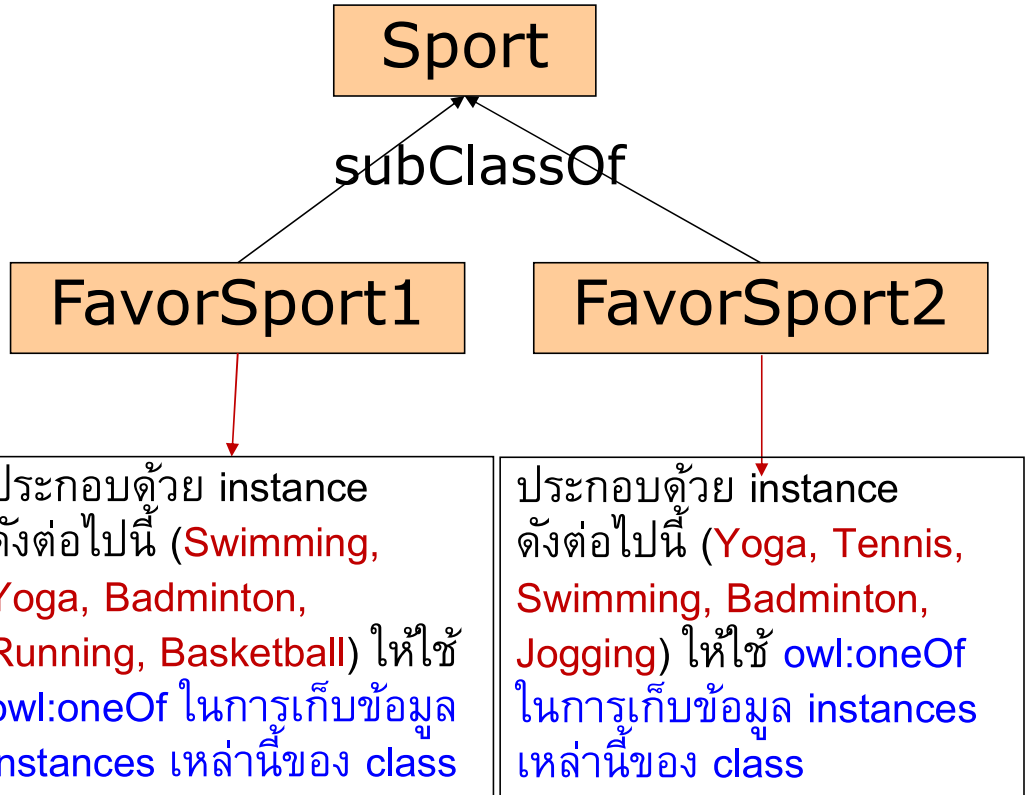
Use Protégé to create this ontology too.

