The system is implemented in the object-oriented programming language Java. The main reason for this choice was simply that it is a popular programming language taught on many programming courses. It is also supported by Together, the CASE tool we have used to produce many of the UML models in this book.

For simplicity, the code reproduced in this chapter is only a partial implementation of the Wheels system: one scenario of the 'Issue bike' use case. The code is limited to handling one customer hiring one bike. This means that some of the methods, e.g. the Payment method, calculateTotalPayment(), seem pointless as they were designed to handle multiple bike hires. As the code we require for this chapter is for illustrative purposes only, it was more important to have simple code than fully functional code.

Many details, such as references to other classes, have been added to the classes. We have, therefore, omitted some attributes and methods that were present on the analysis model. This allows us to keep the diagrams within manageable proportions.

Throughout this chapter we refer to methods rather than operations[1]; this is appropriate at this stage because programmers are concerned with the body of code that implements a process. In earlier chapters we talked about operations rather than methods because analysis and design activities are more concerned with the interface of a process than its implementation.

The implementation class diagram

The implementation class diagram in Figure 11.1 was originally generated from the code by the CASE tool Together. It shows full implementation detail as described in Chapter 10, i.e. visibility, types and initial values of attributes, method parameters (with types and return types) and method return types. For the purpose of producing the diagram, Together ignores *gets* and *sets*, i.e. methods that simply set or return the value of an attribute. For example, Customer has three get methods, getCustomerNumber(), getName() and getPostcode(), which are not shown on the class diagram.

StartUp Class. StartUp is a new class. Applications in Java must always contain the method main(); it is always the first method to be executed when a Java application is run. In Java every method, including main(), must be inside a class; we invented the StartUp class partly to house main().

The other purpose of StartUp is to simulate what would happen if this code were part of a fully implemented system with a front

1. *For a discussion on the difference between operation and method see Chapter 4.*