OWL
-Web Ontology Language

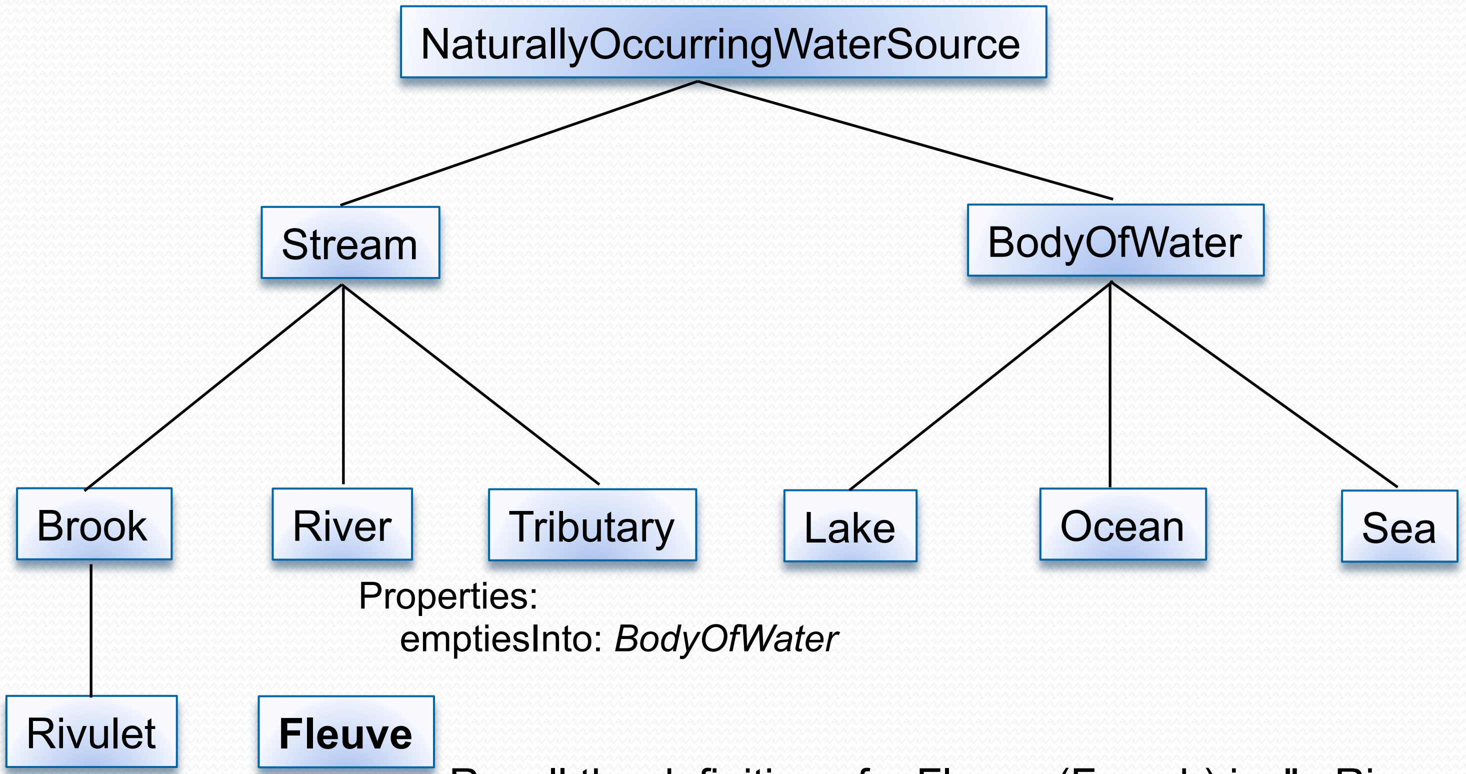
Part 2

Using OWL to Define Classes

Constructing Classes using Set Operators

- OWL gives you the ability to construct classes using these set operators:
  - intersectionOf
  - unionOf
  - complementOf

Defining a Fleuve class using the intersectionOf operator



Recall the definition of a Fleuve (French) is: "a River which emptiesInto a Sea". Thus, a Fleuve may be defined as the intersectionOf the River class and an anonymous class containing the emptiesInto property with allValuesFrom Sea.

Defining Fleuve

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
             xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
             xmlns:owl="http://www.w3.org/2002/07/owl#"
             xml:base="http://www.geodesy.org/water/naturally-occurring">

    <owl:Class rdf:ID="Fleuve">
        <owl:intersectionOf rdf:parseType="Collection">
            <owl:Class rdf:about="#River"/>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#emptiesInto"/>
                <owl:allValuesFrom rdf:resource="#Sea"/>
            </owl:Restriction>
        </owl:intersectionOf>
    </owl:Class>

   ...

</rdf:RDF>
```

naturally-occurring.owl (snippet)

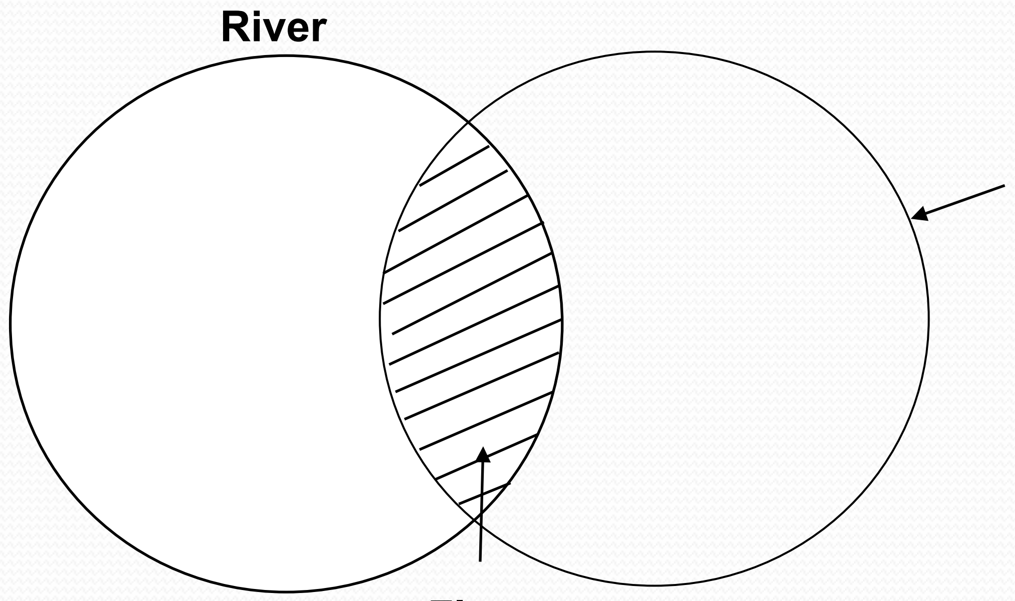
Understanding intersectionOf

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
               xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
               xmlns:owl="http://www.w3.org/2002/07/owl#"
               xml:base="http://www.geodesy.org/water/naturally-occurring">

     <owl:Class rdf:ID="Fleuve">
          <owl:intersectionOf rdf:parseType="Collection">
               <owl:Class rdf:about="#River"/>
               <owl:Restriction>
                       <owl:onProperty rdf:resource="#emptiesInto"/>
                       <owl:allValuesFrom rdf:resource="#Sea"/>
               </owl:Restriction>
          </owl:intersectionOf>
     </owl:Class>
    …
</rdf:RDF>
```

This is read as: "The Fleuve class is the intersection of the River class and an anonymous class that contains a property emptiesInto and all values are instances of Sea."

