Identity. When we say that an object has identity, we mean that each object is unique, it has a separate existence and ultimately a separate space in the computer memory from every other object. Each bike in the Wheels system will be represented by a separate object in the code. Even two objects whose attributes have identical values are totally distinct from one another. In the computer implementation, objects are identified by a unique computer-generated reference (the object-id) which is an internal code not normally visible to the programmer; roughly speaking it corresponds to a memory location. This is quite separate from any business related identifier such as a customer number or a bike number, or any programmer-generated name for an object. Wheels' bikes all have numbers allocated by the bike shop for business purposes. These numbers are recorded as one of the attributes of a bike object, but they are not the way the computer software identifies the bike object. As far as the programmer or other parts of the code are concerned, objects must be named and addressed by their name. For example, if the list of Wheels' 600 bikes is stored in an array wheelsBikes[600] and we want to know the daily hire rate of one of the bikes in the array, we would send a message addressed to that particular bike (say number 105 in the array) asking it to display its daily hire rate – wheelsBikes[105].showDailyHireRate(). wheelsBikes[105] is the name of the bike object we are interested in, showDailyHireRate() is the operation we want it to perform.

Encapsulation and data hiding

One of the advantages of object-oriented software is that objects encapsulate data. Data can be hidden inside an object in such a way that it is protected and cannot be directly accessed by other parts of the program. The big advantage of this is that it cannot be accidentally corrupted. As an example, let us design a counter to keep track of the score in a game of rugby. We shall have an object called blueSide :Counter with an attribute score as in Figure 4.3 – blueSide is the name of the object, Counter is its class.

The scoring system in a game of rugby is that a side gets 3 points for a penalty, 3 for a drop goal, 5 for a try and 2 for a

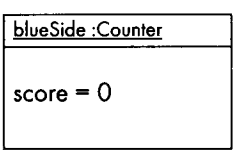

Figure 4.3 *Counter object*