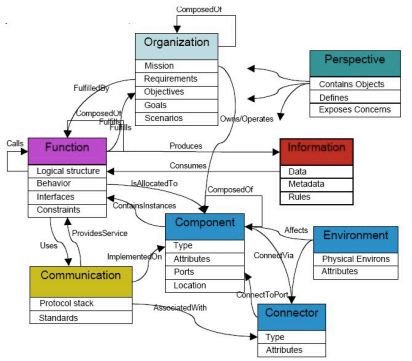
Ontology Engineering



Acknowledgements

Natalya F. Noy Stanford University (noy@smi.stanford.edu)

A large part of this tutorial is based on

"Ontology Development 101: A Guide to Creating Your First Ontology" by Natalya F. Noy and Deborah L. McGuinness

http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html

Lecture Notes edited by

Monika Solanki (monika.solanki@gmail.com)

What is “Ontology Engineering”?

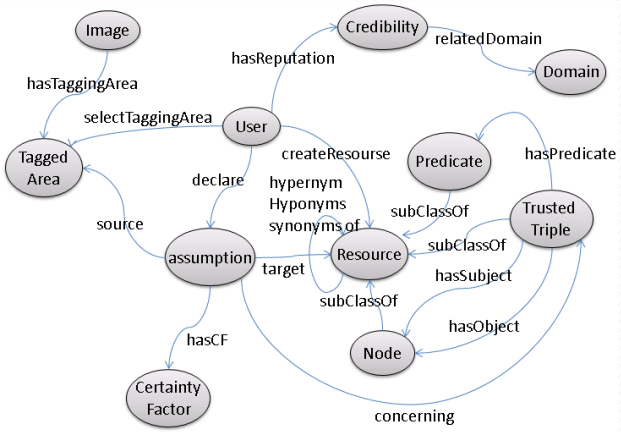
- Ontology Engineering: Defining terms in the domain and relations among them
  - Defining concepts in the domain (classes)
  - Arranging the concepts in a hierarchy (subclass-superclass hierarchy)
  - Defining which attributes and properties classes can have and constraints on their values
  - Defining individuals and filling in slot values

Why Develop an Ontology?

- To share common understanding of the structure of information
  - among people
  - among software agents
- To enable reuse of domain knowledge
  - to avoid “re-inventing the wheel”
  - to introduce standards to allow interoperability
- To make domain assumptions explicit
  - easier to change domain assumptions (consider easier to understand and update legacy data)
- To separate domain knowledge from the operational knowledge
  - re-use domain and operational knowledge separately (e.g., configuration based on constraints)

An Ontology is Often Just the Beginning

Ontology



declare structure

Knowledge base

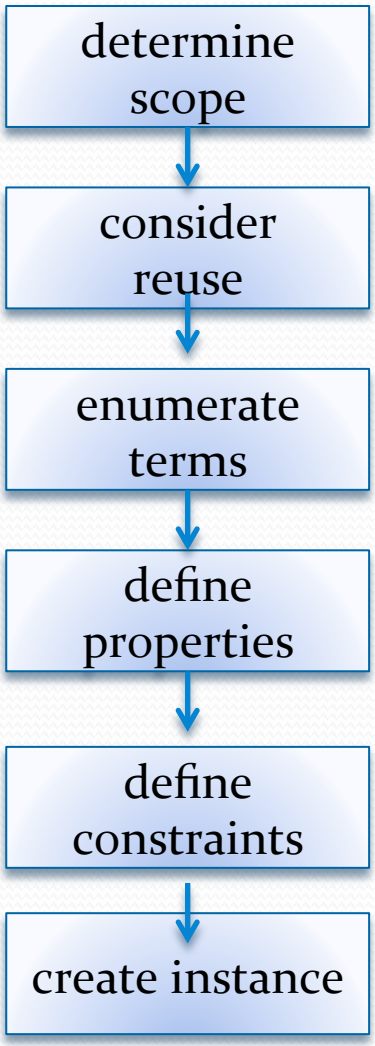
Ontology-based Database

Provide domain description

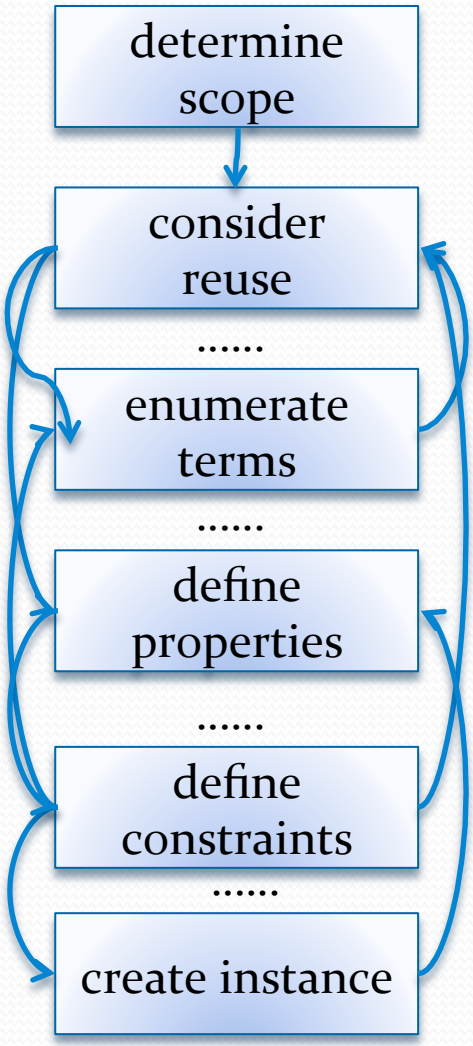
Software agent	Problem solving method	Domain independent applications

Ontology-Development Process

In this tutorial:



Reality – an iterative process



Ontology Engineering vs Object-Oriented Modeling

- An Ontology
  - Reflect the structure of the **world**
  - Often about the structure of **concepts**
  - Actual physical representation is **not** an issues
- An OO class structure
  - Reflect the structure of the data and code
  - Is usually about behavior (methods)
  - Describe the physical representation of data (e.g. long int, char, etc)

Preliminaries - Tools

- Protégé (version 3.4.8 for RDF/S version 5 for OWL)
  - A graphical ontology-development tool
  - Supports a rich knowledge model
  - Open source and freely available
    - http://protege.stanford.edu/download/protege/3.4/installanywhere/Web_Installers/
- Complete example can be downloaded from Blackboard