Here's an easier way to read this: "The Fleuve class is a River that emptiesInto a Sea."

Understanding intersectionOf


River
The members of this anonymous class are instances which have an emptiesInto property in which all values are instances of Sea.

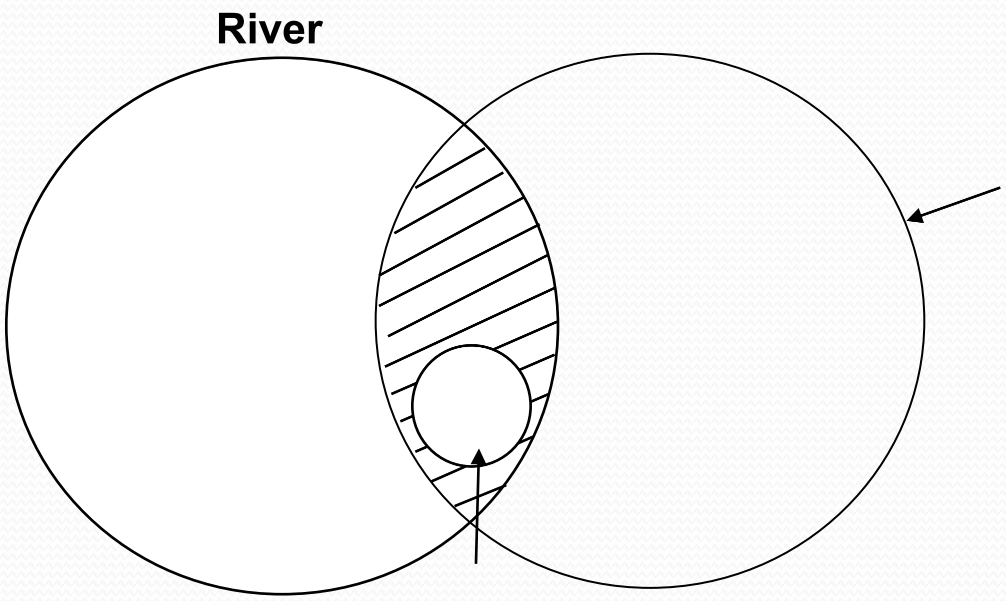
Fleuve- a River that emptiesInto a Sea.

Contrast with defining Fleuve using 2 subClassOf statements

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
            xmlns:owl="http://www.w3.org/2002/07/owl#"
            xml:base="http://www.geodesy.org/water/naturally-occurring">

   <owl:Class rdf:ID="Fleuve">
      <rdfs:subClassOf rdf:resource="#River"/>
      <rdfs:subClassOf>
          <owl:Restriction>
               <owl:onProperty rdf:resource="#emptiesInto"/>
               <owl:allValuesFrom rdf:resource="#Sea"/>
          </owl:Restriction>
      </rdfs:subClassOf>
   </owl:Class>


  ...

