property to state that the restricted class is exactly the class described by the restriction. The rdfs:subClassOf restriction states *necessary* conditions for class membership, while the owl:equivalentClass restriction states *necessary and sufficient* conditions.

In general, a reasoner can only directly infer class membership for individuals based on both necessary and sufficient conditions. For instance, the existential restriction above will not make a reasoner conclude that every individual that has a :hasBathroom relation with an individual of type :LuxuryBathroom must be an instance of :LuxuryBathroomApartment. The apartment is only a *subclass* of the restriction, and we do not have enough information to determine whether the individual is also a member of the class itself. If we make the class *equivalent* to the class specified by the restriction, it is clear that any individual that satisfies the restriction must also be a member of the class.

However, in both cases, if we explicitly assert an individual to be an instance of the :LuxuryBathroomApartment class, the reasoner *will* infer that there is at least some (unknown) individual of type :LuxuryBathroom as value for the :hasBathroom property.

Value Restrictions Value restrictions come in handy when we want to define a class based on relations with known individuals, or specific values for datatype properties. For example, we can define the class of all apartments in Amsterdam:

```
:AmsterdamApartment
    rdf:type             owl:Class;
    owl:equivalentClass  [ rdf:type        owl:Restriction;
                           owl:onProperty  dbpedia-owl:location ;
                           owl:hasValue    dbpedia:Amsterdam
                         ] .
```