Determine Domain and Scope

- What is the domain that the ontology will cover?
  - e.g. Education, Food, Telecommunication, Aircraft, Business Process, Festival and Events etc.
- For what we are going to use the ontology?
  - e.g. As a data model for a system
- For what types of questions the information in the ontology should provide answers (competency questions)?
- Answers to these questions may change during the lifecycle

Determine Domain and Scope

- What is the domain that the ontology will cover?
  - e.g. Education, Food, Telecommunication, Aircraft, Business Process, Festival and Events etc.
- For what we are going to use the ontology?
  - e.g. As a data model for a system
- For what types of questions the information in the ontology should provide answers (competency questions)?
- Answers to these questions may change during the lifecycle

Competency Questions

*Something about **wines and winery** …*

- Which wine characteristics should I consider when choosing a wine?
- Is Bordeaux a red or white wine?
- Does Cabernet Sauvignon go well with seafood?
- What is the best choice of wine for grilled meat?
- Which characteristics of a wine affect its appropriateness for a dish?
- Does a flavor or body of a specific wine change with vintage year?
- What were good vintages for Napa Zinfandel?

Competency Questions

*Something about **wines and winery** …*

- Which wine characteristics should I consider when choosing a wine?
- Is Bordeaux a red or white wine?
- Does Cabernet Sauvignon go well with seafood?
- What is the best choice of wine for grilled meat?
- Which characteristics of a wine affect its appropriateness for a dish?
- Does a flavor or body of a specific wine change with vintage year?
- What were good vintages for Napa Zinfandel?

Consider reuse

- Why reuse other ontologies?
  - to save the effort
  - to interact with the tools that use other ontologies
  - to use ontologies that have been validated through use in applications

What to Reuse?

- Ontology libraries
  - http://protegewiki.stanford.edu/wiki/Protege_Ontology_Library
  - http://owl.cs.manchester.ac.uk/tools/repositories/
  - http://www.cs.ox.ac.uk/isg/ontologies/lib/
  - https://onki.fi/
  - Google and Swoogle…
- General ontologies
  - DMOZ (www.dmoz.org)
  - WordNet (https://wordnet.princeton.edu)
- Domain-specific ontologies
  - UMLS Semantic Net
  - General Formal Ontology (GFO)
  - GO (Gene Ontology) (www.geneontology.org)
  - CIDO-CRM (Cultural Heritage) (http://www.cidoc-crm.org/)

Enumerate Important Terms

- What are the terms we need to talk about?
- What are the properties of these terms?
- What do we want to say about the terms?

Enumerating Terms - The Wine Ontology

- wine, grape, winery, location,
- wine color, wine body, wine flavor, sugar content
- white wine, red wine, Bordeaux wine
- food, seafood, fish, meat, vegetables, cheese

Define Classes and the Class Hierarchy

- A class is a concept in the domain
  - a class of wines
  - a class of wineries
  - a class of red wines
- A class is a collection of elements with similar properties
- Instances of classes
  - a glass of California wine you'll have for lunch

ABox vs TBox

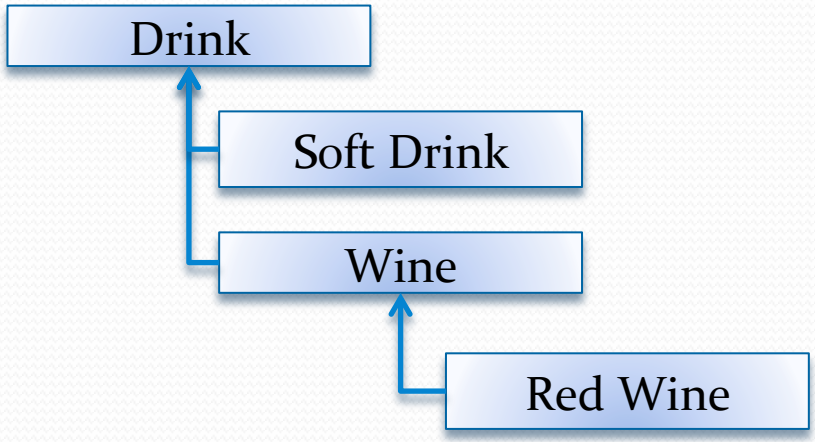
- ABox and TBox are used to describe two different types of statements in ontologies
  - TBox (Terminological component) : a set of classes and properties for concepts. TBox statements are often associated with object-oriented classes. e.g.
    - class `Man` and `Woman` are disjointed
    - `Student` and `Lecturer` are two types of Person
  - Abox (Assertion component): individuals (or instances) belonging to those concepts. ABox statements are associated with instances of those classes. e.g.
    - Bob is a `Student`
    - `a` is an instance of class `A`

Class? instance?

Location	*Class*
France	*Individual (instance of Country)*
US_Region	*Class*
California	*Individual (instance of US_Region)*

Class Hierarchy

- Classes usually constitute a taxonomic hierarchy (a subclass-superclass hierarchy)
- A class hierarchy is usually an IS-A hierarchy:
  - *an instance of a subclass is an instance of a superclass*
- If you think of a class as a set of elements, a subclass is a subset



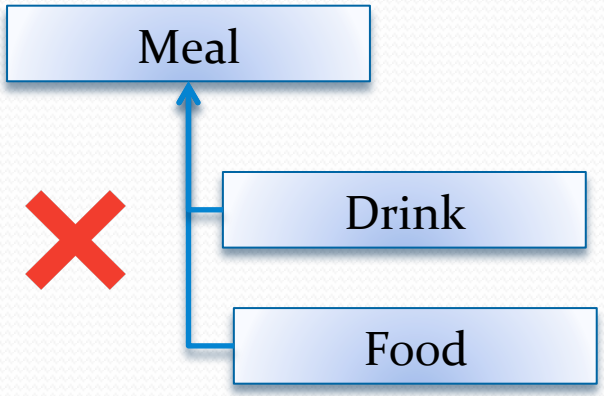
Class Inheritance - Example

- Apple is a subclass of Fruit
  - Every apple is a fruit
- Red wines is a subclass of Wine
  - Every red wine is a wine
- Chianti wine is a subclass of Red wine
  - Every Chianti wine is a red wine
- But this bottle of wine I'm drinking today is an instance of Chianti wine.

