computers and their peripherals. Deployment diagrams can also be used to show on which computers the various software components will be physically located. We include points to consider when designing a simple user interface. We discuss different strategies for dealing with persistent data. Guidelines are given for linking a Java application to a relational database.

Architecture

When we specify the architecture of the system we are describing the software and hardware components of the system, how they are structured and related. Software components can be described logically in terms of classes, packages, subsystems and their dependencies, or physically in terms of executable files, class libraries and databases. The logical software architecture is modelled using class diagrams and package diagrams; the physical software architecture is described using component diagrams (see below). Hardware architecture, from the system developer's perspective, is concerned with the computers, peripherals and networks on which the system will run. Hardware architecture is modelled using deployment diagrams.

Layered architecture

One approach to partitioning a system is to use a *layered architecture*. However, before we embark on a discussion of a layered architecture, we need to know about entity, boundary and control classes, about the visibility of classes, attributes and operations and about partitioning the system using packages.

Entity, boundary and control classes. The classes that we have considered during analysis have all been concerned with modelling the system requirements. These classes are variously referred to as *entity classes*, domain classes or application classes. All of the classes we have met so far have been entity classes. Entity classes model features of the problem domain, like bikes, customers and hires, and are usually classes which have data that needs persistent storage. They are capable of providing the functionality specified in the use cases. However, we have not considered *how* the entity classes will provide the functionality. During system design, we need to add classes to handle the human interface and to control the sequence of execution; these are called *boundary* (*or user interface*) and *control classes*.

Boundary classes model the system's interface with its actors, i.e. with the user or with other systems. These classes are used to capture user input and present results to the user. They take the