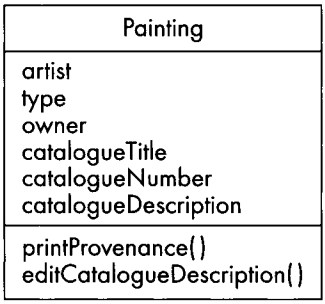

Figure 4.23 *Adding attributes and operations relating to a catalogue destroys the cohesion of the Painting class*

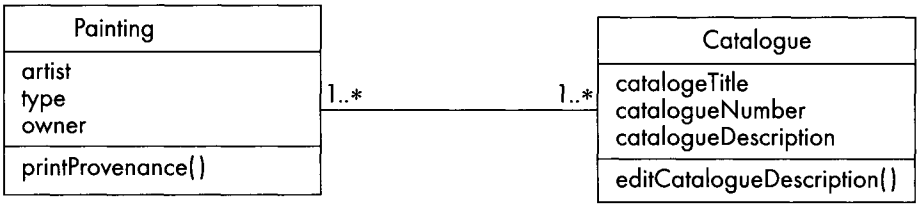

Figure 4.24 *The Painting and Catalogue classes are linked by association*

In the version of the Painting class in Figure 4.23 we have added the attributes catalogueTitle, catalogueNumber and catalogueDescription, and the operation editCatalogueDescription(). This makes the Painting class unbalanced; it is no longer cohesive because these three attributes and the operation belong in a different class, one concerned with catalogues. The two separate classes could be linked by association as in Figure 4.24.

Substitutability. In an inheritance hierarchy, objects of descendant classes should always be substitutable for objects above them in the hierarchy. This is what we saw happening in the European example that we used to illustrate polymorphism. We declared an array of European objects and populated it with objects of European subclasses (see Figure 4.21). The subclass objects could all be treated as if they were European objects; any message that could be understood by a European object could be understood by its descendants. For this to happen we need to construct our inheritance hierarchies with care.

As an example of how not to construct an inheritance tree, we will revisit the art gallery example (see Figure 4.14). We started with the two classes in Figure 4.13, reproduced here in Figure 4.25.