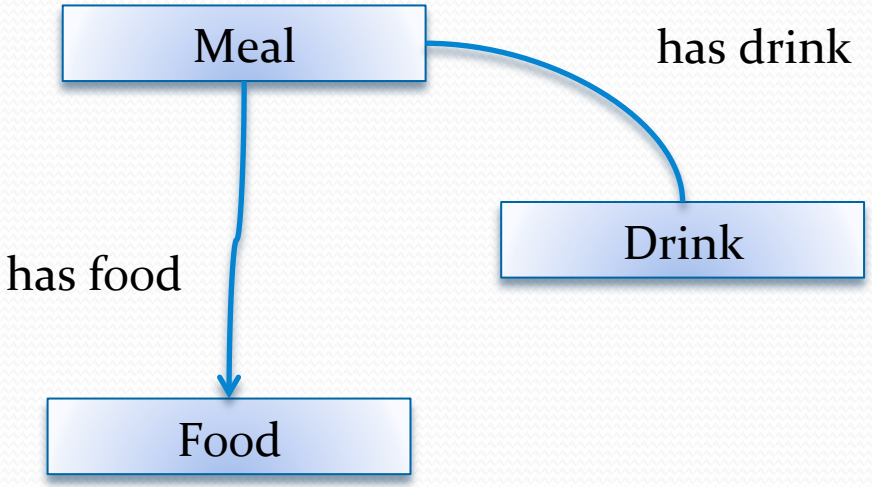
important to know the differences between classes and instances!!

Class Inheritance - Pitfall!!

- Satisfy **IS-A** relationship



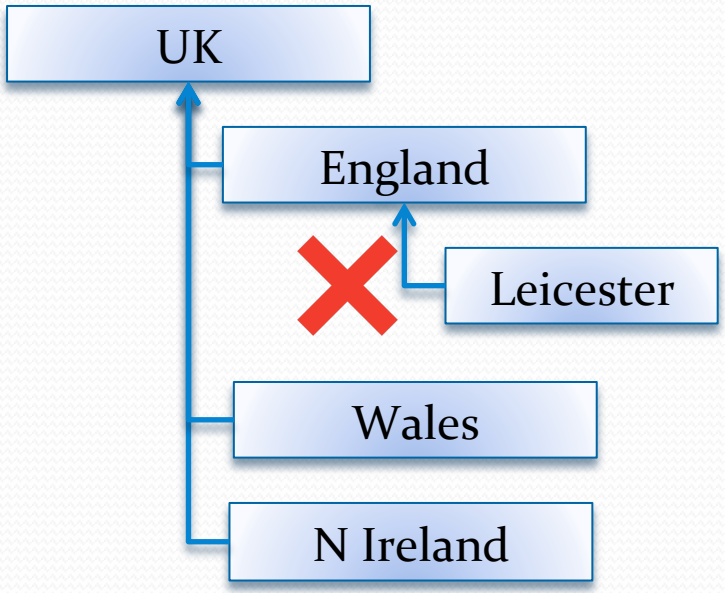
A drink is NOT a meal
A drink **is part of** a meal

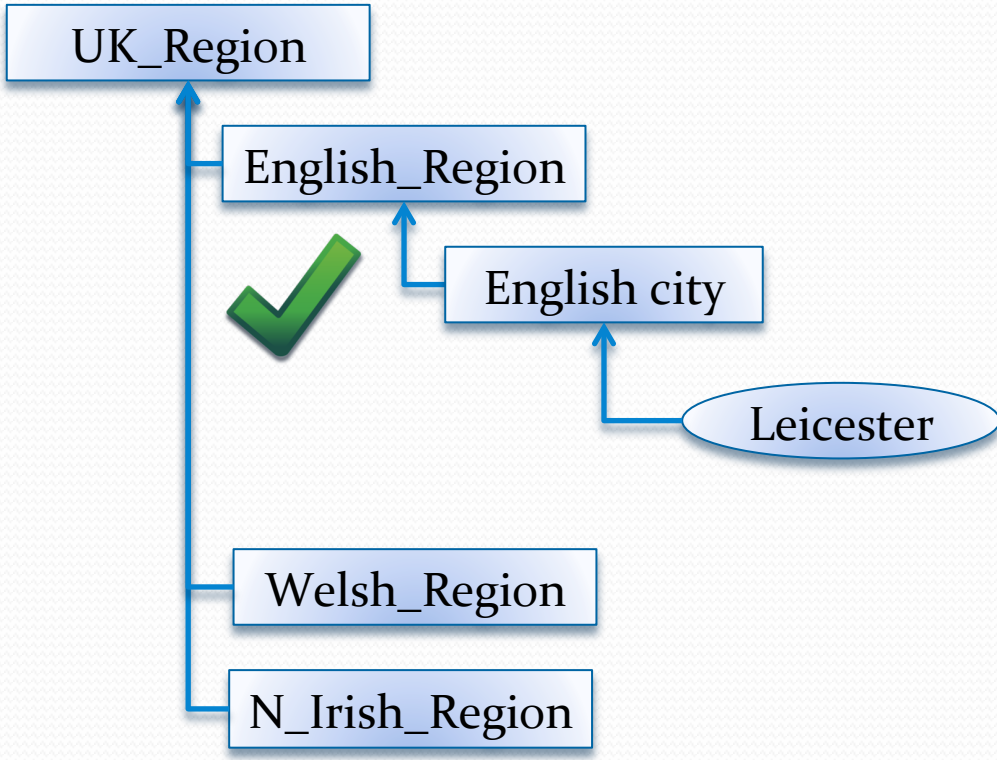


They should be connected through properties such as “has_food”, “has_drink”.

Class Inheritance - Pitfall!!

