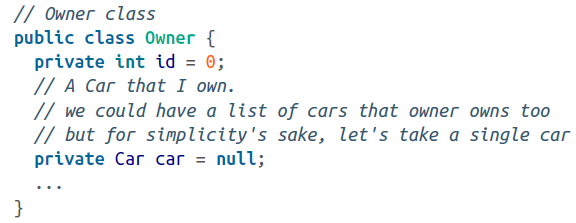
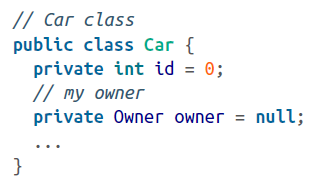
**Directionality**

The other facet of association is directionality. This property defines the direction to which the association is tending. For example, in a *Car* and *Engine* association, by querying *Car’s* attribute, we can figure out the engine; that is if you give me a Car Object, I can tell you what engine it has. Take another example of a student attending university. By querying the *courses* attribute on the *Student* object, we could tell what courses he’s enrolled in.

The association cab be either *unidirectional* or *bidirectional.* So, in the *car* and *engine* example, we can get the make of the engine if we are provided with the car data. However, if I provide you with the engine, would you be able to tell me the car model? Unfortunately, we can’t get the car details from the engine. This type of association is ***unidirectional****:* the directionality is one-sided only. In java, we create a reference of the target object in the source class but not the other way. In the *Car* and *Engine* snippets shown previously, you can see that *Car* has a reference to an engine, but the *Engine* class does not have a reference to a car (hence it can’t get a car for you!).

On the other hand, if we can navigate from source object to target object or vice versa, that relationship is said to be ***bidirectional****.* In the case of *Car* and *Owner,* we can derive the owner of the car given the *Car* object, as well as the owner’s car given the *Owner* object. See the following definition of *Owner* and *Car:*

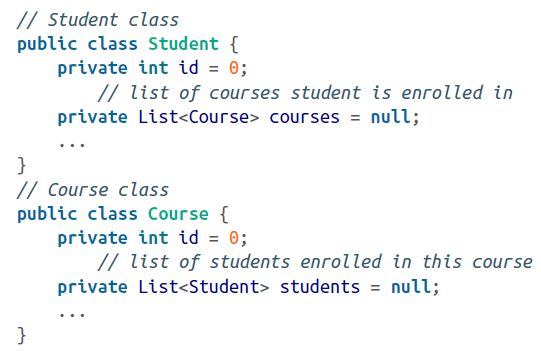




So, basically, what we are doing here is providing a reference of *Car* in the *Owner* object and a reference of *Owner* in the *Car* object to maintain a bidirectional association.

Let’s throw multiplicity in this mix. The preceding example is a *one-to-one bidirectional association.*

How about a many-to-many bidirectional association? The student and course example exhibits this association. We can derive a student’s courses from the *Student* object. At the same time, we can also deduce the list of students enrolled in a particular course by querying the *Course* object. This is defined in the *Student* and *Course* snippets shown here:



Did you notice that both classes have a reference to each other plus collection variables? This is the way we express many-to-many association with bidirectionality.

Now that we have a basic understanding of associations, let’s see how they can be developed and implemented in Java one by one.