</rdf:RDF>
```

naturally-occurring.owl (snippet)

Multiple subClassOf = a subset of the intersection



The members of this anonymous class are instances which have an emptiesInto property in which all values are instances of Sea.

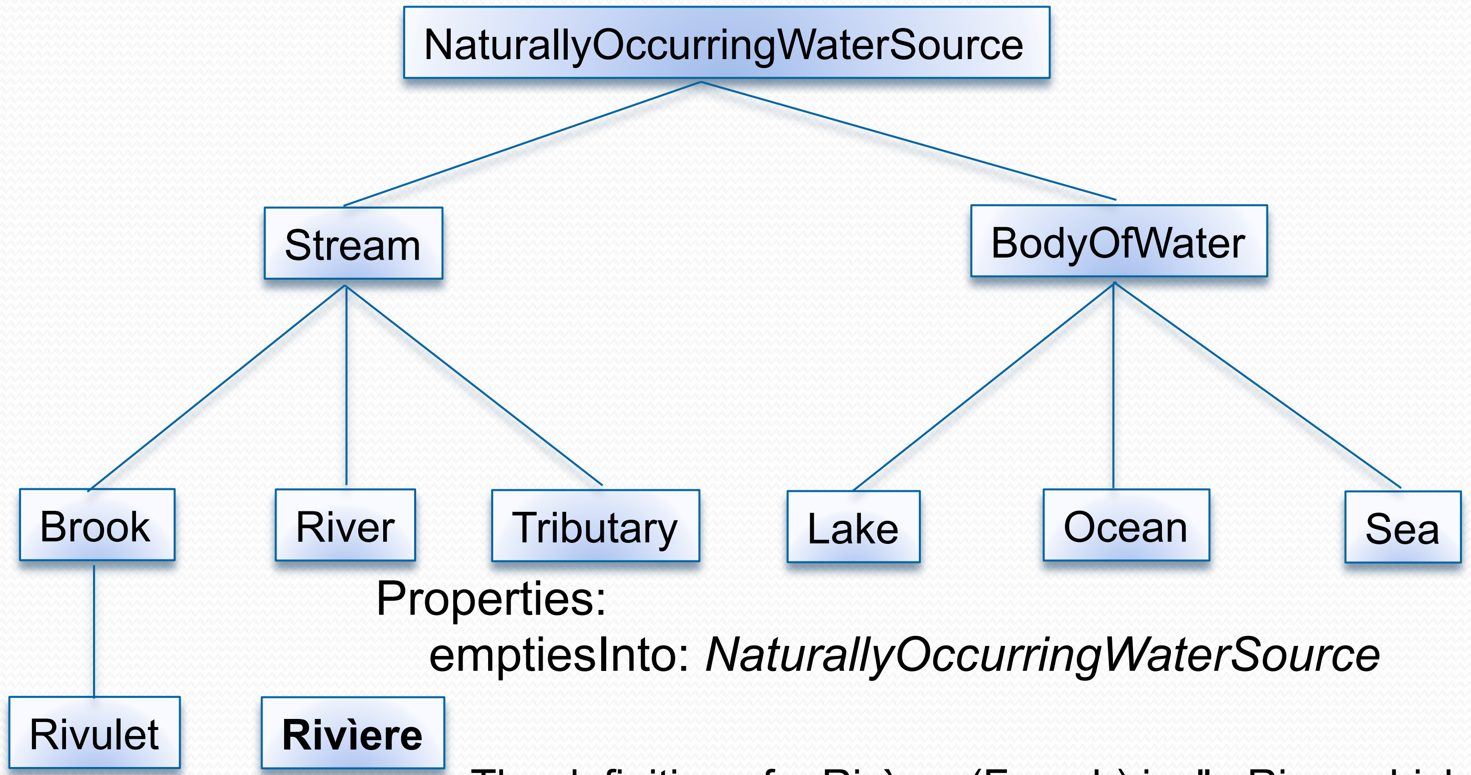
Fleuve- a Fleuve is a River that emptiesInto a Sea. However, as this diagram shows, by using multiple subClassOf elements there may be Rivers which emptiesInto a Sea that are not Fleuves.

The conjunction (AND) of two subClassOf statements is a subset of the intersection of the classes.

Contrast

- Defining a Fleuve using two subClassOf elements: all instances of Fleuve must be a River and emptiesInto Sea.
- Defining a Fleuve using intersectionOf: a Fleuve is the collection of all instances that is both a River and emptiesInto Sea.
- Thus, the subClassOf form merely characterizes a Fleuve is, whereas the intersectionOf form defines a Fleuve.

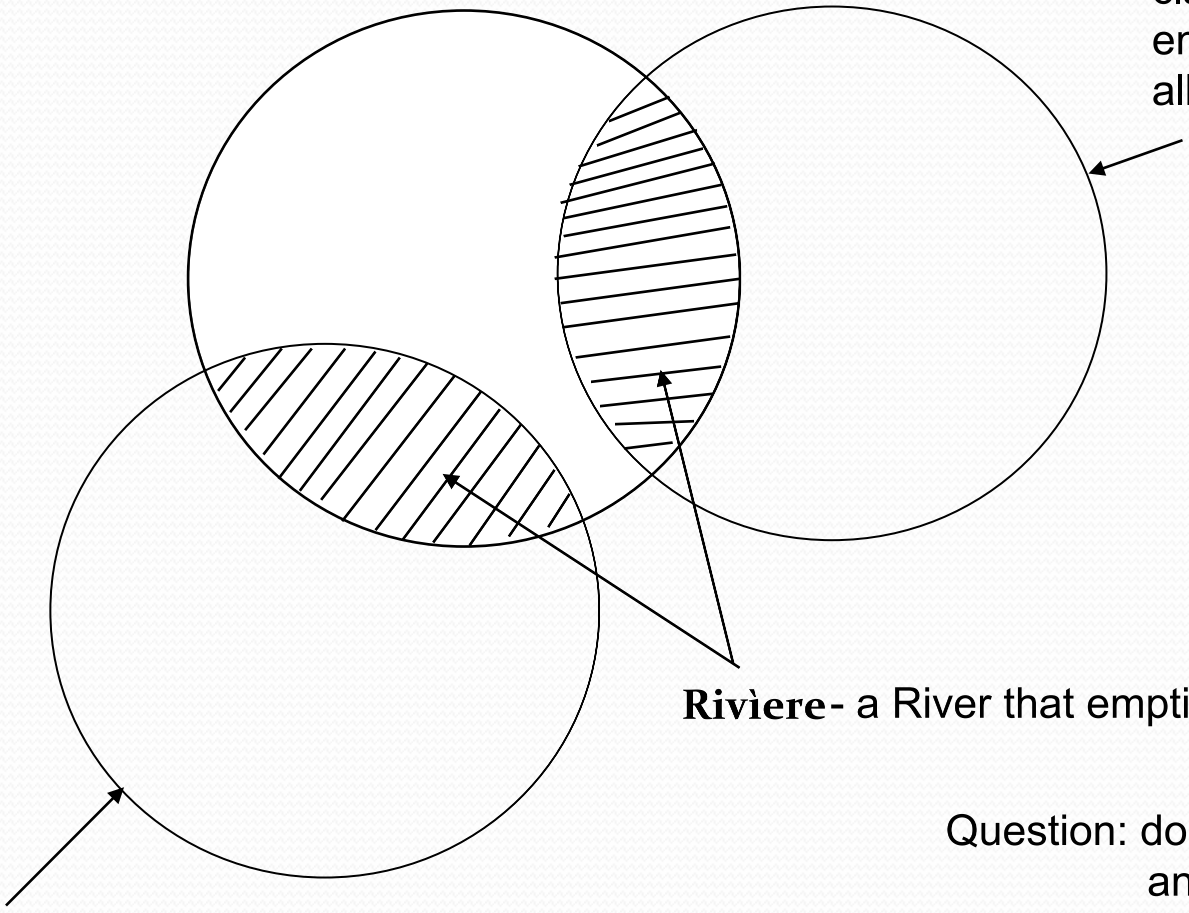
Defining a Rivìere class using the unionOf operator



Properties:
emptiesInto: *NaturallyOccurringWaterSource*

The definition of a Rivìere (French) is: "a River which emptiesInto a Lake or another River". Thus, to define a Rivìere we will need to use both intersectionOf and unionOf …
Note: the range for emptiesInto has been changed for this example.

A Rivìere is the intersection of River with the union of two classes



The members of this anonymous class are instances which have an emptiesInto property in which all values are instances of **Lake**.

Rivìere - a River that emptiesInto a Lake or another River.

The members of this anonymous class are instances which have an emptiesInto property in which all values are instances of **River**.

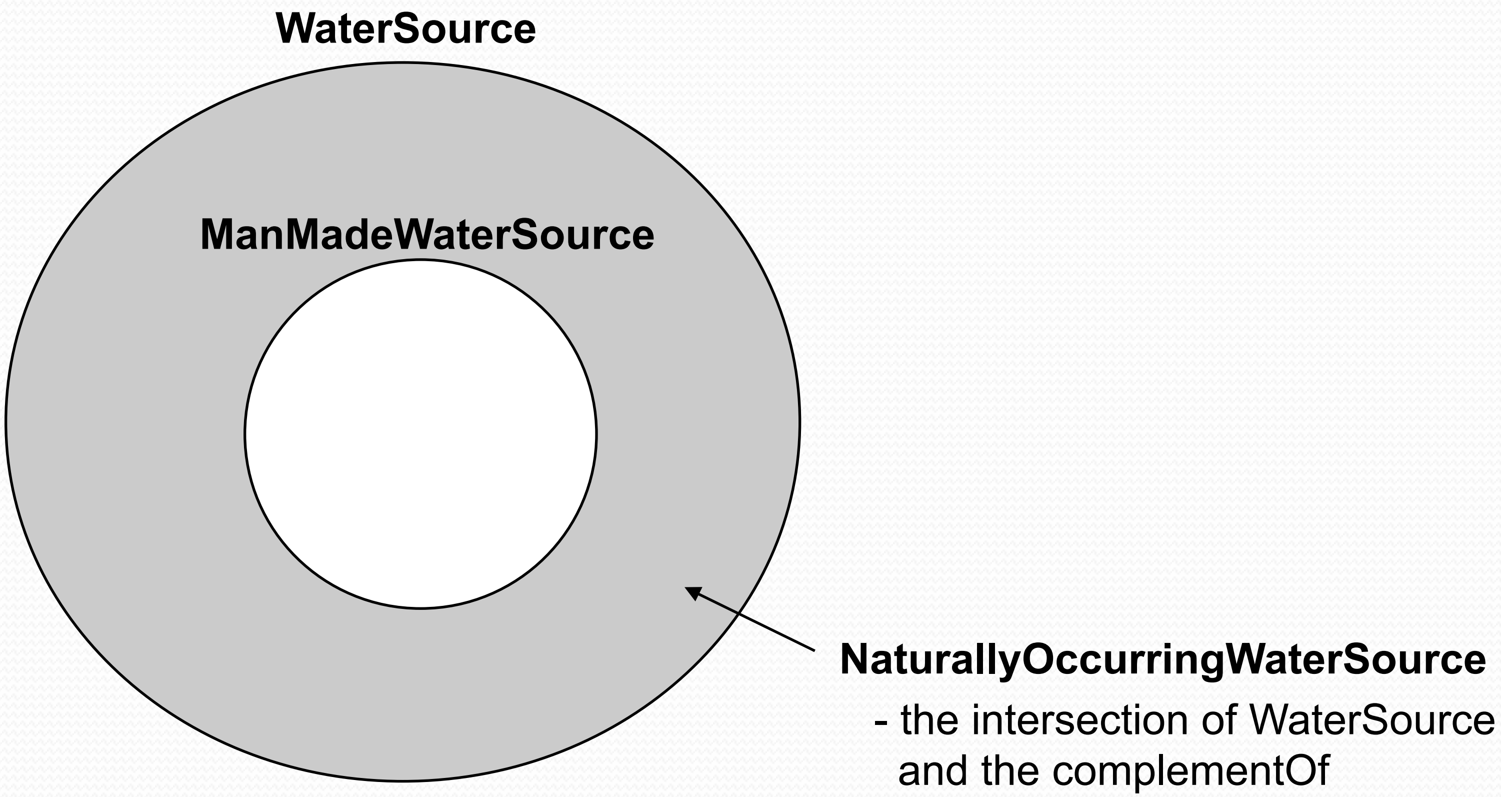
Question: do you understand why the two anonymous classes are disjoint?
Answer: because emptiesInto is a Functional Property!

Defining Rivìere

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
            xmlns:owl="http://www.w3.org/2002/07/owl#"
            xml:base="http://www.geodesy.org/water/naturally-occurring">
    <owl:Class rdf:ID="Rivìere">
        <owl:intersectionOf rdf:parseType="Collection">
            <owl:Class rdf:about="#River"/>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#emptiesInto"/>
                        <owl:allValuesFrom rdf:resource="#Lake"/>
                    </owl:Restriction>
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#emptiesInto"/>
                        <owl:allValuesFrom rdf:resource="#River"/>
                    </owl:Restriction>
                </owl:unionOf>
            </owl:Class>
        </owl:intersectionOf>
    </owl:Class>
    ...

</rdf:RDF>
```

