duration in days. We can do this using the Hire class constructor, Hire(startDate, no.Days).

- Again, a lot of detail is omitted from the diagram. We are not saying anything about where the parameters for the hire constructor come from. We wouldn't want the Receptionist to have to enter the number of days again, so presumably, some sort of interface or control object will deal with holding on to these details when they were entered the first time.

The last three steps in the scenario are: .

9 Stephanie pays the £74
10 Annie records this on the system and the system prints out a receipt
11 Stephanie agrees to bring the bike back by 5.00pm on the following Saturday

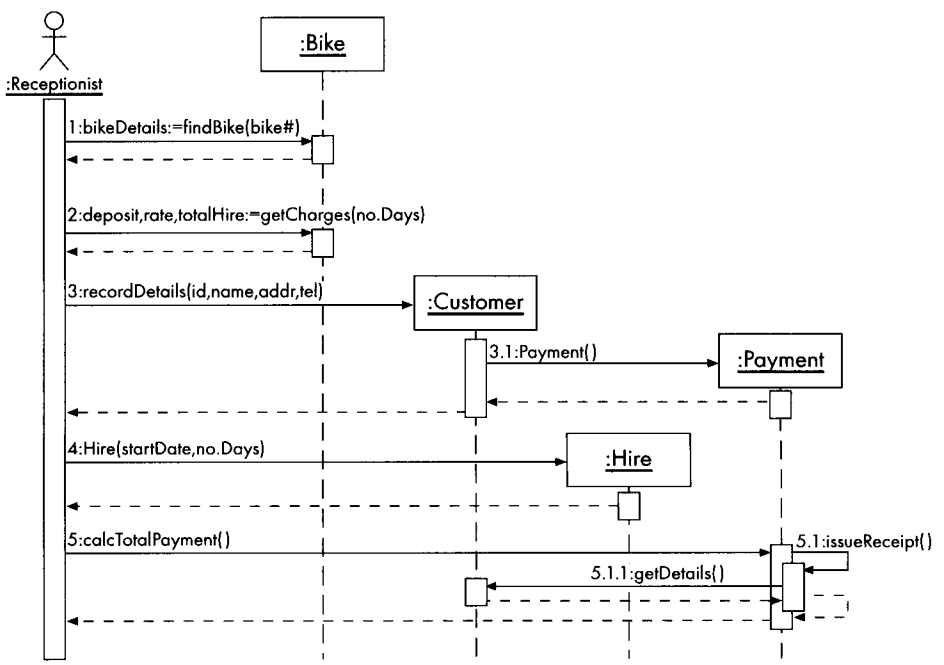

Figure 6.12 *Complete sequence diagram for the 'Issue bike' scenario*

- The system is required to record what the customer has paid, both in hire charges and as a deposit. A Payment object will calculate how much the customer owes and record these figures. It makes sense to create a Payment object from the Customer object, so that they are permanently linked, see Figure 6.12.

- Notice that the message from :Customer to :Payment is numbered 3.1 rather than 4. The UML numbering style for interaction diagrams emphasizes the nesting of the messages, rather than the