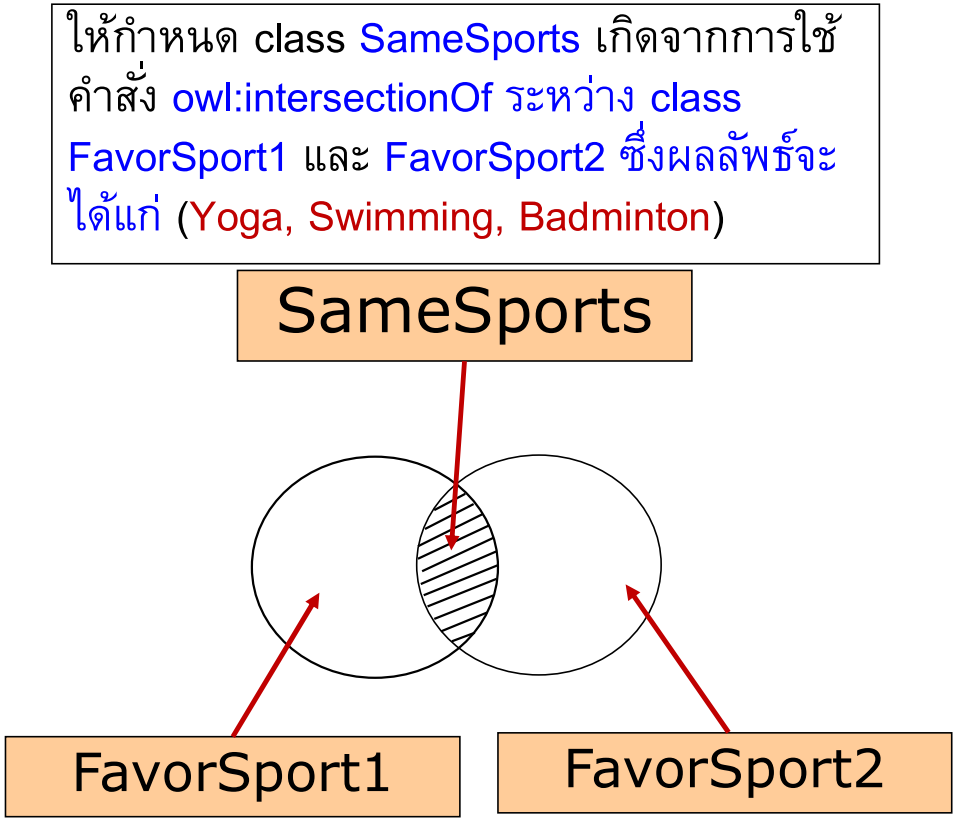
Summary of Class Properties



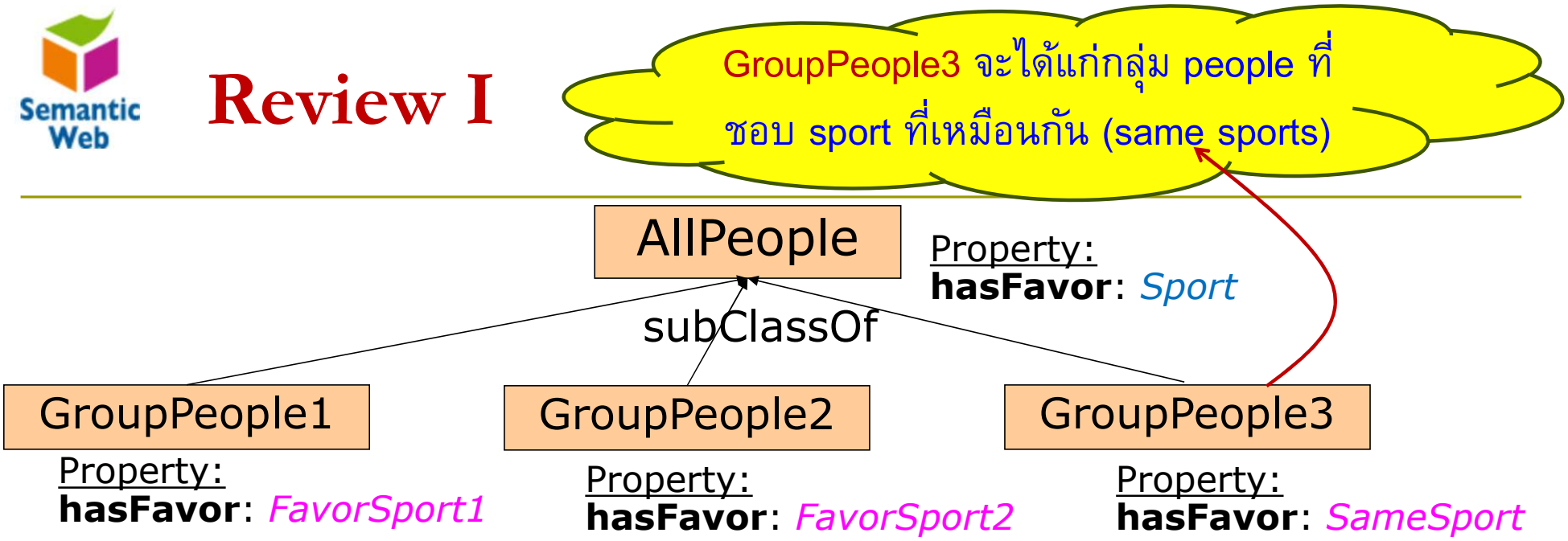
Review I

- ให้ทำการสร้าง ontology ตามโครงสร้าง taxonomy ต่อไปนี้





Review I



- ให้ทำการสร้าง classes และ properties ดังต่อไปนี้
- ให้ใช้ owl:someValuesFrom เพื่อควบคุม range ของ property hasFavor ที่ถูกใช้กับแต่ละ GroupPeople ดังตย.ข้างบน

Review I

Can you answer people who belong to GroupPeople3?

