（5）观察者模式

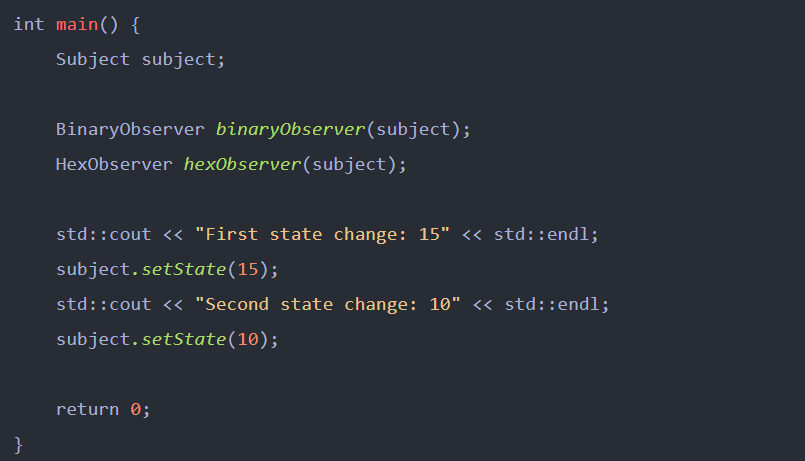
假设我们有一个名为“Subject”的主题类，它维护一个观察者列表，并提供了添加、删除和通知观察者的方法。我们还有一个名为“Observer”的观察者类，它包含一个名为“update”的方法，用于在主题状态发生变化时更新观察者。



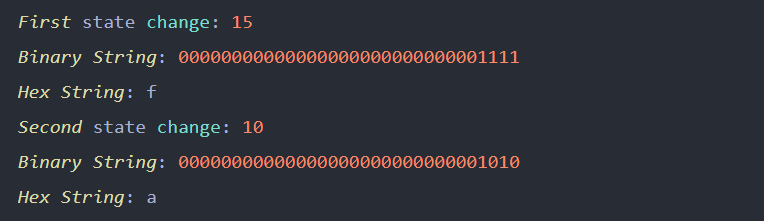
两个具体的观察者类，“BinaryObserver”和“HexObserver”，它们实现了“Observer”接口，并在“Subject”状态发生变化时更新自己的状态。



测试一下这些类的行为：



输出结果：



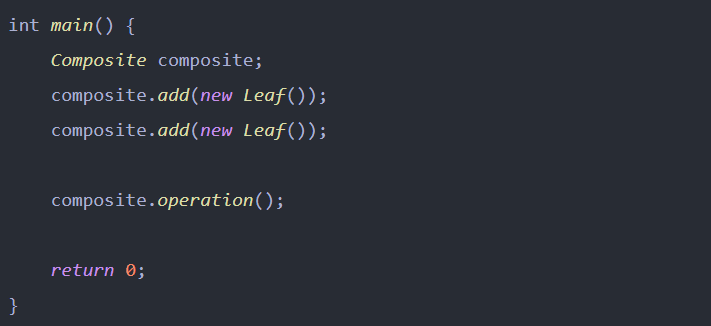
可以看出，当“Subject”的状态发生变化时，它的所有观察者都会收到通知并进行相应的更新。这就是观察者设计模式的基本原理。

（6）组合模式

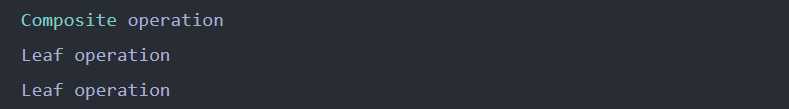
假设我们有一个名为“Component”的组件类，它定义了一个名为“operation”的纯虚函数，用于在组合中处理组件。我们还有一个名为“Composite”的组合类，它包含一个名为“children”的组件列表，以及添加、删除和获取组件的方法。最后，我们有一个名为“Leaf”的叶子类，它实现了“Component”接口，并在组合中表示最终节点。



创建一个组合，并将叶子节点添加到它的子节点列表中：



输出结果：



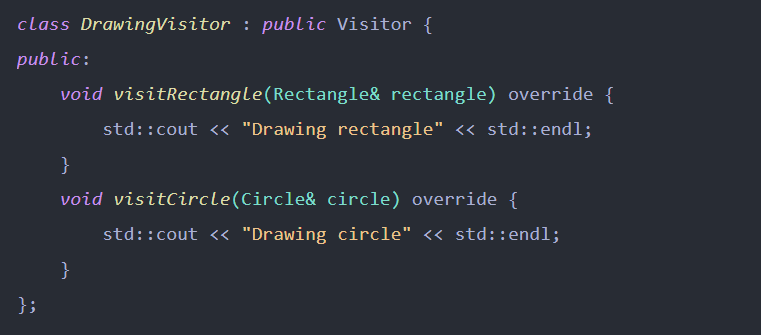
可以看出，组合模式允许我们将对象组合成树状结构，并且能够像处理单个对象一样处理整个树状结构。在上面的例子中，我们创建了一个名为“Composite”的组合，并将两个名为“Leaf”的叶子节点添加到它的子节点列表中。当我们调用“operation”方法时，组合会遍历整个树状结构，并将每个组件的操作作为单个操作处理。

（7）访问者模式

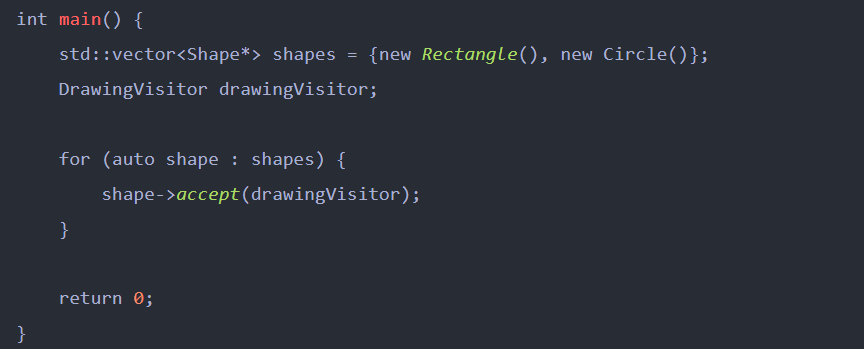
假设我们有两个类，“Circle”和“Rectangle”，它们都有一个名为“accept”的方法，用于接受访问者。我们还有一个名为“Visitor”的访问者类，它包含两个名为“visitCircle”和“visitRectangle”的方法，用于在访问这些形状对象时进行操作。



创建一个访问者，并定义两个操作：“drawRectangle”和“drawCircle”：



创建一些形状对象，并让访问者对它们进行操作：



输出结果：



可以看出，访问者模式允许我们在不改变已有类的前提下定义新的操作。在上面的例子中，我们创建了两个形状对象，“Rectangle”和“Circle”，并创建了一个名为“DrawingVisitor”的访问者，它定义了两个操作：“drawRectangle”和“drawCircle”。当我们对这些形状对象进行操作时，访问者会根据对象类型调用相应的操作。