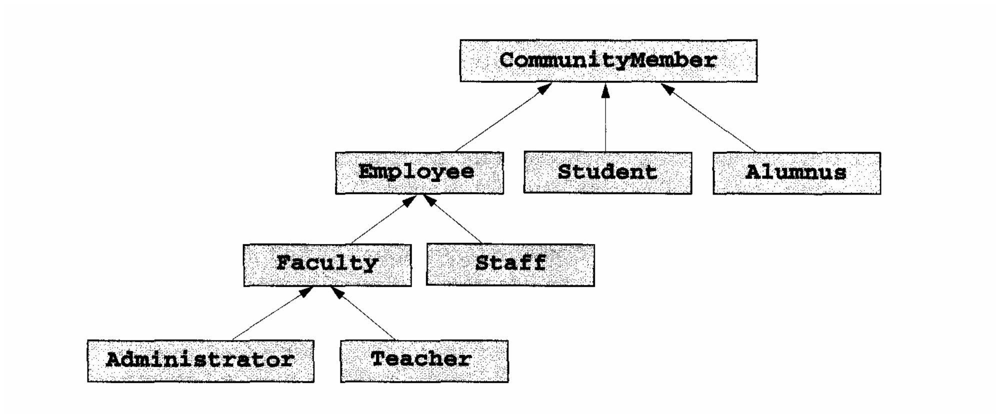
Often, an object of one class "is an" object of another class, as well. For example, a rectangle is a quadrilateral (as are squares, parallelograms and trapezoids). Thus, class Rectangle can be said to inherit from class Quadrilateral. In this context, class Quadrilateral is a base class, and class Rectangle is a derived class. A rectangle is a specific type of quadrilateral, but it is incorrect to claim that a quadrilateral is a rectangle - the quadrilateral could be a parallelogram or some other type of Quadrilateral. Figure 9.1 lists several simple examples of base classes and derived classes.

Every derived-class object "is an" object of its base class, and one base class can have many derived classes; therefore, the set of objects represented by a base class typically is larger than the set of objects represented by any of its derived classes. For example, the base class Vehicle represents all vehicles, including cars, trucks, boats, bicycles and so on. By contrast, derived-class Car represents only a small subset of all Vehicles.

Inheritance relationships form tree-like hierarchical structures. A class exists in a hierarchical relationship with its derived classes. Although classes can exist independently, once they are employed in inheritance arrangements, they become affiliated with other classes. A class becomes either a base class, supplying data and behaviors to other classes, or a derived class, inheriting its data and behaviors from other classes.

Let us develop a simple inheritance hierarchy. A university community has thousands of members. These members consist of employees, students and alumni. Employees are either faculty members or staff members. Faculty members are either administrators (such as deans and department chairpersons) or teachers. This organizational structure yields the inheritance hierarchy, depicted in following figure.



Note that the inheritance hierarchy could contain many other classes. For example, students can be graduate or undergraduate students.

Each arrow in the hierarchy represents an "is-a" relationship. For example, as we follow the arrows in this class hierarchy, we can state, "an Employee is a CommunityMember" and "a Teacher is a Faculty member." CommunityMember is the direct base class of Employee, Student and Alumnus. In addition, CommunityMember is an indirect base class of all the other classes in the hierarchy diagram.

Starting from the bottom of the diagram, the reader can follow the arrows and apply the "is-a" relationship to the topmost base class. For example, an Administrator is a Faculty member, is an Employee and is a CoinmunityMember.

In C#, an Administrator also is an Object, because all classes in C# have Object as either a direct or indirect base class. Thus, all classes in C# are connected via a hierarchical relationship

In general, when we have a hierarchal relationship, a class-declaration is has the following form:

```
class-modifiers class base class identifier
{
    Class body
}
```

A class-declaration consists of class modifiers followed by the keyword class and an identifier that names the class, followed by an optional class-base specification followed by a class-body

Class Modifiers can be:

private
public
protected
sealed
abstract

Example6_2_1:

```
class Animal
{
   string _Name;
   int _Age;

   public Animal(string Name,int Age)
   {
      _Name = Name;
      _Age=Age;
   }
   public int Age
   {
      set { _Age = value; }
      get { return _Age; }
   }
   public virtual void Walk()
   {
      MessageBox.Show(" An Animal is walking");
   }
   public virtual void Talk()
   {
      MessageBox.Show(" an Animal makes some sound");
   }
```