- Distinguish **Class** and **instance**

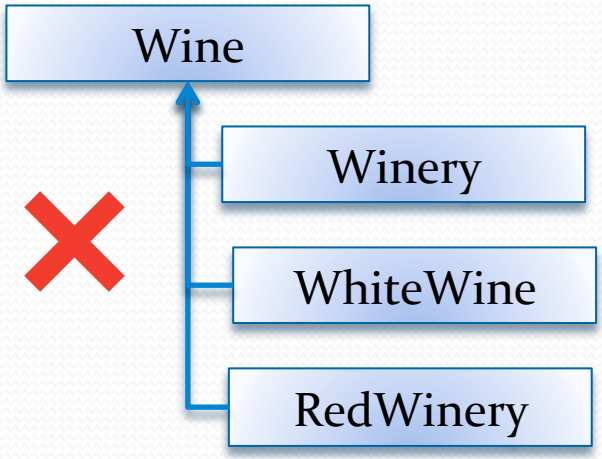


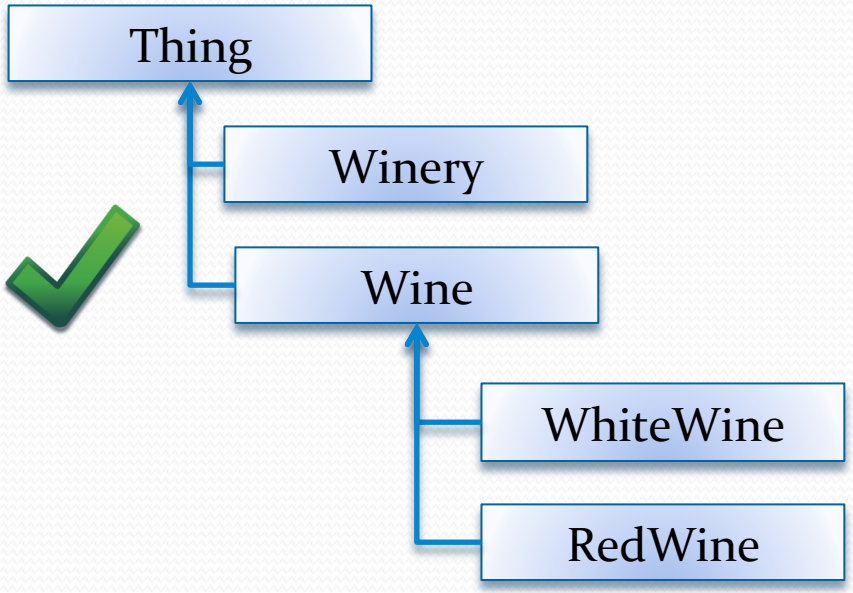


* Bare in mind **Leicester** is an instance NOT a class

Class Inheritance - Pitfall!!

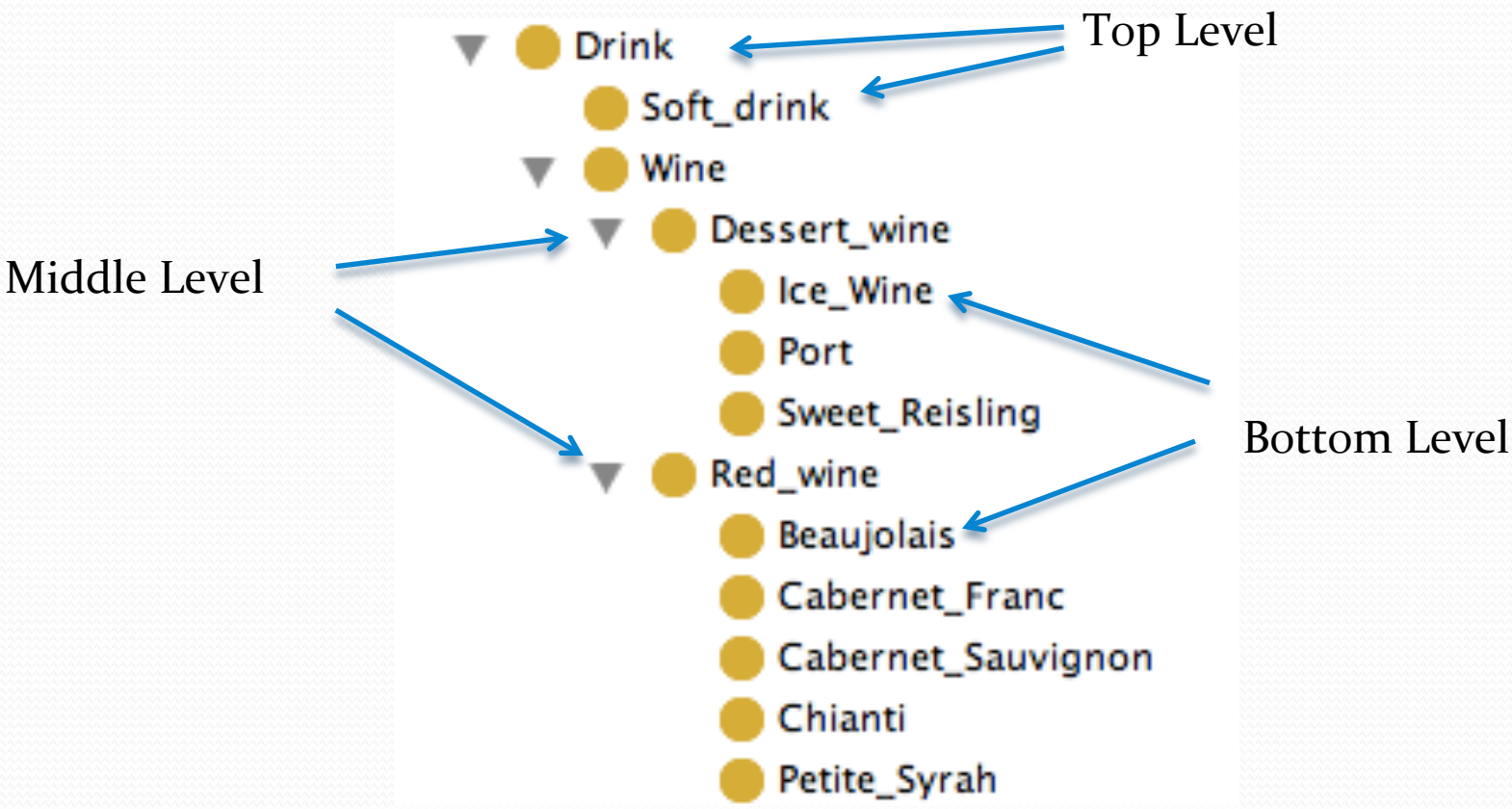
- Domain name is not always the super class of all concepts in the ontology





*Winery is NOT a wine so the hierarchy on the left is incorrect

Levels in the Hierarchy



Modes of Development

- **top-down** – define the most general concepts first and then specialize them
- **bottom-up** – define the most specific concepts and then organize them in more general classes
- **combination** – define the more salient concepts first and then generalize and specialize them

Documentation

- Classes (and slots) usually have documentation
  - Describing the class in natural language
  - Listing domain assumptions relevant to the class definition
  - Listing synonyms
- Documenting classes and slots is as important as documenting computer code!
- How to write documentation?
  - e.g. rdfs:comment, rdfs:label in RDF/S

```
<rdfs:Class rdf:about="#Wine">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >A wine class represents all possible wines</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#Drink"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Wine</rdfs:label>
 </rdfs:Class>
```

Define Properties of Classes – Slots

- Slots in a class definition describe attributes of instances of the class and relations to other instances
- Slots ≈ properties ≈ attributes and relations

  Each wine will have name, color, sugar content, producer, etc.

