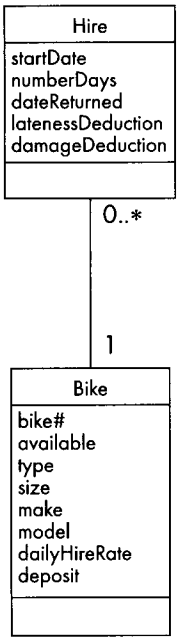

Figure 5.12 *One to many relationship between the Bike and Hire classes*

care; is the Hire object (:Hire[2]) for one Bike object (:Bike) or many? Requirement R6 (Figure 5.2) states that the system must 'cope with a customer who hires more than one bike, each for different amounts of time', therefore we might be dealing with more than one set of dates, but we only want one set per :Hire. Also the damageDeduction attribute in the Hire class relates to a specific bike. Therefore a :Hire is for one :Bike only. A :Bike, on the other hand, can be hired many times or may not be hired at all. The same applies to a SpecialistBike object. The relationship between Bike and Hire will therefore be one to many, as in Figure 5.12.

A :Customer can make many :Hires, but a :Hire is a specific transaction relating to just one :Customer. A :Payment is made by just one :Customer, but a :Customer can make many :Payments. There is no suggestion that one class is part of another one here, so there does not seem to be a reason to make any of these relationships aggregations. This gives us the associations shown in Figure 5.13.

Generalization and inheritance. When we are looking for generalization relationships in a diagram, we look for classes that share attributes and behaviour. Looking at the diagram in Figure 5.13, we can see that Bike and SpecialistBike share the

2. *Remember that an object of a class can be referred to in a number of ways, for example: a Hire object; an object of the Hire class; or :Hire.*