naturally-occurring.owl (snippet)

Defining NaturallyOccurringWaterSource using complementOf



- the intersection of WaterSource and the complementOf ManMadeWaterSource.

Using complementOf

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
                xmlns:owl="http://www.w3.org/2002/07/owl#"
                xml:base="http://www.geodesy.org/water/naturally-occurring">

     <owl:Class rdf:ID="NaturallyOccurringWaterSource">
          <owl:intersectionOf rdf:parseType="Collection">
               <owl:Class rdf:about="#WaterSource"/>
               <owl:Class>
                    <owl:complementOf rdf:resource="#ManMadeWaterSource"/>
               </owl:Class>
          </owl:intersectionOf>
     </owl:Class>

    ...

</rdf:RDF>
```

naturally-occurring.owl (snippet)

Enumeration, equivalence, disjoint

- OWL gives you the ability to:
  - construct a class by enumerating its instances.
  - specify that a class is equivalent to another class.
  - specify that a class is disjoint from another class.

Defining a class by enumerating its instances

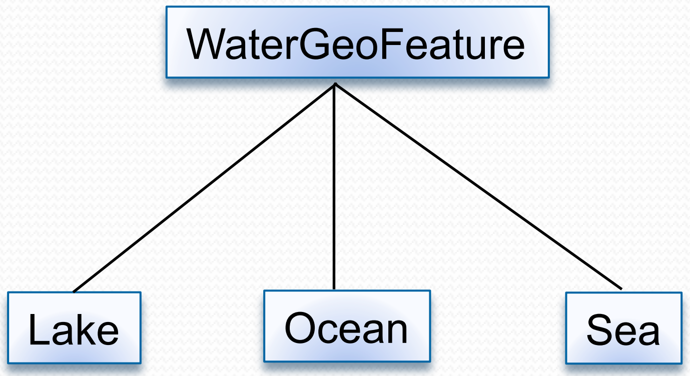
Here we are enumerating the Rivers which are protected by the Kyoto Treaty.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:owl="http://www.w3.org/2002/07/owl#"
         xmlns:geo="http://www.geodesy.org/water/naturally-occurring#"
         xml:base="http://www.geodesy.org/water/naturally-occurring">

   <owl:Class rdf:ID="Kyoto-Protected-River">
      <rdfs:subClassOf rdf:resource="#River"/>
      <owl:oneOf rdf:parseType="Collection">
         <geo:River rdf:about="http://www.china.org/geography/rivers#Yangtze"/>
         <geo:River rdf:about="http://www.us.org/rivers#Mississippi"/>
         <geo:River rdf:about="http://www.africa.org/rivers#Nile"/>
         <geo:River rdf:about="http://www.s-america.org/rivers#Amazon"/>
         …
      </owl:oneOf>
   </owl:Class>

  ...

</rdf:RDF>
```

naturally-occurring.owl (snippet)

Defining a class to be equivalent to another class

- owl:equivalentClass is used to state that a class is equivalent to another class.
- Example: suppose that another OWL document defines a class called WaterGeoFeature as follows:



We would like to state that BodyOfWater is equivalent to WaterGeoFeature.

Defining BodyOfWater to be equivalent to WaterGeoFeature

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
             xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
             xmlns:owl="http://www.w3.org/2002/07/owl#"
             xml:base="http://www.geodesy.org/water/naturally-occurring">

    <owl:Class rdf:ID="BodyOfWater">
        <rdfs:subClassOf rdf:resource="#NaturallyOccurringWaterSource"/>
        <owl:equivalentClass rdf:resource="http://www.other.org#WaterGeoFeature"/>
    </owl:Class>

   ...

</rdf:RDF>
```

naturally-occurring.owl (snippet)

Defining a class to be disjoint from another class

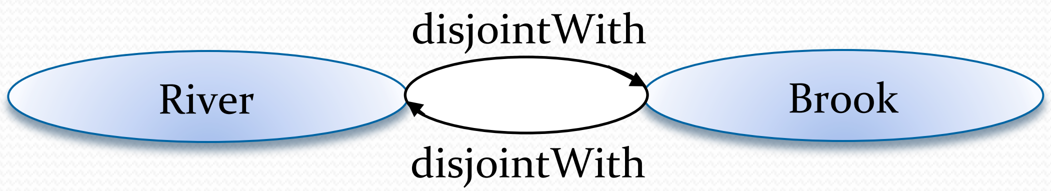
```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:owl="http://www.w3.org/2002/07/owl#"
         xml:base="http://www.geodesy.org/water/naturally-occurring">

