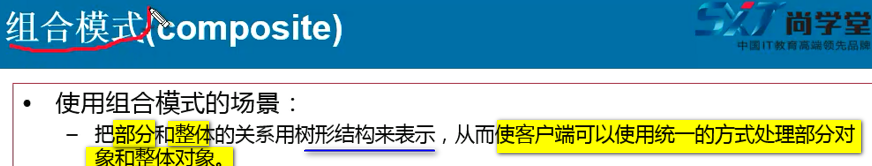
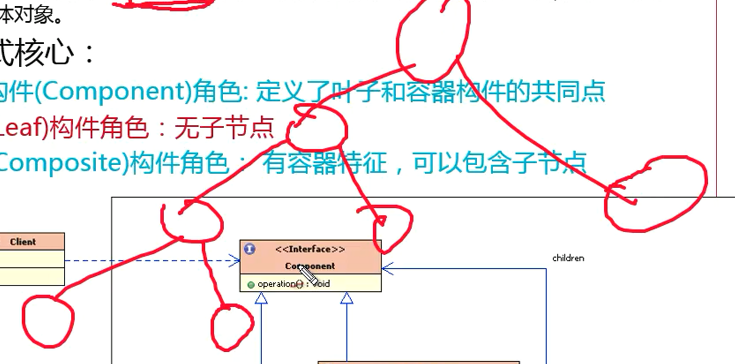
组合模式 Composite



