This defines the class of all apartments in Amsterdam. Arguably, this type of class definition can be very cumbersome if the list of known members is very long, or even impossible if we do not currently know all individuals. For instance, we may decide to knock down the wall between :BaronWayBedroom1 and :BaronWayBedroom2, creating a new room.

Disjoint Classes Disjointness of classes means that no member of one class can also be a member of the other class. The sets of individuals described by the classes do not overlap. We can say that the :LuxuryApartment class is disjoint from :ColdWaterFlat using the owl:disjointWith property:[7]

```
:LuxuryApartment owl:disjointWith :ColdWaterFlat .
```

This means that no :LuxuryApartment can be a :ColdWaterFlat at the same time.

Complement The complement C of a class A is the class of *all* things not belonging to A. In other words, the union of A and C is equivalent to owl:Thing. Note that this means that the complement is always a superclass of the disjoint classes of A. It is important to keep in mind that complementarity is a very powerful modeling construct that should be used with care. For instance:

```
:FurnishedApartment rdfs:subClassOf  :Apartment .
:UnFurnishedApartment  rdfs:subClassOf  :Apartment ;
                       owl:complementOf :FurnishedApartment .
```

This states that the class of furnished apartments is the complement of the class of apartments without furnishing. This is problematic if our ontology contains other classes besides apartments. For instance, if we additionally state:

[7]A cold water flat is an apartment which has no running hot water.