yourself that this is the case, try doing this as an exercise – see Exercise 4.14 based on the Robot hierarchy that is described in the Technical points section of this chapter.

- The program does not know in advance which classes are involved in the processing of the array; as far as it is concerned it is dealing with an array of European objects. As the program works its way through the array, it only knows which methods will be executed when it gets to the next object and works out which class it belongs to. This is known as *late or dynamic binding.*

You will find a more complex example of polymorphism, the Robot hierarchy, in the Technical points section of this chapter.

Technical points

What makes a good class?

Knowing what makes a good class comes with experience, but we can offer a few pointers.

Problem domain. During analysis, classes should correspond to things in the real world of the problem domain – Bike, Customer, Hire and Payment all correspond to things in the Wheels problem domain that the user would know about and understand.

Functionality. A class (at least during analysis) usually has both attributes and behaviour. Be suspicious of a class that seems to be just one big function and has no attributes; it might turn out to be just an operation on some other class. Similarly, a class that has only attributes and basic set and get[7] operations sounds like a badly designed class. The functionality of the system is shared between the classes, so they should be doing more than just maintaining and displaying the values of their attributes.

Cohesion. One of the qualities of a good software construct, listed at the beginning of this chapter, is cohesion. A class is cohesive if it is concerned with only one thing, if all its attributes and operations relate to the same topic. For example, Figure 4.23 shows a version of the art gallery Painting, one of the classes from the hierarchy that we mentioned earlier in the chapter (we have omitted the rest of the classes for simplicity).

[7] *A set operation sets or modifies the value of an attribute. A get operation displays or retrieves the value of an attribute. Set and get operations are not normally shown on a class diagram.*