继承 组合 不要和这个组合模式搞混淆了

碰到树形结构 一般都要用到组合模式

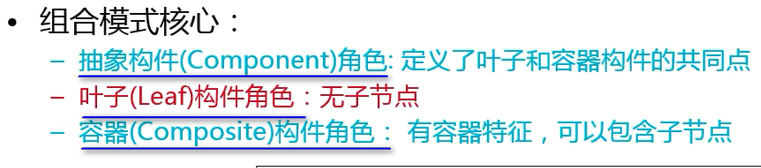
一个数据结构

 在开发中非常常见

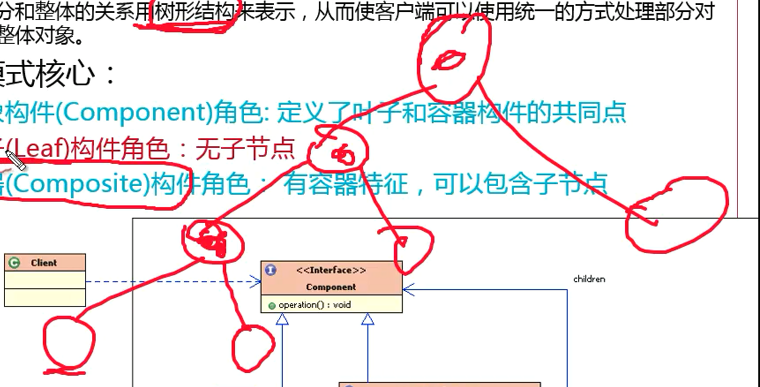
XML文件解析 树形结构

资源管理器 树形结构

树形结构里面 树叶节点 leaf

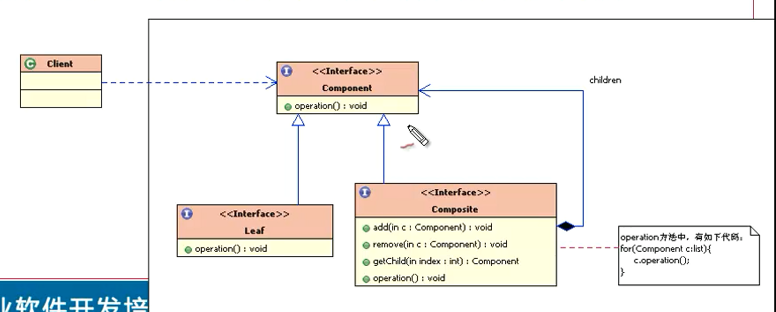


容器在这里面叫Composite



操作里面的时候 对里面的节点 无论哪一种节点 有统一的处理方式

调用某些API的时候 处理容器构件和叶子节点是一样的



【composite就是组合的意思 composite组合了Component和children的东西】

Component就是接口 抽象的

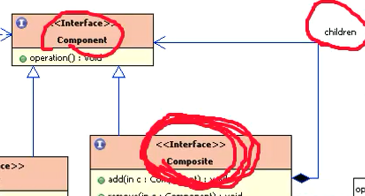
Leaf和Composite都是Component的子接口

Composite接口中 有三个自己的方法 就是对被包含的Component进行CRUD

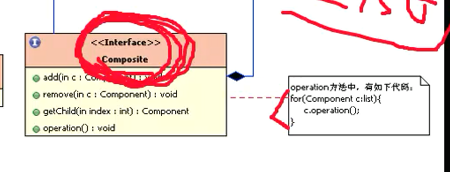
使用代码实现一下



现在增加刚才说的方法

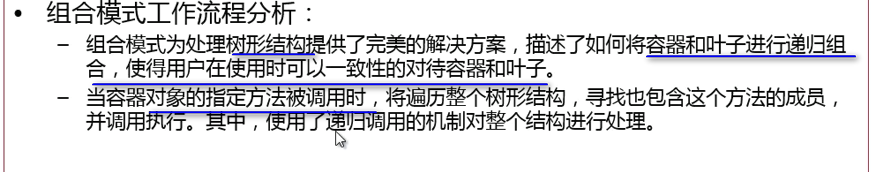
 这个children实际上是一个List

现在操作这个容器的时候



这是这几个类之间的关系

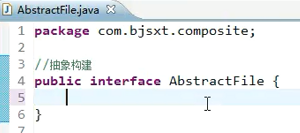
组合模式定义的时候 有一个天然的递归结构

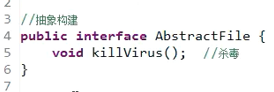


构建一个杀毒软件的框架

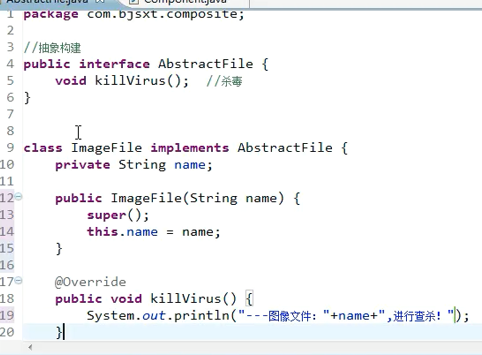
模拟定义好的接口

杀毒 就是对文件和目录的操作

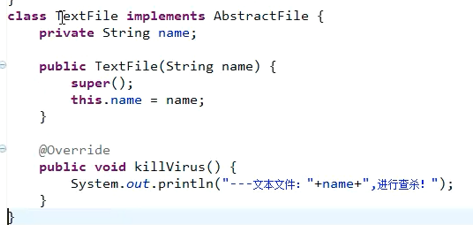
