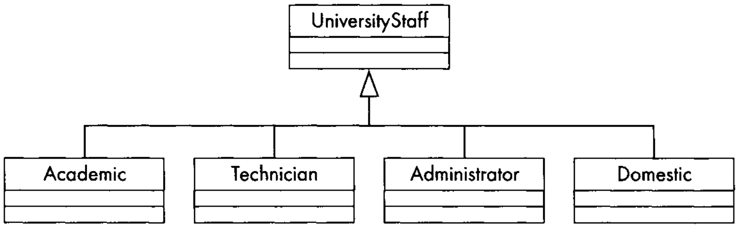

Figure 4.17 *UniversityStaff is an abstract class that is never instantiated*

instantiated in other words no objects of the class exist in the system. In the art gallery example in Figure 4.14, if we assume that a picture must be either a painting or a photograph, there will never be objects that are just pictures. Similarly, in the hierarchy in Figure 4.17, the class UniversityStaff is never instantiated; all employees of the university have to be either academics, technicians, administrators or domestics.

Classes that are never instantiated are known as *abstract* classes; the opposite is a *concrete* or *instantiated* class. The UML notation for an abstract class is {abstract} placed below the class name – see Figure 4.18.

Abstract classes are often very useful candidates for class libraries; their generality means they capture the essence of their type of class without any taint of the specific system for which they were designed.

Polymorphism. To discuss polymorphism, we must first understand the difference between operations and methods. The two terms are generally used as if they were interchangeable, but this is not quite the case. We have described an operation as a process that an object can perform, part of its behaviour. Strictly speaking the word *operation* refers to the interface of the process; the operation must be invoked (by message passing) when a client module wants the process to be executed. The code that implements the process is known as the *method*.[5]

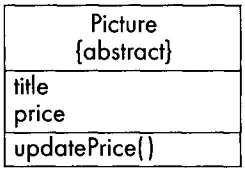

Figure 4.18 *The UML notation for an abstract class*

5. *Programmers often refer to operations as* method signatures.