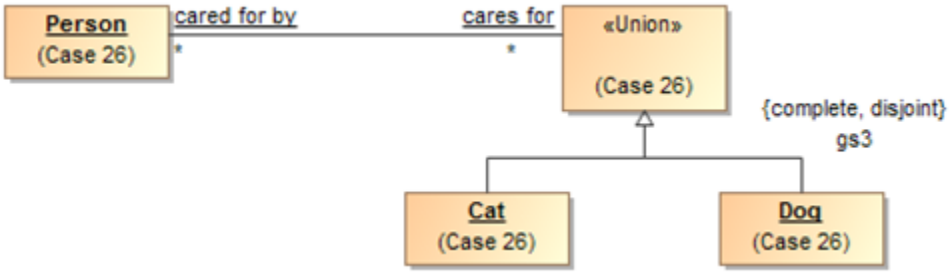
Anonymous union classes

An anonymous union is an unnamed class used to represent a set of classes that can be used as a type of a property. An anonymous union class always implies a complete subclass generalization. (See Complete Subclasses.)

The following diagram states that an instance of a Person may have a value of type Cat or Dog for the *cares for* property. The diagram also states that an instance of a Cat or a Dog may have a value of type Person for the *cared for by* property.

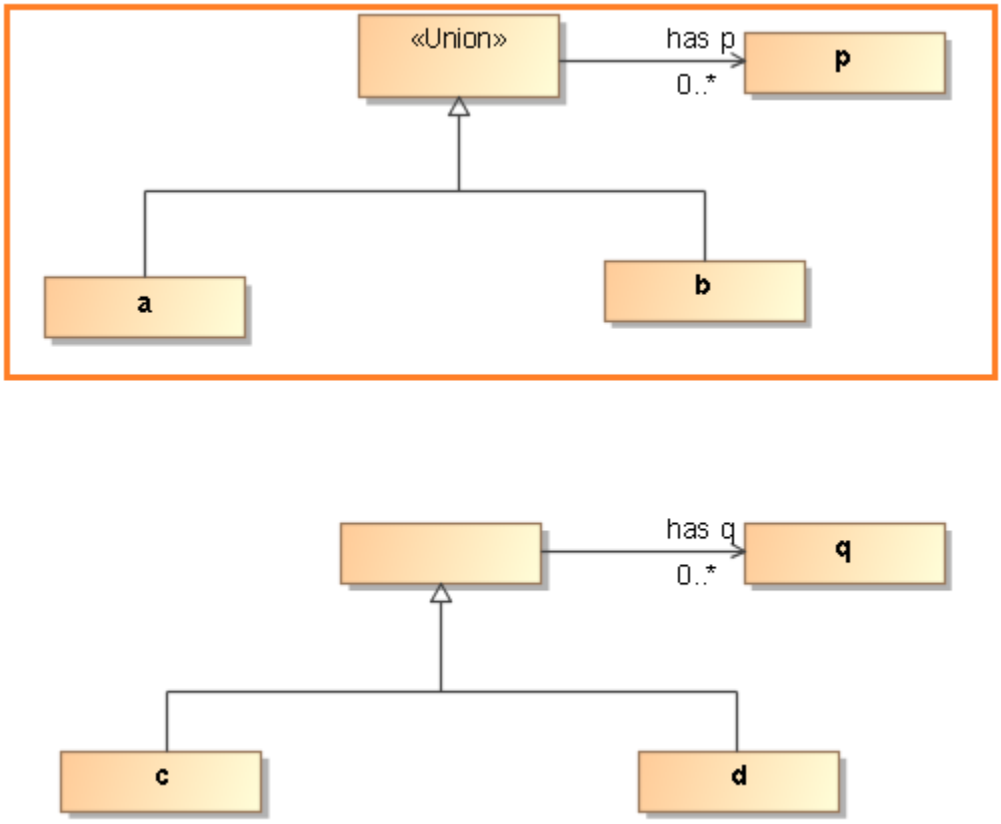


An anonymous union class.

In an ontology, if anonymous union, with same classes within the union, is used in multiple places, the Concept Modeler can distinguish it when importing the ontology. In other words, if the anonymous union has the same union members, the Concept Modeler will identify it as the same anonymous union.

AVAILABLE FROM 19.0 SP2

An anonymous Class with Subclasses but without an explicit «Union» is now treated the same as a union of its Subclasses. The following image show what that means.



The boxed diagram represents the usual relationship between an anonymous Union class. Notice the stereotype, *«Union»*. On the other hand, the diagram not boxed represents a similar structure but there's no anonymous Union class with the stereotype, *«Union»*. The modeling tool is programmed to consider the non-stereotyped class with the subclasses as a Union. Let's see that in the exported OWL of this model.

Reading the following code piece, notice the ObjectUnionOf between A and B, and the ObjectIUnionOf between C and D.

```
# Object Property: :hasP (has p)
ObjectPropertyDomain(:hasP ObjectUnionOf(:A :B))
ObjectPropertyRange(:hasP :P)
# Object Property: :hasQ (has q)
ObjectPropertyDomain(:hasQ ObjectUnionOf(:C :D))
ObjectPropertyRange(:hasQ :Q)
```

The full ontology is displayed below:

```
Ontology(<http://example.com/ontology/uniondemo>

Declaration(Class(:A))
Declaration(Class(:B))
Declaration(Class(:C))
Declaration(Class(:D))
Declaration(Class(:P))
Declaration(Class(:Q))
Declaration(ObjectProperty(:hasP))
Declaration(ObjectProperty(:hasQ))

#############################
#   Object Properties
#############################

# Object Property: :hasP (has p)

AnnotationAssertion(rdfs:label :hasP "has p"^^xsd:string)
ObjectPropertyDomain(:hasP ObjectUnionOf(:A :B))
ObjectPropertyRange(:hasP :P)

# Object Property: :hasQ (has q)

AnnotationAssertion(rdfs:label :hasQ "has q"^^xsd:string)
ObjectPropertyDomain(:hasQ ObjectUnionOf(:C :D))
ObjectPropertyRange(:hasQ :Q)



#############################
#   Classes
#############################

# Class: :A (a)

AnnotationAssertion(rdfs:label :A "a"^^xsd:string)

# Class: :B (b)

AnnotationAssertion(rdfs:label :B "b"^^xsd:string)

# Class: :C (c)

AnnotationAssertion(rdfs:label :C "c"^^xsd:string)

# Class: :D (d)

AnnotationAssertion(rdfs:label :D "d"^^xsd:string)

# Class: :P (p)

AnnotationAssertion(rdfs:label :P "p"^^xsd:string)

# Class: :Q (q)

AnnotationAssertion(rdfs:label :Q "q"^^xsd:string)


)
```

Related page

- Concept Modeling Semantics