```
:SemiDetached owl:disjointWith :Apartment .
```

the :SemiDetached class will be empty. Why? If the class :Apartment covers both :FurnishedApartment and its complement, :Apartment will be equivalent to owl:Thing: there cannot be an individual not belonging to a class nor its complement. If we then additionally introduce a class that is disjoint with :Apartment, this class is effectively disjoint with owl:Thing. The :SemiDetached class cannot contain any individuals, and is thereby equivalent to owl:Nothing.

Union and Disjoint Union We often know for some class that it is equivalent to two or more other classes: every member of the class is a member of at least one of the classes in the union. This can be specified using the owl:unionOf construct. For example:

```
:Apartment   rdf:type      owl:Class ;
        owl:unionOf   ( :ColdWaterFlat
                        :LuxuryApartment
                        :PenthouseApartment
                        :StudioApartment
                        :BasementApartment
                        :FurnishedApartment
                        :UnFurnishedApartment
                      ) .
```

In many cases, the member classes of the union are mutually disjoint. Of course, we can explicitly assert owl:disjointWith relations between each class, but it is more convenient to state this directly:

```
:Apartment   rdf:type        owl:Class;
        owl:disjointUnionOf (
                             :FurnishedApartment
                             :UnFurnishedApartment ) .
```