- **Create the following instances into the ontology:**

```
:Peter a :AllPeople;
  :hasFavor :Yoga.
:Robert a :AllPeople;
  :hasFavor :Badminton.
:Smith a :AllPeople;
  :hasFavor :Yoga.
:Katty a :AllPeople;
  :hasFavor :Swimming.
:Bob a :AllPeople;
  :hasFavor :Jogging.
:Marry a :AllPeople;
  :hasFavor :Jogging.
:John a :AllPeople;
  :hasFavor :Tennis.
```

```
:Bobby a :AllPeople;
  :hasFavor :Tennis.
:Harry a :AllPeople;
  :hasFavor :Swimming.
:Nancy a :AllPeople;
  :hasFavor :Badminton.
:Willy a :AllPeople;
  :hasFavor :Yoga.
:Randy a :AllPeople;
  :hasFavor :Yoga.
:Patty a :AllPeople;
  :hasFavor :Running.
:Cataryn a :AllPeople;
  :hasFavor :Basketball.
```

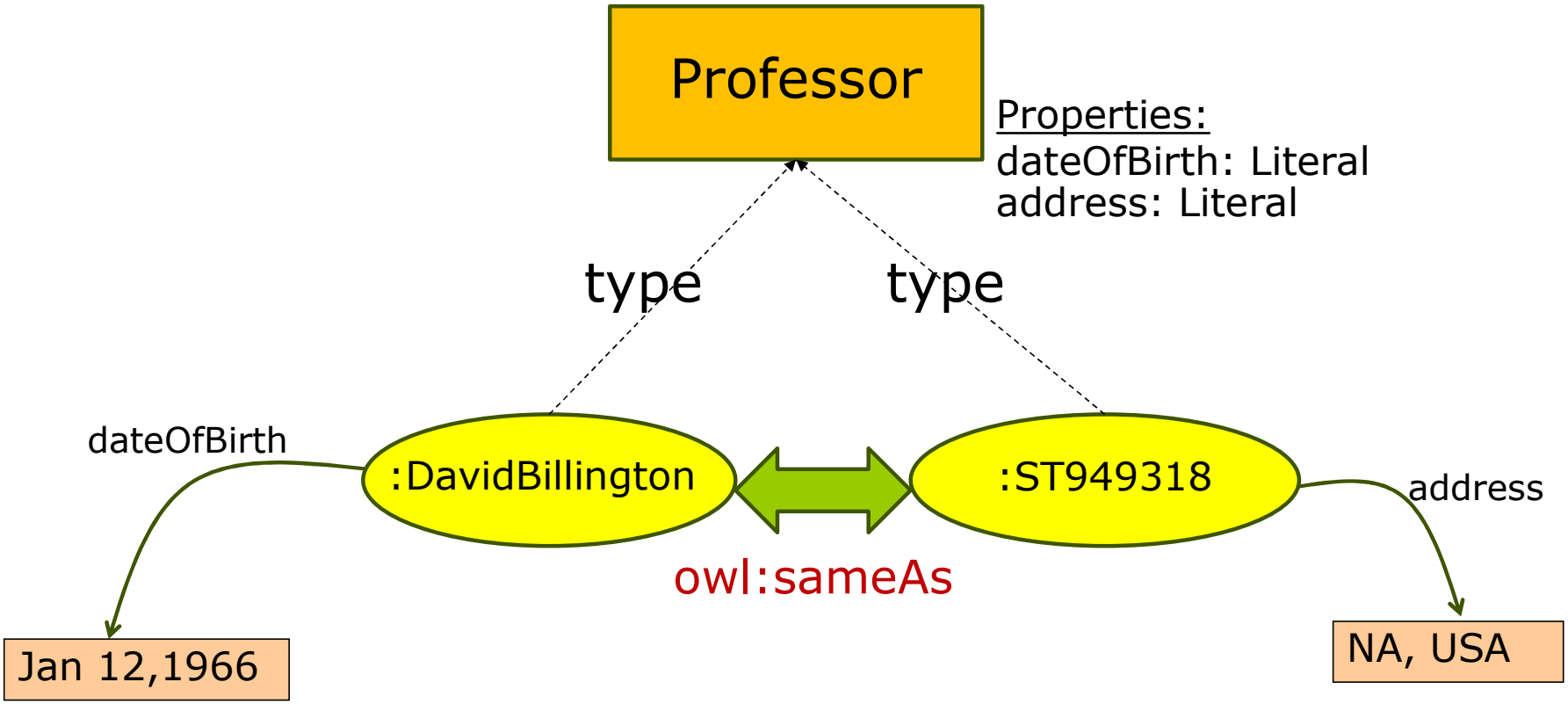
OWL statements that you can incorporate into instances

owl:sameAs

owl:differentFrom

owl:allDifferent

Indicating that two instances are the same



Indicating that two instances are the same

Use Protégé to create this ontology too.