   <owl:Class rdf:ID="River">
      <rdfs:subClassOf rdf:resource="#Stream"/>
      <owl:disjointWith rdf:resource="#Brook"/>
      <owl:disjointWith rdf:resource="#Rivulet"/>
      <owl:disjointWith rdf:resource="#Tributary"/>
   </owl:Class>
  …
</rdf:RDF>
```

naturally-occurring.owl (snippet)

This definition of River indicates that a River instance cannot also be a Brook, Rivulet, or Tributary. Thus, for example, you cannot have an instance which defines the Yangtze as a Tributary.

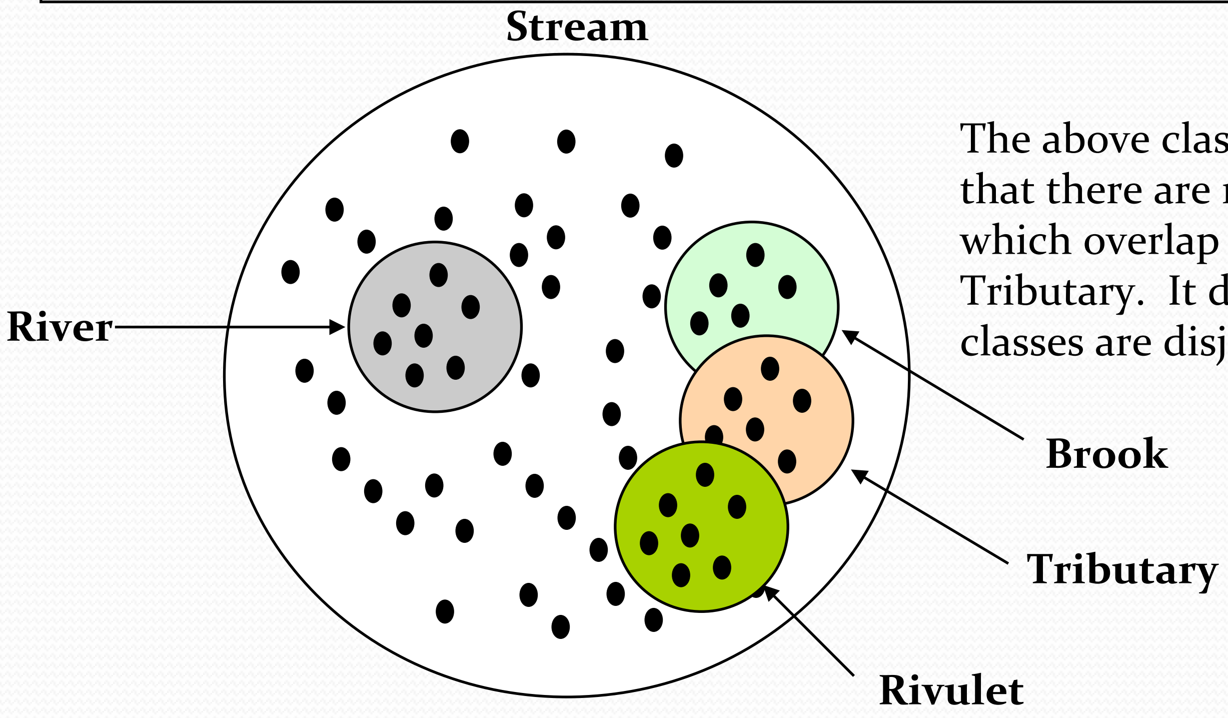
Note: disjointWith is a SymmetricProperty!

- Example: if River is disjointWith Brook, then Brook is disjointWith River.



River is (only) disjoint from the others

```
<owl:Class rdf:ID="River">
        <rdfs:subClassOf rdf:resource="#Stream"/>
        <owl:disjointWith rdf:resource="#Brook"/>
        <owl:disjointWith rdf:resource="#Rivulet"/>
        <owl:disjointWith rdf:resource="#Tributary"/>
</owl:Class>
```



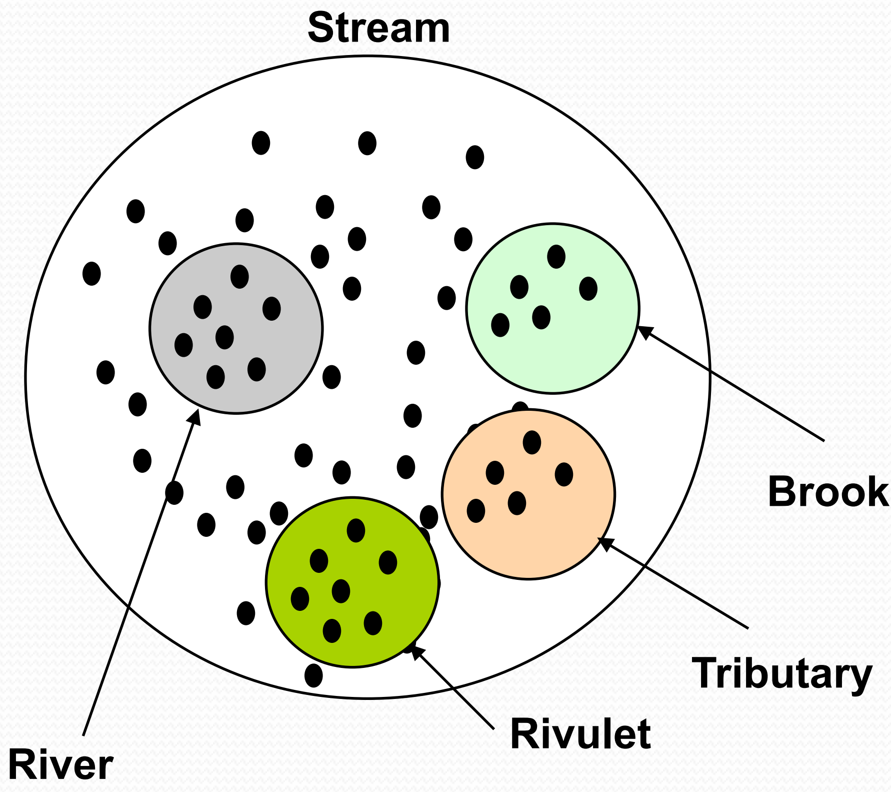
The above class definition only states that there are no instances of River which overlap with Brook, Rivulet, or Tributary. It does not state that all four classes are disjoint.

Now we know that all are disjoint

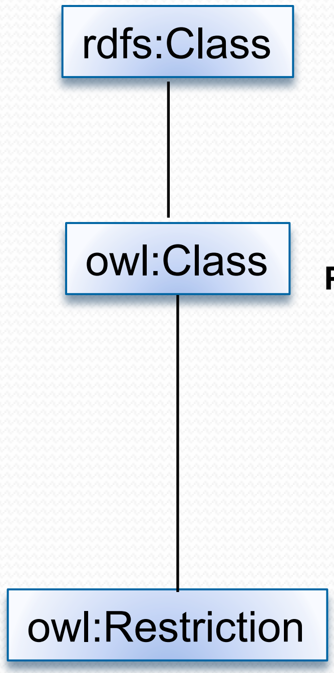
```
<owl:Class rdf:ID="River">
        <rdfs:subClassOf rdf:resource="#Stream"/>
        <owl:disjointWith rdf:resource="#Brook"/>
        <owl:disjointWith rdf:resource="#Rivulet"/>
        <owl:disjointWith rdf:resource="#Tributary"/>
</owl:Class>
```

