details about the bike (bike# + available + type + make + model + size + dailyHireRate + deposit)

We can see a more complex example in the description of the operation calcTotalPayment(amt, deposit) in the Payment class, as shown below:

calcTotalPayment(amt, deposit)

This operation calculates and records the sum of amounts paid as hire fees and the sum of deposits paid. This operation must find all current customer hire objects and for each one calculate the hire fee (Bike.dailyHireRate[1] * Hire.numberOfDays). The hire fees for all of the customer's hires are summed and recorded in Payment.totalAmountPaid. It also finds the deposit for each bike hired, sums them and records the result in Payment.totalDepositPaid.

As development progresses, we need to know more about how each operation carries out its processing. The UML does not provide any particular method for specifying operations, but activity diagrams (see Chapter 8) are an effective way of showing diagrammatically how an operation works.

An alternative approach, known as specification by contract, describes operations in terms of the services they deliver. This type of specification defines:

- The signature of the operation (its name, any arguments, and the type of values it returns)[2]
- The purpose of the operation
- What the client object must provide in order to obtain the required service
- A description of the internal logic of the operation
- Any other operations that are called by this operation
- Any attributes of objects whose values are changed by the operation.

1. *This notation means the attribute dailyHireRate in the class Bike.*
2. *This is a very important part of the specification because the signature of the operation is its public interface; as long as the signature remains unchanged, the internal details of the operation can be modified without affecting the rest of the system.*