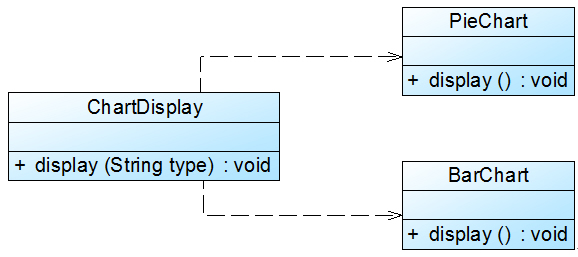
开闭原则(Open-Closed Principle, OCP)：一个软件实体应当对扩展开放，对修改关闭。即软件实体（可以指一个软件模块、一个由多个类组成的局部结构或一个独立的类）应尽量在不修改原有代码的情况下进行扩展。

产生背景：

任何软件都需要面临一个很重要的问题，即它们的需求会随时间的推移而发生变化。当软件系统需要面对新的需求时，我们应该尽量保证系统的设计框架是稳定的。如果一个软件设计符合开闭原则，那么可以非常方便地对系统进行扩展，而且在扩展时无须修改现有代码，**使得软件系统在拥有适应性和灵活性的同时具备较好的稳定性和延续性。**随着软件规模越来越大，软件寿命越来越长，软件维护成本越来越高，设计满足开闭原则的软件系统也变得越来越重要。

抽象化是开闭原则的关键。

实例：Sunny软件公司开发的CRM系统可以显示各种类型的图表，如饼状图和柱状图等，为了支持多种图表显示方式。



在ChartDisplay类的display()方法中存在如下代码片段：

......

if (type.equals("pie")) {

PieChart chart = new PieChart();

chart.display();

} else if (type.equals("bar")) {

BarChart chart = new BarChart();

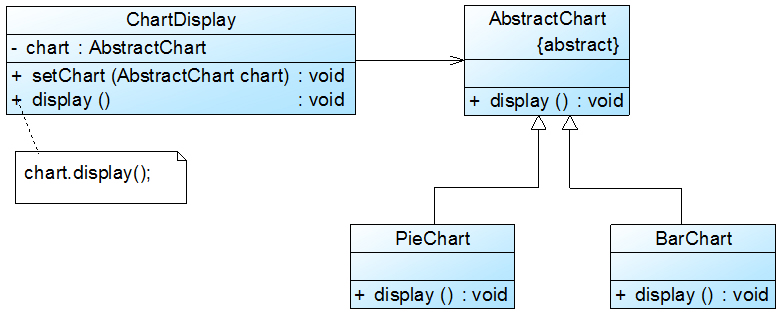
chart.display();

}

......

在该代码中，如果需要增加一个新的图表类，如折线图LineChart，则需要修改ChartDisplay类的display()方法的源代码，增加新的判断逻辑，违反了开闭原则。

根据开闭原则对该类进行重构：



我们引入了抽象图表类AbstractChart，且ChartDisplay针对抽象图表类进行编程，并通过setChart()方法由客户端来设置实例化的具体图表对象，在ChartDisplay的display()方法中调用chart对象的display()方法显示图表。如果需要增加一种新的图表，如折线图LineChart，只需要将LineChart也作为AbstractChart的子类，在客户端向ChartDisplay中注入一个LineChart对象即可，无须修改现有类库的源代码。

[面向对象设计原则之开闭原则](http://blog.csdn.net/lovelion/article/details/7537584" \t "https://blog.csdn.net/LoveLion/article/details/_blank)