```
<owl:Class rdf:ID="Brook">
        <rdfs:subClassOf rdf:resource="#Stream"/>
        <owl:disjointWith rdf:resource="#Rivulet"/>
        <owl:disjointWith rdf:resource="#Tributary"/>
</owl:Class>
```

```
<owl:Class rdf:ID="Tributary">
        <rdfs:subClassOf rdf:resource="#Stream"/>
        <owl:disjointWith rdf:resource="#Rivulet"/>
</owl:Class>
```



Hierarchy



Properties:
subClassOf: *rdfs:Class*

Properties:
intersectionOf: *rdf:List*
unionOf: *rdf:List*
complementOf: *owl:Class*
oneOf: *rdf:List*
equivalentClass: *owl:Class*
disjointWith: *owl:Class*

Properties:
onProperty: *rdf:Property*
allValuesFrom: *rdfs:Class*
hasValue:
someValuesFrom: *rdfs:Class*
cardinality: *xsd:nonNegativeInteger*
minCardinality: *xsd:nonNegativeInteger*
maxCardinality: *xsd:nonNegativeInteger*

OWL statements that you can incorporate into instances

Indicating that two instances are the same

Consider these two instance documents:

```
<?xml version="1.0"?>
<Sea rdf:ID="EastChinaSea"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns="http://www.geodesy.org/water/naturally-occurring#">
    ...
</Sea>
```

```
<?xml version="1.0"?>
<Sea rdf:ID="S1001-x-302"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns="http://www.geodesy.org/water/naturally-occurring#">
    ...
</Sea>
```

Are they referring to the same Sea? In fact, S1001-x-302 is the catalog number for the East China Sea. So, these two instances do refer to the same Sea. It would be useful if we could state in an instance document that it is describing the same thing as another instance document. We use owl:sameIndividualAs to express this sameness ...

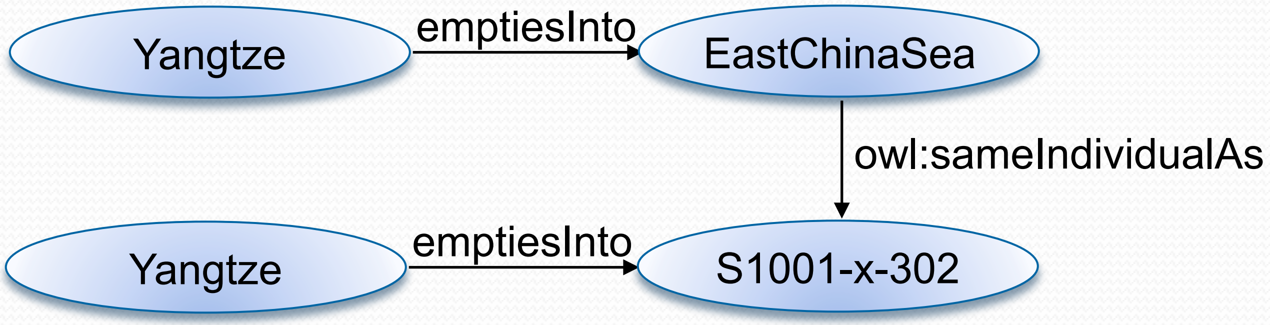
owl:sameIndividualAs

```
<?xml version="1.0"?>
<Sea rdf:ID="EastChinaSea"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:owl="http://www.w3.org/2002/07/owl#"
        xmlns="http://www.geodesy.org/water/naturally-occurring#">
    <owl:sameIndividualAs rdf:resource="http://www.national-geographic.org#S1001-x-302"/>
     ...
</Sea>
```

We are clearly indicating that this instance is describing the same thing as the S1001-x-302 instance.

owl:FunctionalProperty and owl:sameIndividualAs can reinforce each other!

```
<?xml version="1.0"?>
<River rdf:ID="Yangtze"
          xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns="http://www.geodesy.org/water/naturally-occurring#">
    <emptiesInto rdf:resource="http://www.china.org/geography#EastChinaSea"/>
    <emptiesInto rdf:resource="http://www.national-geographic.org#S1001-x-302"/>
</River>
```



By defining emptiesInto as a FunctionalProperty we assert that EastChinaSea and S1001-x-302 must be the same. The owl:sameIndividualAs is reconfirming this!

Indicating that two instances are different

Consider these two instance documents:

```
<?xml version="1.0"?>
<Sea rdf:ID="EastChinaSea"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns="http://www.geodesy.org/water/naturally-occurring#">
    ...
</Sea>
```

```
<?xml version="1.0"?>
<Sea rdf:ID="RedSea"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns="http://www.geodesy.org/water/naturally-occurring#">
    ...
</Sea>
```

It may be useful to clearly state in an instance document that it is different from another instance. This is accomplished using owl:differentFrom.

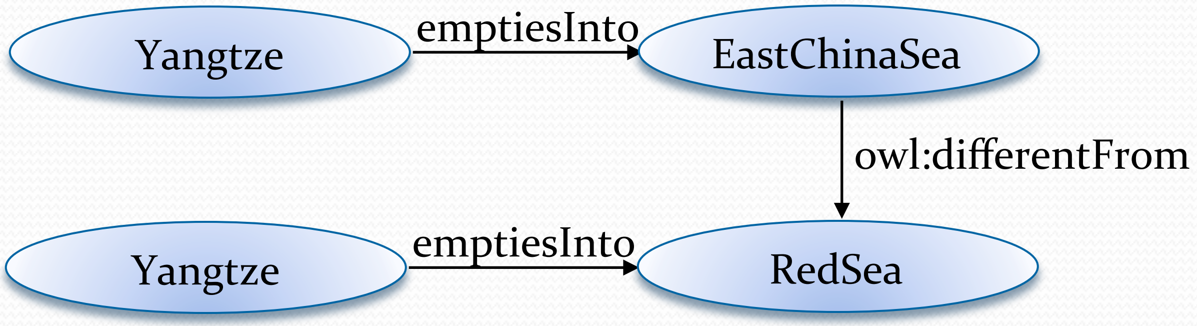
owl:differentFrom

```
<?xml version="1.0"?>
<Sea rdf:ID="EastChinaSea"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:owl="http://www.w3.org/2002/07/owl#"
        xmlns="http://www.geodesy.org/water/naturally-occurring#">
   <owl:differentFrom rdf:resource="http://
www.mediterranean.org#RedSea"/>
    ...
</Sea>
```

We are clearly indicating that this instance is different from the Red Sea instance.

owl:FunctionalProperty combined with owl:differentFrom can expose contradictions!

