 设想这样一个需求，我们需要为自己的框架提供一个负责排序的组件。目前需要实现的是冒泡排序算法和快速排序算法，根据“ 面向接口编程 ”的，我们可以为这些排序算法提供一个统一的接口ISort，在这个接口中有一个方法Sort()，它能接受一个object数组参数。对数组进行排序后，返回该数组。接口的定义如下：public interface ISort{ void Sort(ref object[] beSorted).} 其类图如下： 然而一般对于排序而言，排列是有顺序之分的，例如升序，或者降序，返回的结果也不相同。最简单的方法我们可以利用if语句来实现这一目的，例如在QuickSort类中： public class QuickSort:ISort{ private string m_SortType. public QuickSort(string sortType) { m_SortType = sortType. } public void Sort(ref object[] beSorted) { if (m_SortType.ToUpper().Trim() == “ ASCENDING ”) { //执行升序的快速排序. } else { //执行降序的快速排序. } } } 当然，我们也可以将string类型的SortType定义为枚举类型，减少出现错误的可能性。然而仔细阅读代码，我们可以发现这样的代码是非常僵化的，一旦需要扩展，如果要求我们增加新的排序顺序，例如字典顺序，那么我们面临的工作会非常繁重。也就是说，变化产生了。通过分析，我们发现所谓排序的顺序，恰恰是排序算法中最关键的一环，它决定了谁排列在前，谁排列在后。然而它并不属于排序算法，而是一种比较的策略，后者说是比较的行为。