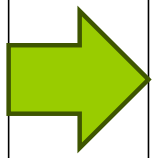
```
@prefix : <http://www.university.edu/staff/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .


:Professor a owl:Class.
:dateOfBirth a owl:DatatypeProperty;
    rdfs:domain :Professor;
    rdfs:range rdfs:Literal.
:address a owl:DatatypeProperty;
    rdfs:domain :Professor;
    rdfs:range rdfs:Literal.
```

```
:DavidBillington
    a :Professor;
    :dateOfBirth "Jan 12,1966".

:ST949318
    a :Professor;
    :address "NA, USA";
    owl:sameAs :DavidBillington.
```

สามารถใช้คำสั่ง owl:sameAs ในระดับของ instance level เพื่อระบุว่า instance :ST949318 จะเหมือนกับ instance :DavidBillington (คนสองคนนี้เป็นคนเดียวกัน)

owl:sameAs

```
:DavidBillington
    a :Professor;
    :dateOfBirth "Jan 12,1966".

:ST949318
    a :Professor;
    :address "NA, USA";
    owl:sameAs :DavidBillington.
```



```
:DavidBillington
    a :Professor;
    :dateOfBirth "Jan 12,1966";
    :address "NA, USA".

:ST949318
    a :Professor;
    :address "NA, USA";
    :dateOfBirth "Jan 12,1966";
```

การใช้คุณสมบัติ owl:sameAs จะทำให้ inference engine สามารถอนุมานได้ว่าคนสองคนนี้เป็นคนๆเดียวกัน ดังนั้น ถ้าแต่ละคนมีการกำหนด properties ต่างๆที่เป็นของตนเอง Properties เหล่านั้นก็จะถูกยุบรวมเข้าด้วยกัน เพราะถือว่าเป็นคนๆเดียวกัน ดังนั้น เราจึงสามารถ query หา date of birth ของคนรหัส :ST949318 ได้หรือ query หา address ของคนชื่อ :DavidBillington ได้ด้วยเช่นกัน

Indicating that two instances are different

```
:DavidBillington
    a :Professor;
    :dateOfBirth "Jan 12,1966".

:ST949318
    a :Professor;
    :address "NA, USA";
    owl:differentFrom :DavidBillington.
```

ในกรณีที่เราต้องการกำหนดว่า instance :ST949318 มีความแตกต่างกับ instance :DavidBillington จะสามารถทำได้โดยใช้คำสั่ง owl:differentFrom

owl:AllDifferent

- เรายังสามารถใช้คุณสมบัติ owl:AllDifferent class เพื่อเก็บ instances ทุกตัวที่มีความแตกต่างกัน

ตัวนี้เป็นการกำหนด blank node ขึ้นมาให้มี type เป็น owl:AllDifferent และใช้ owl:distinctMembers เพื่อระบุว่า Instances ต่อไปนี้ได้แก่ :DavidBillington, :DavidBill และ :BillDavid ต่างก็มีความแตกต่างกันทุกตัว

```
:DavidBillington a :Professor.
:DavidBill a :Professor.
:BillDavid a :Professor.

_:x1 a owl:AllDifferent;
   owl:distinctMembers (:DavidBillington :DavidBill :BillDavid).
```

Summary of the different statements you can incorporate into instances

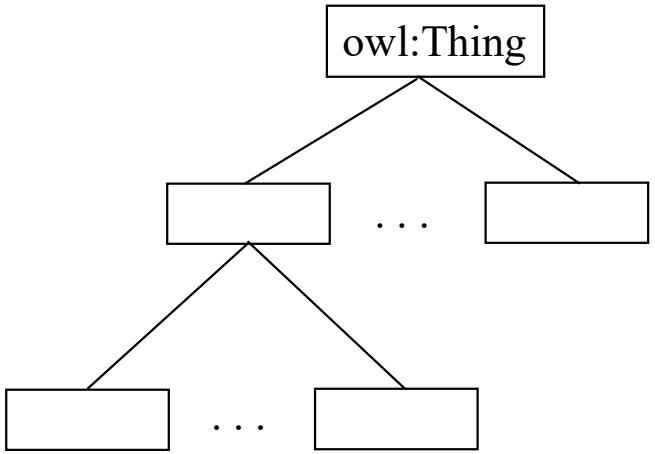
- สรุปว่าคำสั่งต่อไปนี้ จะเป็นคำสั่งที่สามารถถูกใช้อยู่ภายใน instance level ได้เลย
  - owl:sameAs
  - owl:differentFrom
  - owl:AllDifferent