```
<?xml version="1.0"?>
<River rdf:ID="Yangtze"
          xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns="http://www.geodesy.org/water/naturally-occurring#">
    <emptiesInto rdf:resource="http://www.china.org/geography#EastChinaSea"/>
    <emptiesInto rdf:resource="http://www.mediterranean.org#RedSea"/>
</River>
```



By defining emptiesInto as a FunctionalProperty we assert that EastChinaSea and RedSea must be the same. But owl:differentFrom indicates that they are different! Thus, there is a contradiction. It indicates that the instance is in error.

owl:AllDifferent

Using the owl:AllDifferent class we can indicate that a collection of instances are different:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
               xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
               xmlns:owl="http://www.w3.org/2002/07/owl#"
               xmlns:geo="http://www.geodesy.org/water/naturally-occurring#"
               xml:base="http://www.geodesy.org/water/naturally-occurring">

    <owl:AllDifferent>
        <owl:distinctMembers rdf:parseType="Collection">
            <geo:Sea rdf:about="http://www.china.org/geography/rivers#EastChinaSea"/>
            <geo:Sea rdf:about="http://www.mediterranean.org#RedSea"/>
            <geo:Sea rdf:about="http://www.africa.org/rivers#ArabianSea"/>
            <geo:Sea rdf:about="http://www.philippines.org#PhilippineSea"/>
        </owl:distinctMembers>
    </owl:AllDifferent>

    ...

</rdf:RDF>
```

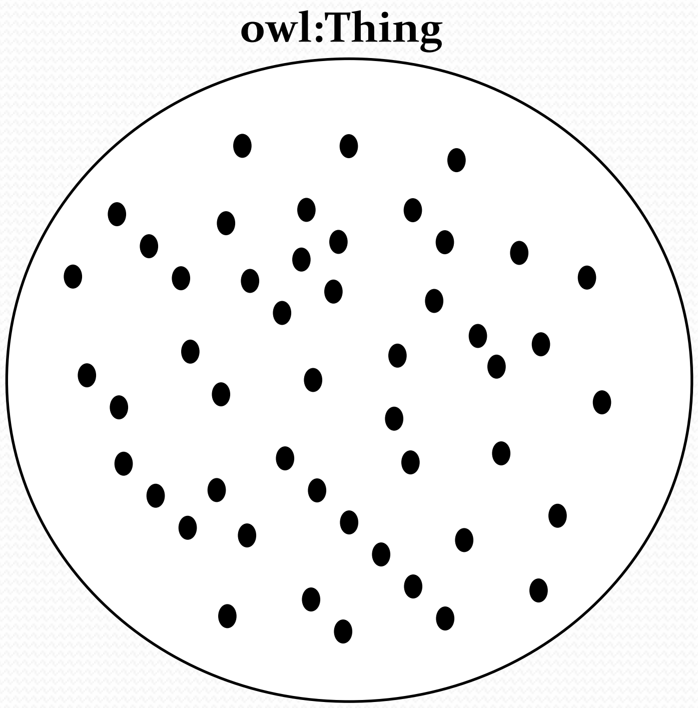
This indicates that the EastChinaSea, RedSea, ArabianSea, and PhilippineSea are all different!

Summary of the different statements you can incorporate into instances

- In the preceding slides we saw the OWL statements that you can put into instance documents:
  - sameIndividualAs
  - differentFrom
- Question: what about AllDifferent? Answer: the owl:AllDifferent class is typically used in an ontology document, not an instance document (however, you can use it in an instance document if you like).

owl:Thing

owl:Thing is a predefined OWL class.
All instances are members of owl:Thing.



Every instance in the universe is a member of owl:Thing!

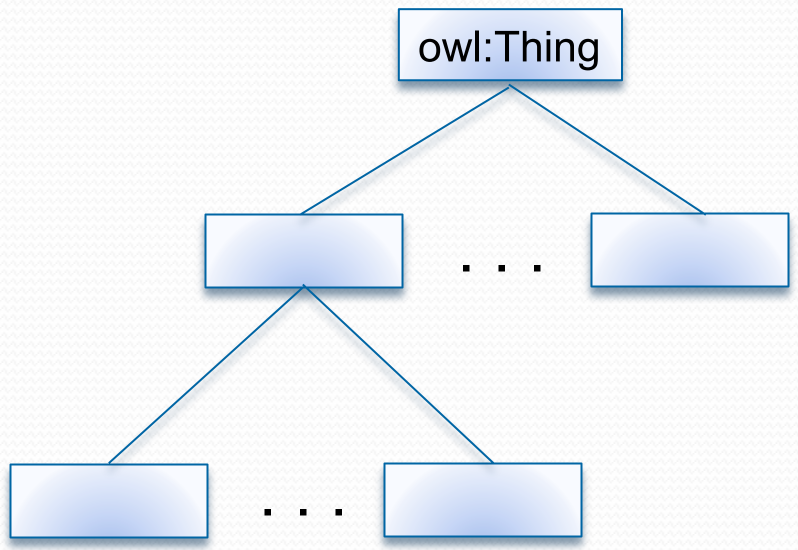
Equivalent!

```
<?xml version="1.0"?>
<River rdf:ID="Yangtze"
          xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns="http://www.geodesy.org/water/naturally-occurring#">
    <length>6300 kilometers</length>
    <emptiesInto rdf:resource="http://www.china.org/geography#EastChinaSea"/>
</River>
```

```
<?xml version="1.0"?>
<owl:Thing rdf:ID="Yangtze"
              xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                     xmlns:owl="http://www.w3.org/2002/07/owl#"
              xmlns="http://www.geodesy.org/water/naturally-occurring#">
    <rdf:type rdf:resource="http://www.geodesy.org/water/naturally-occurring#River"/>
    <length>6300 kilometers</length>
    <emptiesInto rdf:resource="http://www.china.org/geography#EastChinaSea"/>
</owl:Thing>
```

"Yangtze is a Thing. Specifically, it is a River Thing."

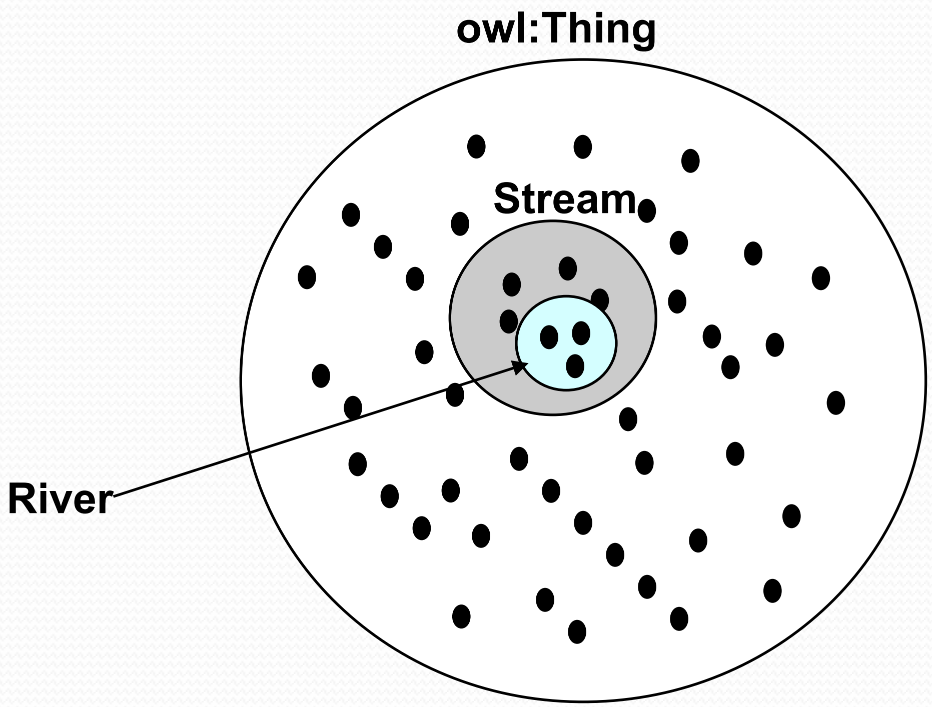
The owl:Thing class is the root of all classes



Equivalent!

```
<owl:Class rdf:ID="River">
        <rdfs:subClassOf rdf:resource="#Stream"/>
</owl:Class>
```

```
<owl:Class rdf:ID="River">
        <rdfs:subClassOf rdf:resource="#Stream"/>
        <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
</owl:Class>
```



Importing other OWL documents

Other OWL documents must be specified in-band

- With RDF Schema you can simply use another Schema. You don't have to "import" the Schema. Tools are expected to locate the Schema "out-of-band".

