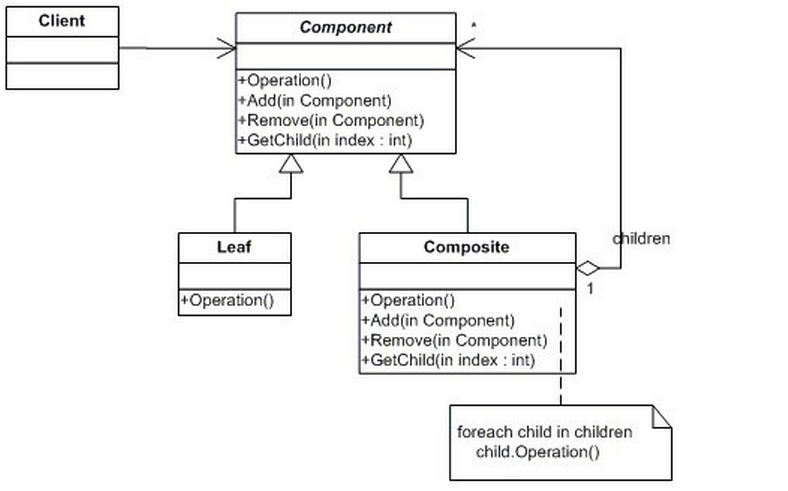
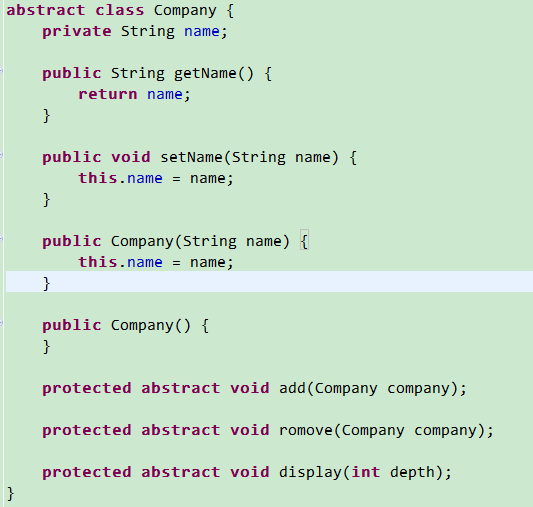
1. 定义：将对象组合成树形结构以表示“部分整体”的层次结构。组合模式使得用户对单个对象和组合对象的访问具有一致性，即能让客户以一致的方式处理个别对象以及对象组合。它创建了对象组的**树形结构**，将对象**组合成树形结构**以表示『整体-部分』的层次关系
2. 类图：

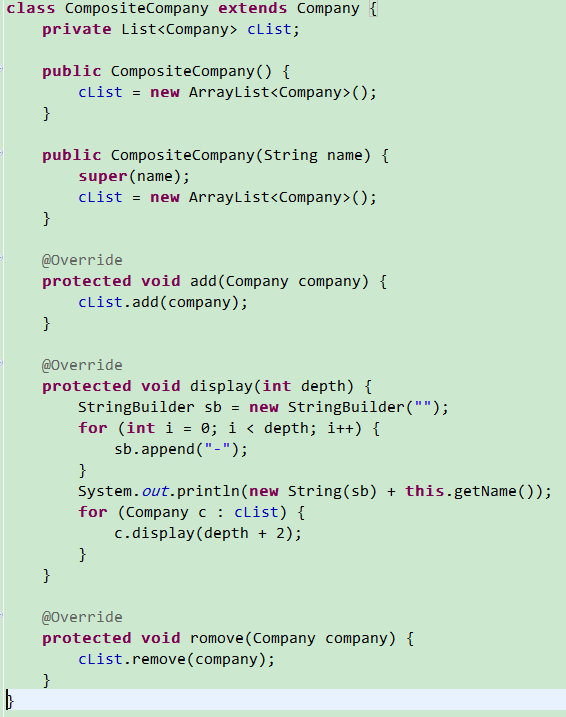


* 1. Component：对象声明接口，在适当情况下，实现所有类共有的接口默认行为，用于访问和管理Component 子部件，Component可以是抽象类或者接口
  2. Leaf：叶子节点，叶子节点没有子节点
  3. Composite：非叶子节点，用于存储子部件，在Component接口中实现子部件的相关操作，比如add、remove等

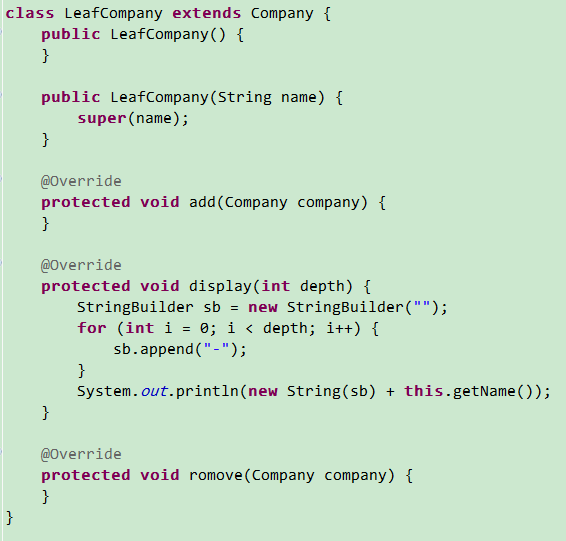
1. 实例

抽象公司接口，树枝节点与叶子节点均继承自该类，处理两类节点时均统一当做该类节点对待。  


树枝节点



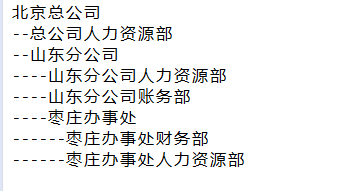
叶子节点(对于叶子节点来说，不需要子节点，当然也就不需要添加和删除子节点的方法)



测试:



输出：



1. 开发中的应用场景
   1. Junit中的Test(抽象)、TestSuite(容器) 、TestCase(叶子)
   2. OS中的资源管理器
   3. OA系统中组织结构的处理
2. 组合模式注意事项
   1. 简化客户端操作。客户端只需要面对一致的对象而不用考虑整体部分或者节点叶子的问题
   2. 方便创建出复杂的层次结构。客户端不用理会组合里面的组成细节，容易添加节点或者叶子从而创建出复杂的树形结构
   3. 需要**遍历组织机构**，或者处理的对象**具有树形结构时**, 非常适合使用组合模式
   4. 要求较高的抽象性，如果节点和叶子有很多差异性的话，比如很多方法和属性都不一样，不适合使用组合模式