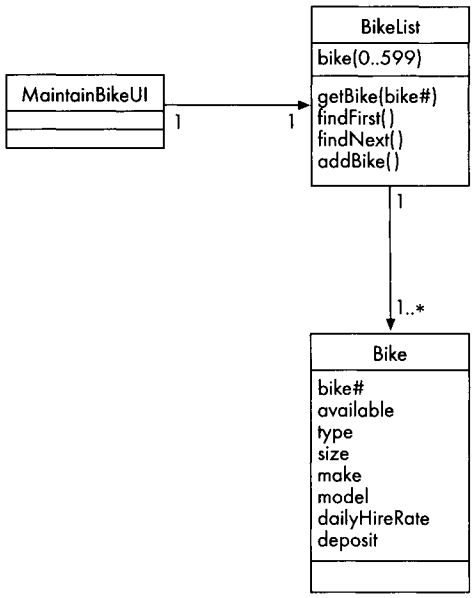

Figure 10.5 *A collection class used to implement a one to many association*

sometimes to all of the :Bikes. To do this :MaintainBikeUI has to know the identifiers of all the Bike objects. It could hold these in a simple array; however, it would also need operations to manage the array: to facilitate adding, deleting and finding bike identifiers. The problem with this is partly that this is not the job of a UI class and partly that other classes (for example the IssueBikeUI class) will need the same functionality: they too will need a set of :Bike identifiers and code to manipulate it. It would be more efficient to create a class specially to hold and manipulate the identifiers, a *collection class.*

Figure 10.5 shows BikeList, a collection class which can find a specific bike by bike#, add and delete bikes, etc. MaintainBikeUI has a 1:1 relationship with BikeList. This association therefore can be implemented as explained above, by keeping a reference to BikeList in MaintainBikeUI. A :BikeList will contain a list of :Bike references – bike[0..599] (in arrays we count from zero).

Accessing a particular :Bike will be done as follows. :MaintainBikeUI will send a getBike(bike#) message to :BikeList asking it to find a bike with a particular bike#. :BikeList will iterate through the list of :Bikes looking for one whose bike# matches. The identifier of this :Bike will then be returned to :MaintainBikeUI which can then send a message directly to that :Bike. This interaction is modelled in the sequence diagram in Figure 10.6.