**Use case:** Issue bike
**Preconditions:** 'Maintain bike list' must have been executed
**Actors:** Receptionist
**Goal:** To hire out a bike

**Overview:**
When a customer comes into the shop they choose a bike to hire. The Receptionist looks up the bike on the system and tells the customer how much it will cost to hire the bike for a specified period. The customer pays, is issued with a receipt, then leaves with the bike.

**Cross-reference:**
R3, R4, R5, R6, R7, R8, R9, R10

**Typical course of events:**

| Actor action | System response |
|---|---|
| 1 The customer chooses a bike | |
| 2 The Receptionist keys in the bike number | 3 Displays the bike details including the daily hire rate and deposit |
| 4 Customer specifies length of hire | |
| 5 Receptionist keys this in | 6 Displays total hire cost |
| 7 Customer agrees the price | |
| 8 Receptionist keys in the customer details | 9 Displays customer details |
| 10 Customer pays the total cost | |
| 11 Receptionist records amount paid | 12 Prints a receipt |

**Alternative courses:**

| | |
|---|---|
| Steps 8 and 9 | The customer details are already in the system so the Receptionist needs only to key in an identifier and the system will display the customer details. |
| Steps 7–12 | The customer may not be happy with the price and may terminate the transaction |

*Figure 6.5* *Use case description for 'Issue bike'*

providing information about bikes is the responsibility of the Bike class, so findBike() should be an operation on Bike, allowing us to retrieve information from the appropriate bike object.

2 We then want to be able to find out the hire cost for a specific number of days' hire. This again is the responsibility of the Bike class, so we need another operation on Bike, getCharges(no.Days).

3 The next step is to record the customer details. Knowing about customers is the responsibility of the Customer class. We need an operation recordDetails(custID, name, address, tel).