Properties (Slots)

- Types of properties
  - “intrinsic” properties: *flavor* and *color* of wine
  - “extrinsic” properties: *name* and *price* of wine
  - parts: *ingredients* in a dish
  - relations to other objects: *producer* of wine (winery)
- Simple and complex properties
  - simple properties (attributes): contain primitive values (strings, numbers)
  - complex properties: contain (or point to) other objects (e.g., a winery instance)

Properties and Class Inheritance

- A subclass inherits all the slots (attributes and relations) from the superclass
  - If a wine has a name and flavor, a red wine also has a name and flavor
- If a class has multiple superclasses, it inherits slots from all of them
  - *Port is both a dessert wine and a red wine. It inherits "sugar content: high" from the former and "color:red" from the latter*
  - *A university employee doing a part-time degree course inherits all attributes (student_ID, staff_ID etc) and relations (teach, study) defined for Employee and Student*

Property Constraints

- Property constraints (facets) describe or limit the set of possible values for a slot
  - The name of a wine is a string
  - The wine producer is an instance of Winery
  - A winery has exactly one location

Common Constraints

- Property cardinality – the number of values a slot has
- Property value type – the type of values a slot has
- Minimum and maximum value – a range of values for a numeric property
- Default value – the value a property has unless explicitly specified otherwise

Common Constraints: *Cardinality*

- Cardinality
  - Cardinality N means that the slot must have N values.

    e.g. Cardinality of "biologicalMother" is 1; Cardinality of property "hasFinger" is 10;
- Minimum cardinality
  - Minimum cardinality 1 means that the slot must have a value (required)

    e.g. "biologicalMother"; "dateOfBirth"; winery has exactly one location
  - Minimum cardinality 0 means that the slot value is optional

    e.g. "hasHomeTelephone",
- Maximum cardinality
  - Maximum cardinality 1 means that the slot can have at most one value (single-valued slot)

    e.g "hasNInumber"
  - Maximum cardinality greater than 1 means that the slot can have more than one value (multiple-valued slot)

    e.g. For a mother, the property "hasChild"

Common Constraints: *Value Type*

- String: a string of characters (“Chateau Latour”)
- Number: an integer or a float (5,11.5)
- Boolean: True/False
- Enumerated type: a list of allowed values (
  - *e.g. high, medium, low);*

  RGB (red, green, blue)
- Complex type: an instance of another class
  - Specify the class to which the instances belong

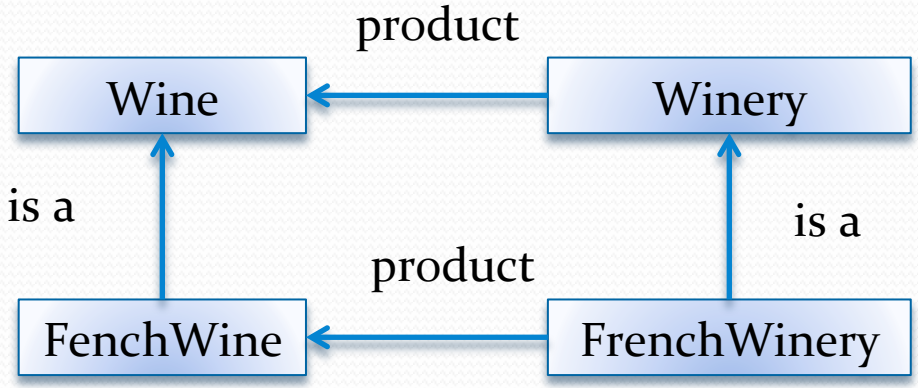
  The Wine class is the value type for the slot “produces” at the Winery class

Domain and Range of Property

- Domain:
  - A domain of a property limits the individuals to which the property can be applied.
  - More precisely: class (or classes) instances of which can have the property
- Range:
  - The range of a property limits the individuals that the property may have as its value
  - the class (or classes) to which property values belong

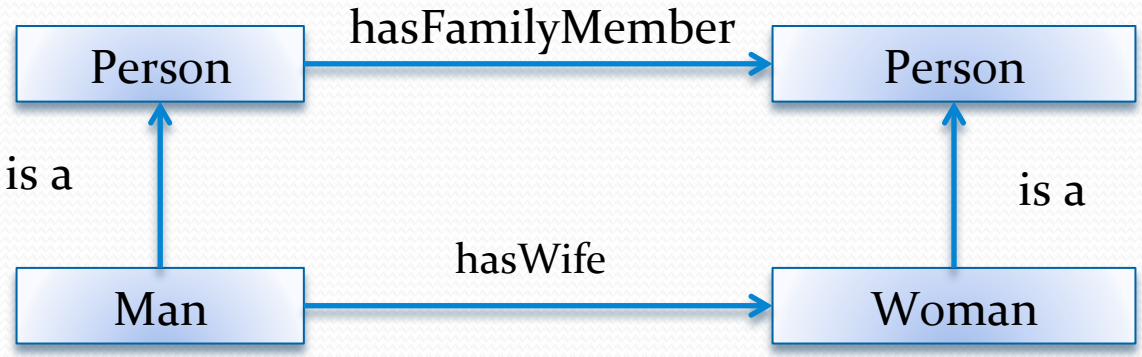
Domain/Range and Class Inheritance

- A subclass inherits all the slots from the superclass
- A subclass can override the facets to "narrow" the list of allowed values
  - Make the cardinality range smaller
  - Replace a class in the range with a subclass



Domain/Range and Property Inheritance (2)

- A sub property will inherit all domains and ranges from its super properties
- It is possible to specify more specific domain and range classes



Create Instances

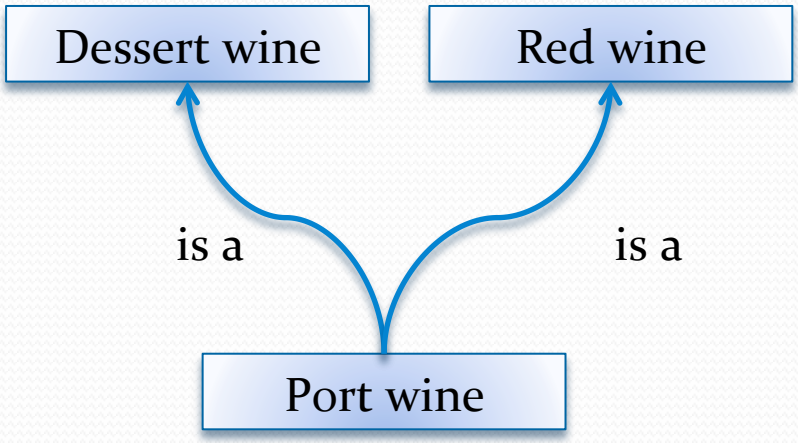
- Create an instance of a class
  - The class becomes a direct type of the instance
  - Any superclass of the direct type is a type of the instance
- Assign property values for the instance frame
  - Property values should conform to the facet constraints
  - *A reasoner* can be used to check that

Defining Classes and a Class Hierarchy

- The things to remember:
  - There is no single correct class hierarchy
  - But there are some guidelines
- The question to ask:
  - “Is each instance of the subclass an instance of its superclass?”

