need to revisit the sequence of events in each scenario, this time looking at them in terms of the messages between objects spawned by each event. This is what interaction diagrams do. There are two types of interaction diagram: sequence and collaboration. Each shows more or less the same information, but with a different emphasis. We shall describe each in turn.

Sequence diagrams

When students who are new to object-oriented technology, especially those who have been trained in procedural methods, first meet object-oriented code, they are staggered by the way the flow of control jumps about on the page. In procedural programming, the sequence of execution proceeds in an orderly fashion from the top of a page of code to the bottom, with only the odd jump off to a procedure. Quite the opposite happens in the execution of an object-oriented program; the sequence of control jumps from object to object in an apparently random manner. This is because, while the code is structured into classes, the sequence of events is dictated by the use case scenarios. Even for those experienced in object technology, it is very hard to follow the overall flow of control in object-oriented code. Programmers, maintainers and developers need a route map to guide them; this is provided by the sequence diagram. The specification of the functionality in a sequence diagram is literally a sequence of messages, but the object-oriented code underlying the sequence of functionality in the diagram jumps about because the code is structured into classes not functions.

Sequence diagrams show clearly and simply the flow of control between objects required to execute a scenario. A scenario outlines the sequence of steps in one instance of a use case from the user's side of the computer screen, a sequence diagram shows how these steps translate into messaging between objects on the computer's side of the screen. We will illustrate this by walking through a simple scenario, Figure 6.7 (repeated from Figure 6.2), translating it into a sequence diagram.

As we saw in Figure 6.3 the objects required by the 'Issue bike' use case are :Bike, :Customer, :Hire and :Payment. The sequence diagram displays this collaboration horizontally across the page as in Figure 6.8.

The order in which the objects appear is not important. The Receptionist icon is the actor involved in the 'Issue bike' use case. As far as interaction diagrams are concerned, actor is treated as a special sort of object. On a sequence diagram produced during analysis she represents the system interface – she inputs information and receives the output from the system. The dashed vertical line below each object symbol is called the object's *lifeline*. It represents the object's life for the duration of the scenario we are enacting. A message is represented as a labelled arrow from one