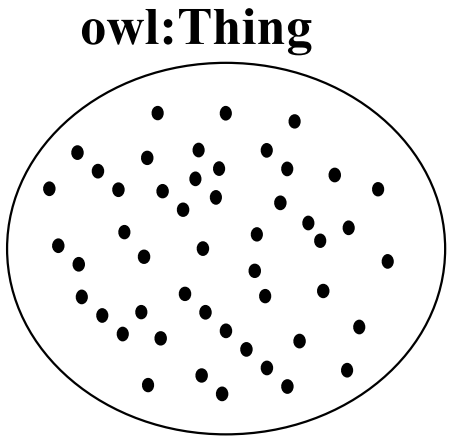
The owl:Thing class is the root of all classes

owl:Thing

The owl:Thing class is the root of all classes

- owl:Thing จะเป็นที่รวมของ classes ทุก classes ที่เราสร้างขึ้นมา ซึ่ง classes ทุก classes นี้จะต้องเป็น subclass ของ owl:Thing ทั้งหมด ดังนั้น instances ของทุก classes ก็จะเป็นสมาชิกของ owl:Thing ด้วย

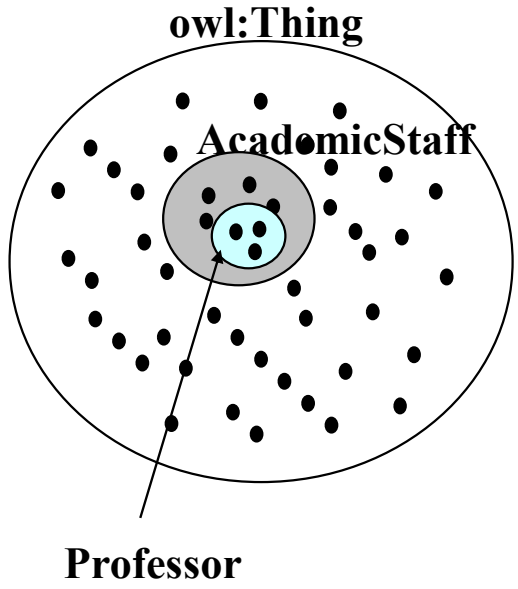




ทุกๆ instances ในทุก classes จะเป็น instances ของ owl:Thing!

owl:Thing

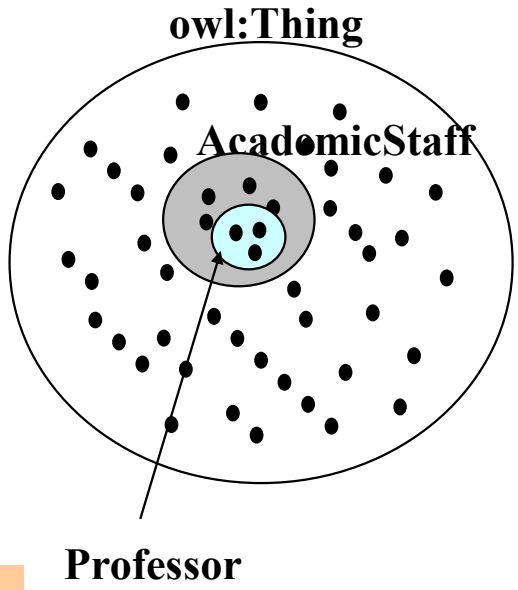
```
@base <http://www.university.edu/staff/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

:Professor a owl:Class;
        rdfs:subClassOf :AcademicStaff.
```



Class Professor และ class AcademicStaff ต่างก็ถูกอนุมานว่าเป็น subclasses ของ owl:Thing ด้วย

owl:Thing

```
@base <http://www.university.edu/staff/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

:Professor a owl:Class;
        rdfs:subClassOf :AcademicStaff.


:ST949318 a :Professor.
```