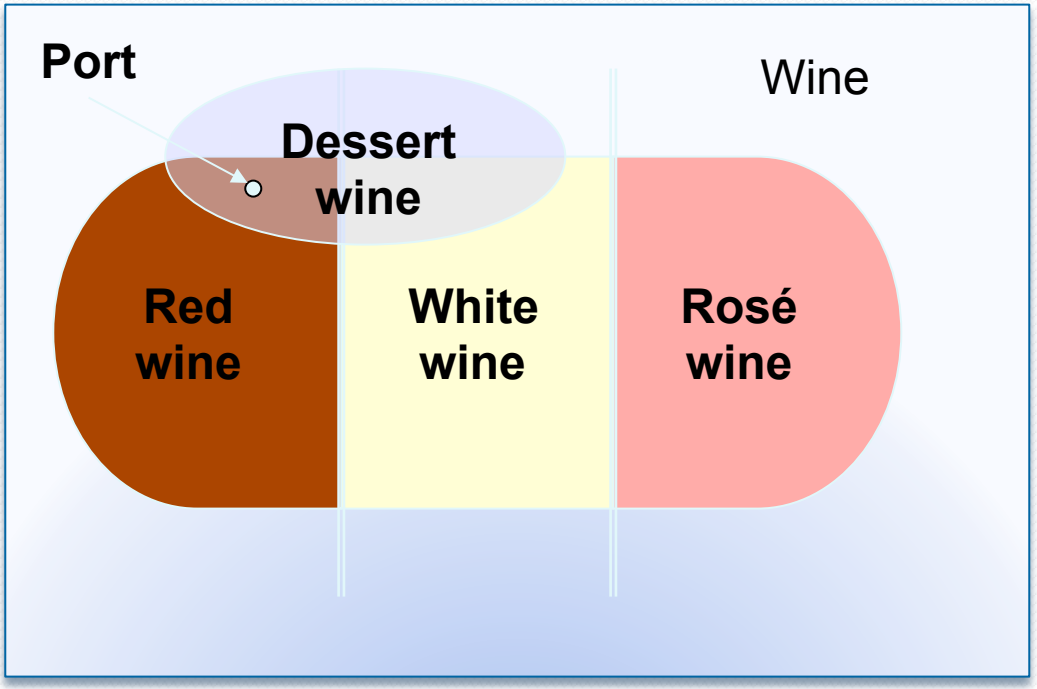
Multiple Inheritance

- A class can have more than one superclass
- A subclass inherits properties and restrictions from all the parents
- Different systems resolve conflicts differently



Disjoint Classes

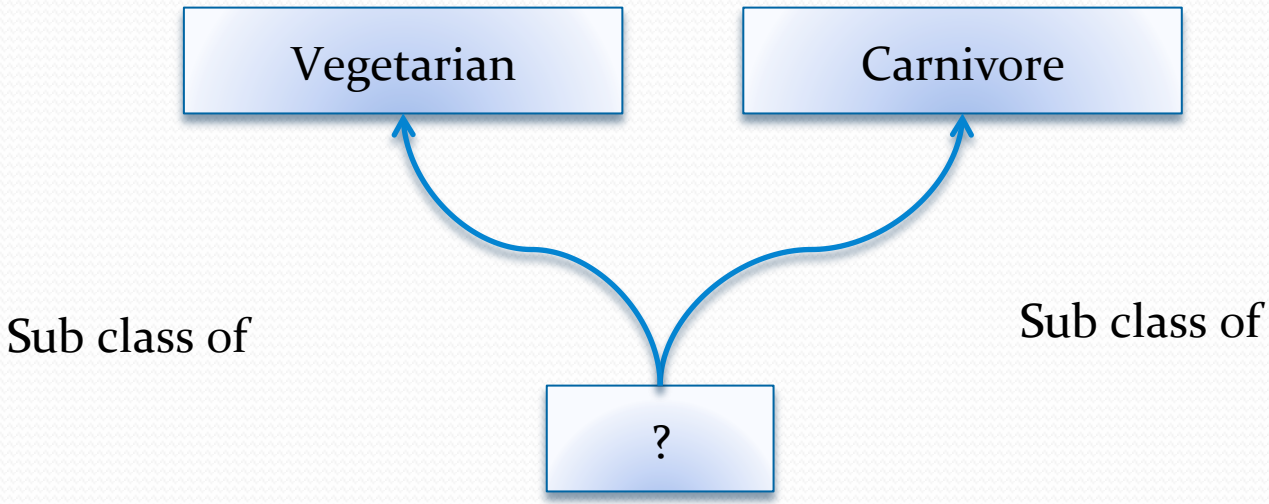
- Classes are disjoint if they cannot have common instances