- With OWL you must explicitly import the OWL documents you will use.

The Ontology Header

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
            xmlns:owl="http://www.w3.org/2002/07/owl#"
            xmlns:geo="http://www.geodesy.org/water/naturally-occurring#"
            xml:base="http://www.geodesy.org/water/naturally-occurring">

   <owl:Ontology rdf:about="http://www.geodesy.org/water/naturally-
occurring">
      <owl:priorVersion rdf:resource="http://www.geodesy.org/water/
naturally-occurring/july-02#"/>
      <owl:versionInfo>naturally-occurring.owl v 2.0</owl:versionInfo>
      <owl:imports rdf:resource="http://www.other.org/other.owl"/>
   </owl:Ontology>

   ...

</rdf:RDF>
```

owl:Ontology Properties

owl:Ontology

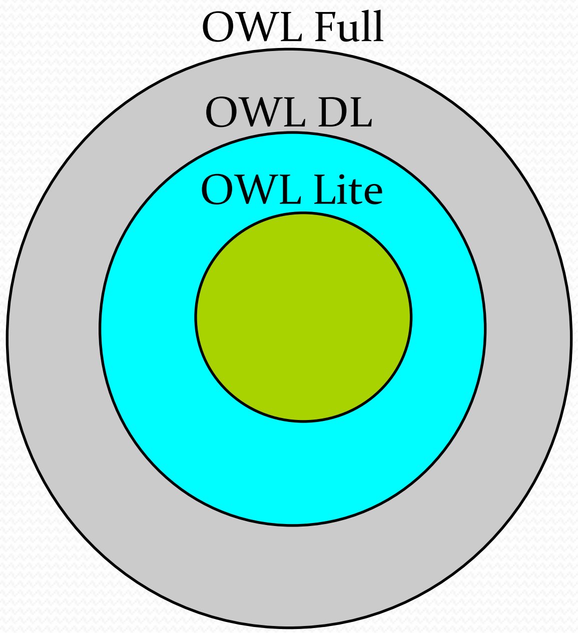
Properties:

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

Note: *Ontology* refers to the OWL document, e.g., naturally-occurring.owl.

OWL Full, OWL DL, and OWL Lite

- Not everyone will need all of the capabilities that OWL provides. Thus, there are three versions of OWL:



Comparison

OWL Full	OWL DL	OWL Lite
Everything that has been shown in this tutorial is available. Further, you can mix RDF Schema definitions with OWL definitions.	You cannot use owl:cardinality with TransitiveProperty. A DL ontology cannot import an OWL Full ontology. You cannot use a class as a member of another class, i.e., you cannot have metaclasses. FunctionalProperty and InverseFunctionalProperty cannot be used with datatypes (they can only be used with ObjectProperty).	All the DL restrictions plus: You cannot use owl:minCardinality or owl:maxCardinality. The only allowed values for owl:cardinality is 0 and 1. Cannot use owl:hasValue. Cannot use owl:disjointWith. Cannot use owl:oneOf. Cannot use owl:complementOf. Cannot use owl:unionOf.

Advantages/Disadvantages

- Full:
  - The advantage of the Full version of OWL is that you get the full power of the OWL language.
  - The disadvantage of the Full version of OWL is that it is difficult to build a Full tool. Also, the user of a Full-compliant tool may not get a quick and complete answer.
- DL/Lite:
  - The advantage of the DL or Lite version of OWL is that tools can be built more quickly and easily, and users can expect responses from such tools to come quicker and be more complete.
  - The disadvantage of the DL or Lite version of OWL is that you don't have access to the full power of the language.

OWL2 EL

Suitable for applications employing ontologies that define very large numbers of classes and/or properties

For example, OWL2 EL does **NOT** support

- universal quantification to a class expression (AllValuesFrom)
- cardinality restrictions (MaxCardinality, MinCardinality, ExactCardinality,
- MaxCardinality, DataMinCardinality, and DataExactCardinality)
- Union (UnionOf)
- class negation (ComplementOf)
- enumerations involving more than one individual (OneOf)
- disjoint properties (DisjointProperties)
- irreflexive object properties (IrreflexiveObjectProperty)
- inverse object properties (InverseProperties)
- functional and inverse-functional object properties (FunctionalProperty and
- InverseFunctionalProperty)
- symmetric object properties (SymmetricProperty)
- asymmetric object properties (AsymmetricProperty)

OWL2 QL

is designed so that data (assertions) that is stored in a standard relational database system can be queried through an ontology via a simple rewriting mechanism

For example, OWL2 QL does **NOT** support

- existential quantification to a class expression or a data range (SomeValuesFrom)
- existential quantification to an individual or a literal (HasValue)
- enumeration of individuals and literals (ObjectOneOf, DataOneOf)
- universal quantification to a class expression or a data range (AllValuesFrom)
- cardinality restrictions (MaxCardinality, MinCardinality, ExactCardinality,
- MaxCardinality, MinCardinality, ExactCardinality)
- union (ObjectUnionOf)
- property inclusions (SubPropertyOf)
- functional and inverse-functional properties (FunctionalProperty, InverseFunctionalProperty)
- transitive properties (TransitiveProperty)

OWL2 RL

aimed at applications that require scalable reasoning without sacrificing too much expressive power

For example, OWL2 QL does **NOT** support

- disjoint unions of classes (DisjointUnion)
- reflexive object property axioms (ReflexiveProperty)

OWL2 Features

- Some useful features were introduced in OWL2, for example:
  - HasKey
    - To uniquely identify individuals of a given class by values. e.g. each person can be identified by property hasDNAsequence.
  - PropertyChain
    - To define a property as a composition of other properties. e.g. property hasUncle can be defined as chain of hasParent and hasBrother
  - DisjointUnion
    - To define a class as the union of other classes, all of which are pairwise disjoint.
  - …

https://www.w3.org/TR/owl2-new-features/

Related Documents

- The OWL Guide provides a description of OWL, with many examples:
  - http://www.w3.org/TR/owl-guide/
- Here is the URL to the OWL Reference document:
  - http://www.w3.org/TR/owl-ref/
- OWL New Features
  - https://www.w3.org/TR/owl2-new-features/
- OWL2 Specification
  - https://www.w3.org/TR/owl2-syntax/
- For all other OWL documents, and information on the Semantic Web see:
  - http://www.w3.org/2001/sw/