ดังนั้น ถ้า ST949318 มี type เป็น Professor, instance ST949318 ก็จะเป็น instance ของ AcademicStaff และเป็น instance ของ owl:Thing ด้วย

Importing other OWL documents

owl:Ontology
owl:imports

owl:Ontology Class

- owl:Ontology class จะเป็นอีก class ที่ถูกสร้างขึ้นมาเพื่อใช้อธิบาย ontology โดยผ่านทาง properties ต่างๆ รวมไปถึงการ import ontology จากที่อื่นๆด้วย

owl:Ontology

Properties:
imports:
versionInfo:
priorVersion: *Ontology*
incompatibleWith: *Ontology*
backwardCompatibleWith: *Ontology*

Note: *Ontology* นี้จะเหมายถึง OWL document เช่น university.ttl.

The Ontology Header

person.ttl

```
@prefix : <http://www.mydomain.com/person/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

:person a owl:Ontology;
     owl:imports <file:/D:/SemanticWeb/Content/test/equivalentClass.ttl>.

:388888888
    a :Woman.
:488888888
    a :Woman.
```

You can also specify physical URL such as:
<https://computing.kku.ac.th/staff/myfile.ttl>

ตัวอย่างนี้จะเป็นการ import ไฟล์ชื่อ “equivalentClass.ttl” เข้ามายังไฟล์ ontology ปัจจุบัน (person.ttl) ซึ่งจะเป็นการรวมไฟล์ ontologies ทั้งสองไฟล์นี้เข้าเป็นไฟล์เดียวกัน