Red wine, White wine, Rosé wine are disjoint

Dessert wine and Red wine are not disjoint

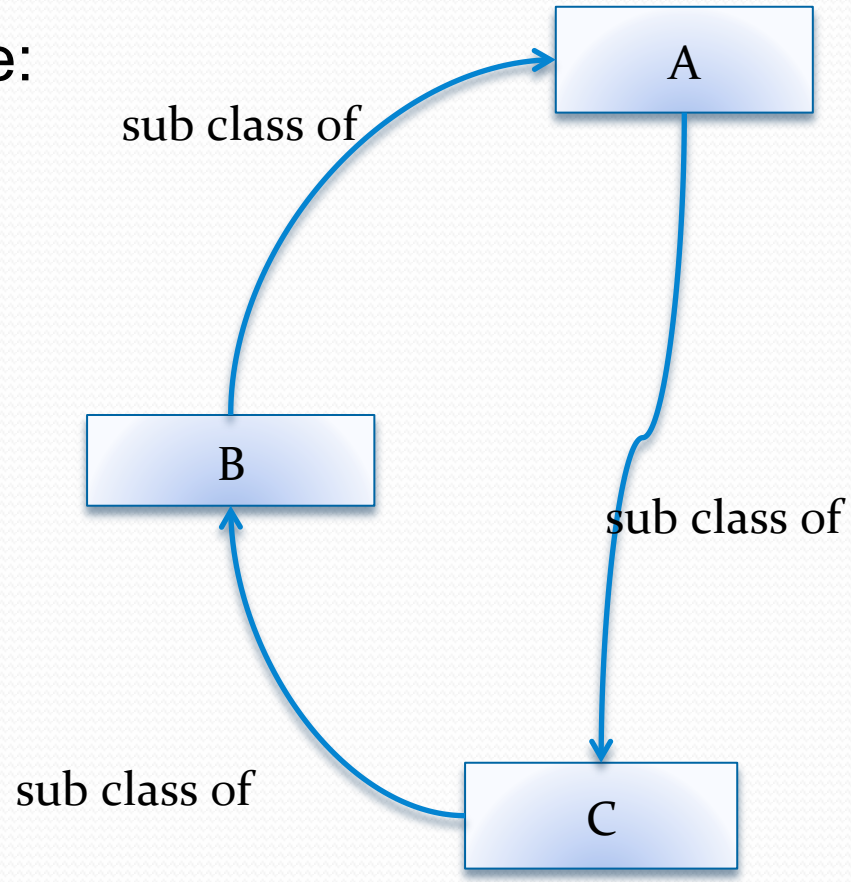
Disjoint Classes



Disjoint classes cannot have any common subclasses either

Avoiding Class Cycles

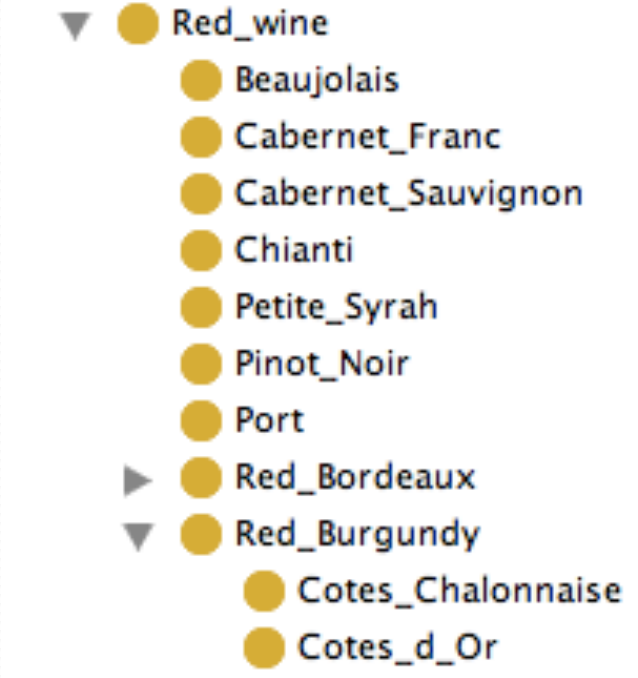
- Danger of multiple inheritance: cycles in the class hierarchy
- Classes A, B, and C have equivalent sets of instances
  - By many definitions, A, B, and C are thus equivalent



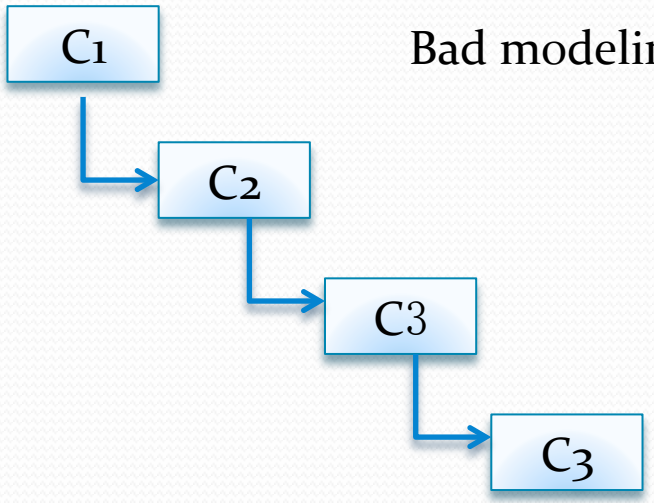
Siblings in a Class Hierarchy

- All the siblings in the class hierarchy must be at the same level of generality
- Compare to section and subsections in a book

Siblings in a Class Hierarchy



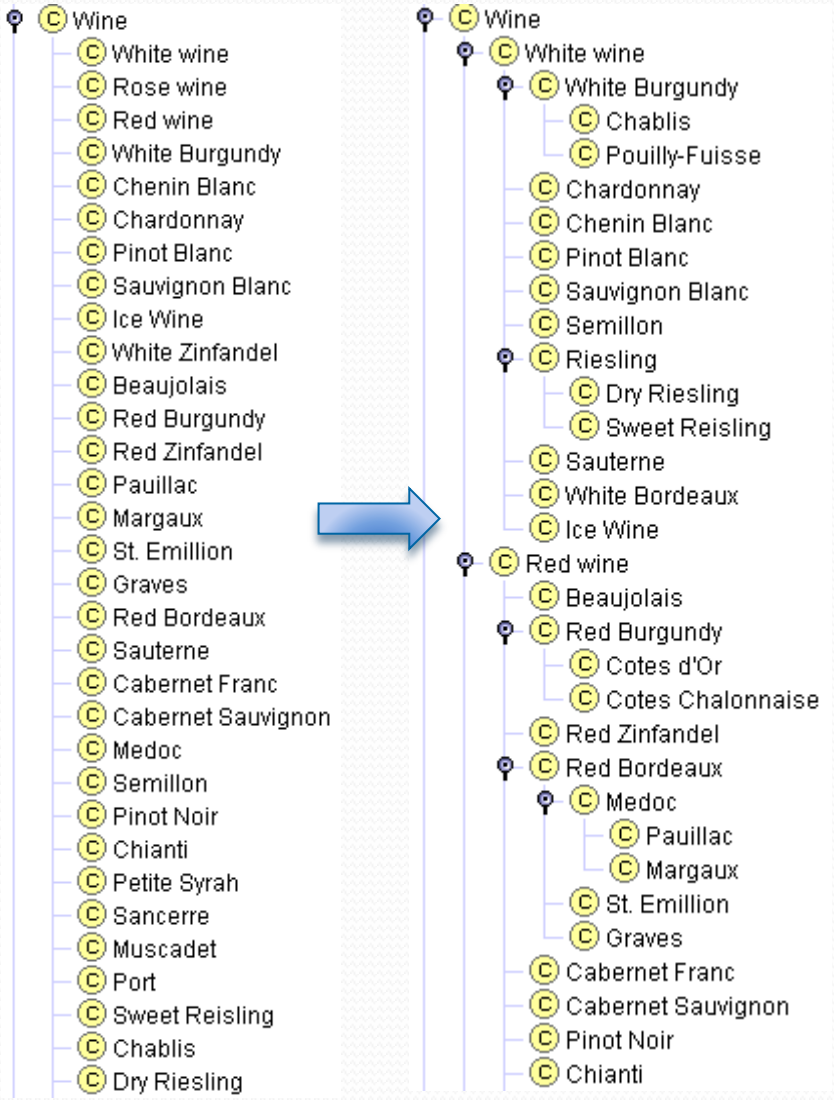
- If a class has only one child, there may be a modeling problem
- If the only Red Burgundy we have is Côtes d'Or, why introduce the subhierarchy?
- Compare to bullets in a bulleted list