 ----- - 这个相当于Component



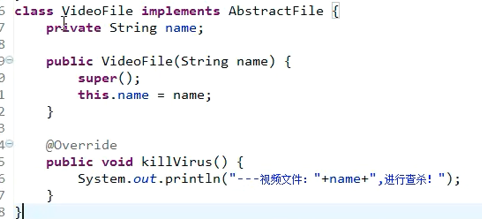
查杀图片文件



现在杀文本文件



甚至可以杀视频文件



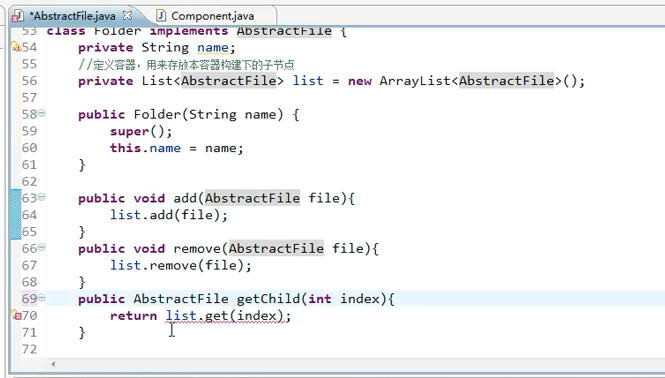
这几个都是leaf

图像文件不能包含别的了 所以是leaf

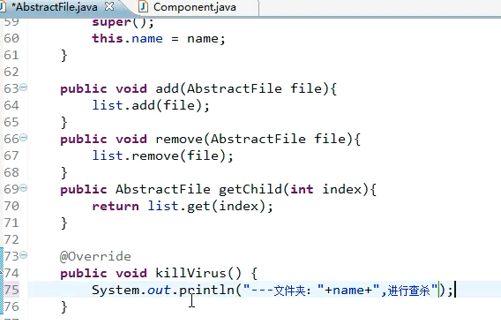
文件夹显然是容器组件

首先实现了抽象组件的方法

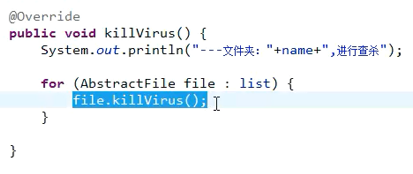
这个容器就来从我们容器构建的子节点来存储我们的children



还有一个接口 杀毒

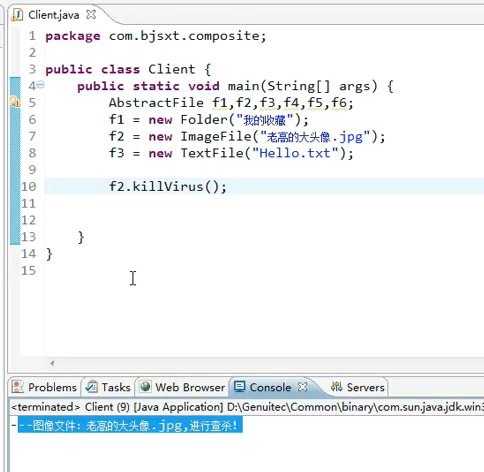


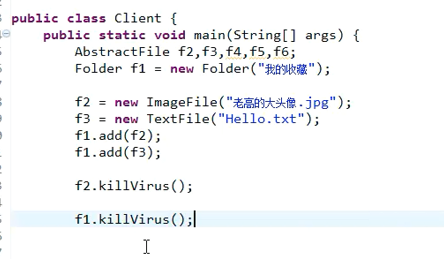
这个文件夹里面还有子容器 遍历一下



这就有天然的递归 ---- 如果这里面是List中含有文件夹 就会递归

组合模式优势 --- 可以处理容器节点和叶子节点的方式是一样的



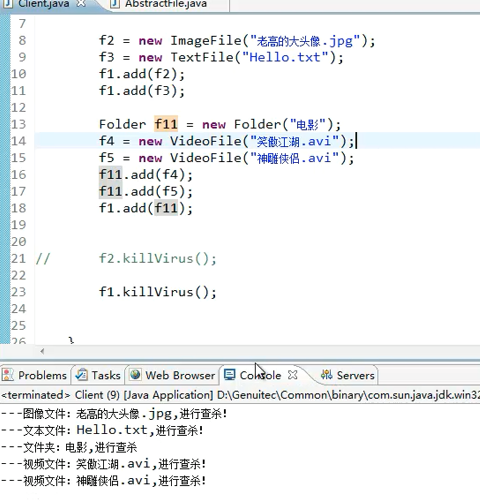


f1是文件夹 和 f2文件都是一样的东西调用 都是直接调用killVirus 调用方式都是一样的 一样的接口调用

某个节点 或者 一个节点 处理方式一样 因为继承了相同的父类



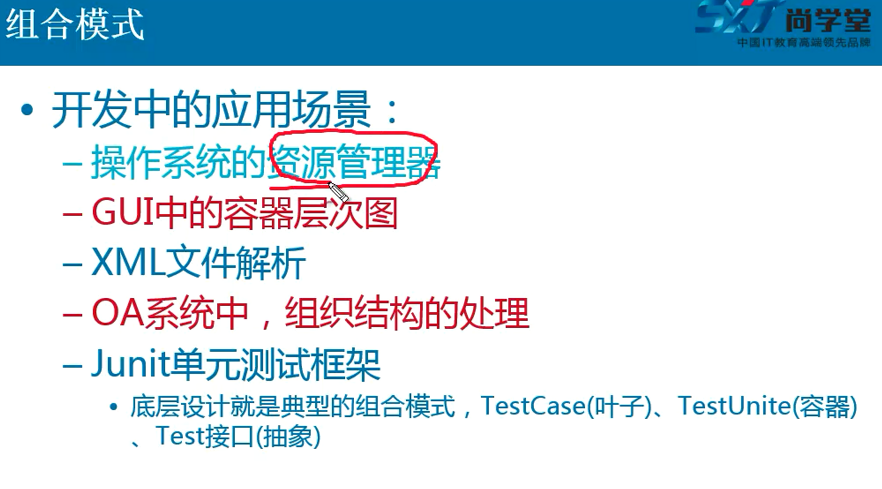
这样 直接杀f1



Z组合模式 对部分对象和整体对象都能统一处理

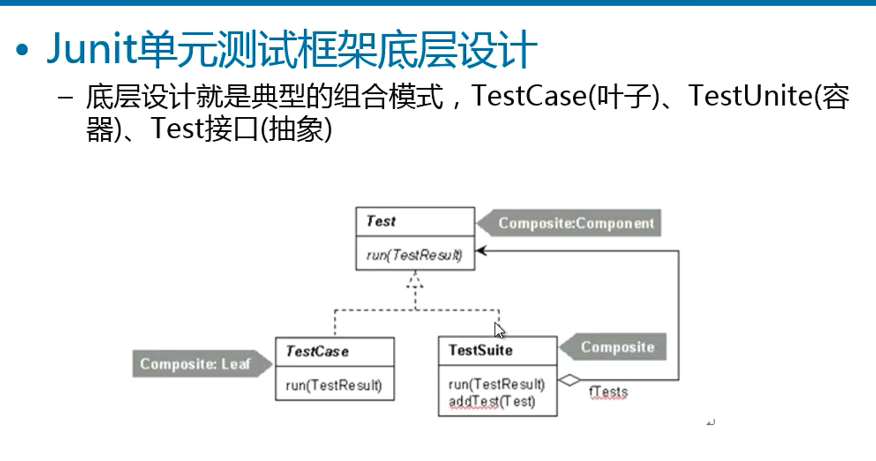
以后碰到树形数据结构 就用组合模式

很多开源组件都用组合模式



这些特点都是典型的树状结构

Junit里面有三个内容 是典型的树状结构 TestCase等



后面的JUnit讲到了 会说