Objective-C语言中的协议类似于java语言中的接口，定义一套方法，让使用者自己实现，其中这最有用的是实现委托模式。分类可以让开发者把某个类的实现文件分开在不同的文件中，分而治之，除此之外，分类还可以给现成的类，比如NSString添加别的方法。

第二十三条：通过委托和数据源协议进行对象间的通信

此模式的主旨为：定义一套接口，某个对象如果想接受另一个对象的委托，则要遵从此接口，以便成为其“委托对象”，而另一个对象则可以给其委托对象回传一些信息，也可以在发生相关事件时通知委托对象。此模式可以将数据和业务逻辑解耦。

委托协议名通常是在类名后面加上Delegate一词，采用驼峰式书写的，一定要命名得当，应该准确的说明当前发生的事情以及delegate为何获知此事件。

注意：存放委托对象的属性需要定义为**weak类型而非strong**，因为两者之间必须为非拥有的关系。否则由于委托对象本来就包含此对象，为扮演委托者的对象也拥有此对象就会造成循环应用问题。

数据源模式是定义一套接口，令某个类经由此接口获得所需要的数据；而普通的委托模式中是信息从类流向委托者。

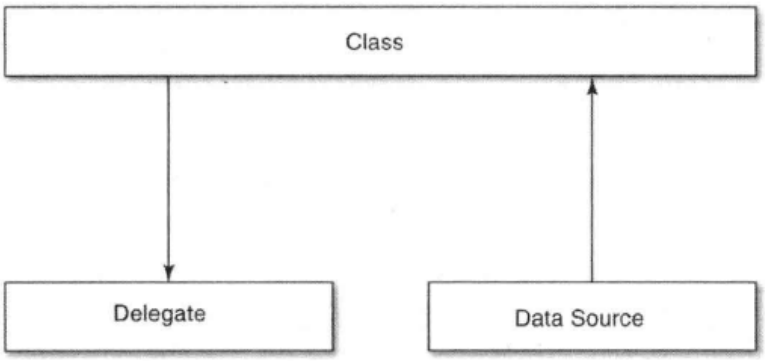

图 4-3　在信息源模式中，信息从数据源流向类，而在普通的委托模式中，信息则从类流向受委托者

要点：

1. 委托模式为对象提供了一套接口，使其可以将相关事件告知其他对象；
2. 将委托对象应该支持的接口定义为协议，在协议中把需要处理的事件定义为方法；
3. 当某个对象需要从另一个对象获得数据时，可以使用委托模式，称为“数据源模式”；
4. 如有必要，可以实现含有位段的结构体，将委托对象是否能响应协议中的方法这一信息缓存在其中。

第二十四条：将类的实现代码分散到便于管理的数个分类中

要点：

1. 使用分类机制把类的实现代码划分到易于管理的小块；
2. 将应该视为“私有”的方法归入到Private的分类中，以隐藏实现细节。

第二十五条：总是为第三方类的分类名称加前缀

要点：

1. 向第三方类中添加分类时，总应该给其名称加上你专用的前缀。因为为类添加的分类实现是在运行期添加到类的方法列表中的，因为类的方法列表中不会出现相同名称的不同实现，这样如果不同的分类或者不同的开发者所写的分类

如果名称相同，后面加载的分类中的方法就会覆盖掉前面加载的方法，造成隐晦的问题；
2. 向第三方类中添加分类时，总应该为其中的方法加上你的专属前缀。

第二十六条：勿在分类中声明属性

要点：

1. 把封装数据所用的全部属性都定义在主接口里面；
2. 在“class-continuation分类”之外的其他分类中，可以定义存取方法，但是最好不要定义属性。

第二十七条：使用“class-continuation分类“（延展extension）隐藏实现细节

要点：

1. 通过”class-continuation分类“向类中新增实例变量；
2. 如果某属性在主接口中声明为”只读“，而类的内部又要设置方法修改此属性，那么就在”class-continuation分类“中将其扩展为”可读写“；
3. 把私有方法的原型声明在”class-continuation分类“里面；
4. 若想使类所所遵从的协议不为人所知，则可于”class-continuation分类“中声明。

第二十八条：通过协议提供匿名对象

Objective-C中的匿名对象和其他语言中的匿名对象不同，在其他语言中一般指的是以内联形式创建出的无名类或者对象；而objective-c中具体指形如

```
@property (nonatomic,weak) id <SomeDelegate> delegate;
-(void)setObject:(id)object forKey:(id<NSCoping>)key;
```

因为该属性的类型为id <SomeDelegate>，实际上任何类都可以充当这一属性，但是这个对象必须遵循SomeDelegate，对于具备此属性的对象来说，delegate就是“匿名的”。

要点：

1. 协议可在某种程度上提供匿名类型。具体的对象类型可以淡化成遵从某个协议的id类型，协议里面规定了对象所应实现的方法；
2. 使用匿名对象来隐藏类型名称（或者类名）；
3. 如果具体类型不重要，重要的是对象能够响应（定义在协议里面的）特定方法，那么可以使用匿名对象来表示。