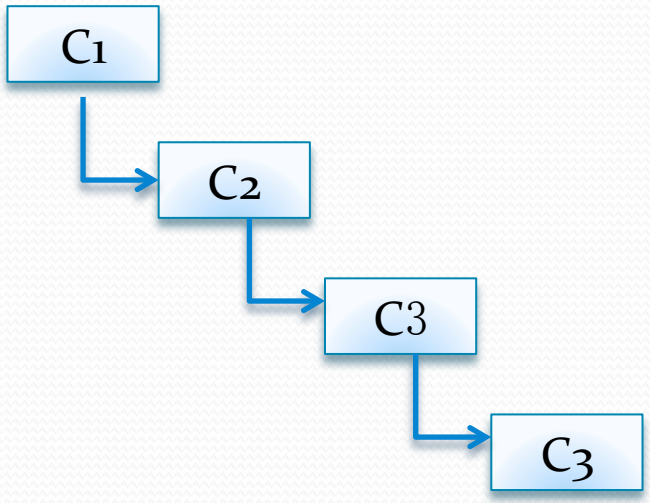
Bad modeling practice

Try not to introduce unnecessary subcategory

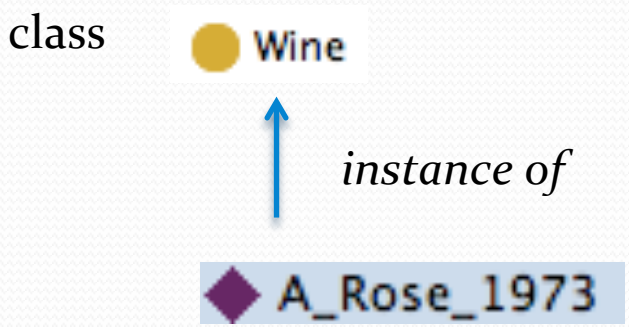
Siblings in a Class Hierarchy



- If a class has only one child, there may be a modeling problem
- If the only Red Burgundy we have is Côtes d'Or, why introduce the subhierarchy?
- Compare to bullets in a bulleted list



Single and Plural Class Names



- A “wine” is not a kind-of “wines”
- A wine is an instance of the class Wines
- Class names should be either
  - all singular
  - all plural

Classes and Their Names

- Classes represent concepts in the domain, not their names
- The class name can change, but it will still refer to the same concept
- Synonym names for the same concept are not different classes
  - Many systems allow listing synonyms as part of the class definition

Back to the Properties(Slots): Domain and Range

- When defining a domain or range for a property, find the most general class or classes
- Consider the flavor slot
  - Domain: Red wine, White wine, Rosé wine
  - Domain: Wine
- Consider the produces slot for a Winery:
  - Range: Red wine, White wine, Rosé wine
  - Range: Wine

Defining Domain and Range

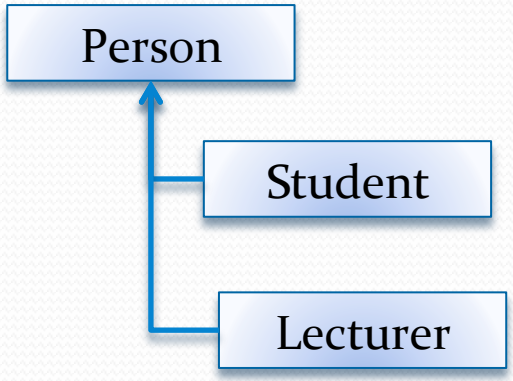
- A class and a superclass
  - – replace with the superclass
- All subclasses of a class
  - – replace with the superclass
- Most subclasses of a class
  - – consider replacing with the superclass

Back to the Properties(Slots): Domain and Range

Example

(1)

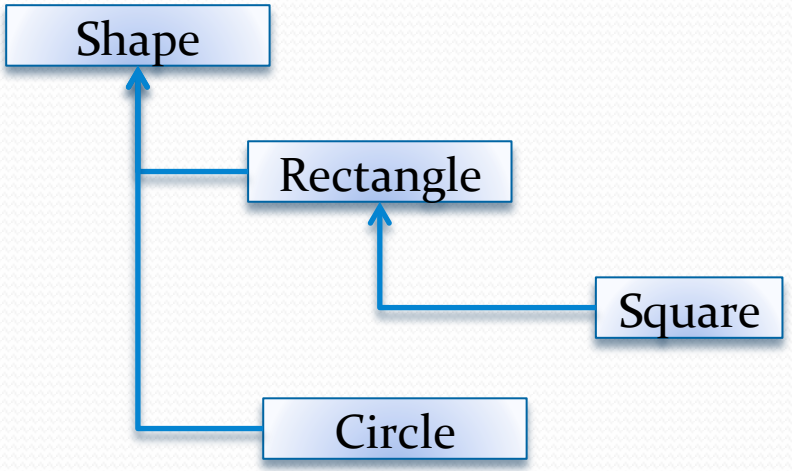
Property: *isFriendOf*
- Domain: Person
- Range: Person



(2)

Property: *hasDiameter*
- Domain: Circle
- Range: float (number)

Property: *hasX, hasY*
- Domain: Rectangle
- Range: float (number)



Limiting the Scope

- An ontology should not contain all the possible information about the domain
  - No need to specialize or generalize more than the application requires
- No need to include all possible properties of a class
  - Only the most salient properties
  - Only the properties that the applications require

Limiting the Scope (2)

- Ontology of wine, food, and their pairings probably will not include
  - Bottle size
  - Label color
  - My favorite food and wine
- An ontology of biological experiments will contain
  - Biological organism
  - Experimenter
- Is the class Experimenter a subclass of Biological organism?

Essential Reading

- Chapter 7 of the recommended textbook1
- Chapter 8 of the recommended textbook2