Designing operations. During detailed design, we look more closely at operations or more precisely at the methods that implement the operations. We design the algorithms that will be used. Our decisions can be documented using an activity diagram (see Chapter 8), or one of the techniques for describing processes described in Chapter 6, structured English, decision tables, decision trees or specification by contract.

In general, when designing (and when programming), it is best to keep algorithms as simple as possible. If you are tempted to use clever and intellectually satisfying or performance enhancing techniques, think of the poor maintenance programmer who will be forced to follow your tortuous thought patterns.

Interaction diagrams

During analysis we are using interaction diagrams primarily to work out how groups of objects will collaborate in a process, usually in the form of a use case scenario. At this stage we are still ironing out the wrinkles in the allocation of responsibilities to classes and the implications of this in terms of operations on classes and messaging between objects. During design we focus on the code; we use the interaction diagrams, essentially, to give an abstract view of what is happening when the program is running. To achieve this we need to add more detail to the models: we discuss how to specify the creation and deletion of objects, how to specify conditional behaviour and repeated behaviour, how to use interaction diagrams at different levels of detail using multiple objects and packages.

In the following section we introduce you to a lot of extra notation which will allow you to build more information into your sequence diagrams. However, it is important to remember that the greatest appeal of sequence diagrams is their simplicity, so use the extra notation with care. If you try to show too much detail on a sequence diagram it loses its chief attraction.

Object creation and deletion. During the execution of a use case scenario, the group of objects is not necessarily fixed; new objects can be created, existing objects can be deleted. Objects that are created or deleted in an interaction are called *transient* objects.

We have already met the UML notation for object creation in Chapter 6; instead of appearing at the top of the page the object icon appears further down the page, at the point where it is created. The notation for object destruction is a large X at the bottom of the object's lifeline, see Figure 10.7. Objects can either self-destruct or be destroyed by receiving a message from another