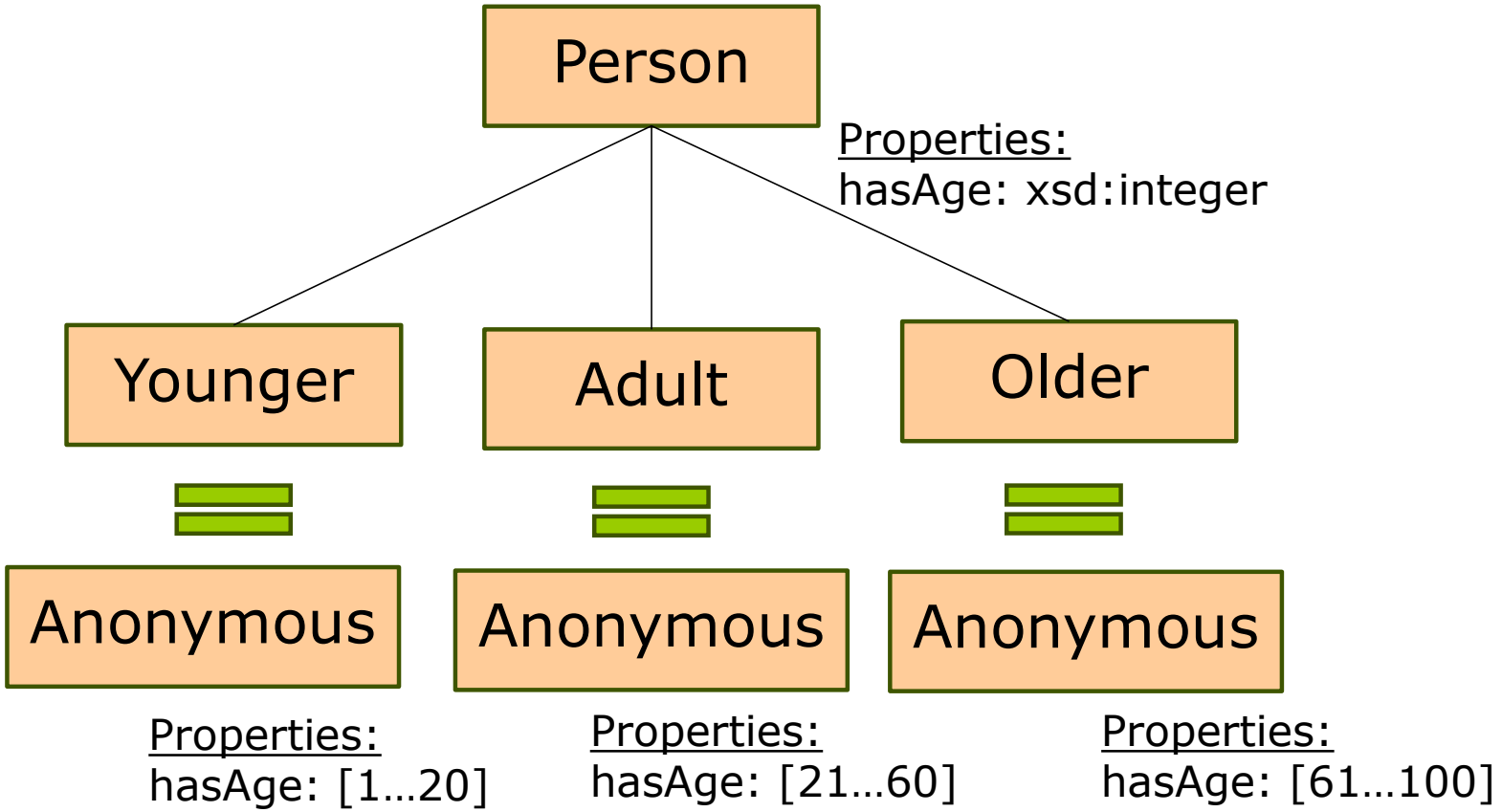
Additional Example

owl:DataRange

owl:onDatatype

owl:withRestrictions

Classifying Groups of Data



Classifying Groups of Data

```
@prefix : <http://www.example.com/person/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

:Person a owl:Class.
:hasAge a owl:DatatypeProperty;
    rdfs:domain :Person;
    rdfs:range xsd:integer.

:Younger a owl:Class;
   rdfs:subClassOf :Person;
   owl:equivalentClass
  [ a owl:Restriction;
     owl:onProperty :hasAge;
     owl:someValuesFrom
     [ a owl:DataRange;
      owl:onDatatype  xsd:integer;
      owl:withRestrictions ( [xsd:minInclusive 1] [xsd:maxInclusive 20] )
     ];
].
```

Use Protégé to create this ontology too.

Classifying Groups of Data

```
:Adult a owl:Class;
   rdfs:subClassOf :Person;
   owl:equivalentClass
  [ a owl:Restriction;
     owl:onProperty :hasAge;
     owl:someValuesFrom
     [ a owl:DataRange;
      owl:onDatatype  xsd:integer;
      owl:withRestrictions ( [xsd:minInclusive 21] [xsd:maxInclusive 60] )
     ];
].

:Older a owl:Class;
   rdfs:subClassOf :Person;
   owl:equivalentClass
  [ a owl:Restriction;
     owl:onProperty :hasAge;
     owl:someValuesFrom
     [ a owl:DataRange;
      owl:onDatatype  xsd:integer;
      owl:withRestrictions ( [xsd:minInclusive 61] [xsd:maxInclusive 100] )
     ];  ].
```

Classifying Groups of Data

```
:David :hasAge "25"^^xsd:int.
:Marry :hasAge "30"^^xsd:int.
:John :hasAge "10"^^xsd:int.
:Willy :hasAge "8"^^xsd:int.
:Max :hasAge "61"^^xsd:int.
:Bob :hasAge "90"^^xsd:int.
:Smith :hasAge "40"^^xsd:int.
```

จากตัวอย่างนี้ จะทำให้เราสามารถ query ถาม inference engine ว่ามีใคร (instances ไหนบ้าง) ที่อยู่ class Younger (Willy, and John), มีใครบ้าง (instances ไหนบ้าง)ที่อยู่ใน class Adult (David, Marry, and Smith) และมีใครบ้าง (instances ไหนบ้าง) ที่อยู่ใน class Older (Max, and Bob) ซึ่งการ query จะใช้ SPARQL ที่เราจะได้เรียนในบทถัดไป

Reference

- OWL
  - https://www.w3.org/OWL/
  - https://www.w3.org/TR/owl-guide/
  - https://en.wikipedia.org/wiki/Web_Ontology_Language
  - https://cambridgesemantics.com/blog/semantic-university/learn-owl-rdfs/owl-references-humans/
  - https://www.w3.org/2007/OWL/wiki/Quick_Reference_Guide
- Turtle
  - https://www.w3.org/TR/turtle/
  - https://www.w3.org/2007/OWL/wiki/PrimerExampleTurtle
- Tools
  - https://www.w3.org/wiki/SemanticWebTools
- Semantic Web
  - https://www.w3.org/2001/sw/