The seamless development also provides traceability of user requirements from initial identification through to the code. In the traditional structured approach to development some of the models that were used to capture user requirements (such as the data flow diagram) were abandoned at the design stage where a new modelling technique was used to specify the structure of the code (for example, structure charts). This made it hard to ensure that all of the user requirements were incorporated in the final implementation: it was easy to lose features in the changeover of models. In the object-oriented approach we identify objects which represent things in the problem domain and about which we need to store information. Later in the development process we add objects to provide the software structure we want to implement (for example, control and boundary objects) and objects that are to do with how the software will work (buttons, windows, mouse-listeners etc.). However, the original objects, although they may gather some extra features in the development process, will still be identifiable in the code.

What is an object?

The object is the most important concept in object-oriented software development; object-oriented systems are based on the object. At its simplest, an object is a representation of something in the application area about which we need to store data to enable the system to do what the users want it to. In the Wheels system, for example, we will undoubtedly want to store data about bikes. Bikes, therefore, would be objects in our model of the Wheels system and eventually in the code. The bits of data we need to store about these bikes, such as the type, the daily hire rate and the deposit, are known as the *attributes* of the bike object.

In the UML an object is represented as shown in Figure 4.2 as a rectangle with two sections. The top section is for the name of the object, and the second section is used for the object's attribute values.

In the case of Figure 4.2 the name of the object is aBike :Bike. Object names are always underlined and can have two parts, either of which can be used on its own. The first part of the name, aBike, labels the object; the second part, :Bike, identifies it as an object of the Bike class. There are a number of ways in which we can refer to an object: we can talk about 'an object of the Bike class', 'a Bike object', or ':Bike' – all of these names mean the same thing in this context.

Every object belongs to a *class* which is a template or factory for producing objects. All of the objects of a class have the same