visitor模式概念visitor模式进一步 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/144/2021_2022_visitor_E6_A8_A1_c104_144722.htm 一，访问者模式的角色：抽象访问者：声明一个或者多个访问操作，形成所有的具体元素都要实现的接口具体访问者：实现抽象访问者所声明的接口抽象节点：声明一个接受操作，接受一个访问者对象作为参量具体节点：实现了抽象元素所规定的接受操作结构对象：遍历结构中的所有元素，类似List Set等二，在什么情况下应当使用访问者模式访问者模式应该用在被访问类结构比较稳定的时候，换言之系统很少出现增加新节点的情况。因为访问者模式对开－闭原则的支持并不好，访问者模式允许在节点中加入方法，是倾斜的开闭原则，类似抽象工厂。三，访问者模式的缺点：1，增加节点困难2，破坏了封装因为访问者模式的缺点和复杂性，很多设计师反对使用访问者模式。个人感觉应该在了解的情况下考虑衡量选择.静态分派，动态分派，多分派，单分派 ---------- visitor模式准备 一，静态分派:1，定义：发生在编译时期，分派根据静态类型信息发生，重载就是静态分派2，什么是静态类型：变量被声明时的类型是静态类型 什么是动态类型：变量所引用的对象的真实类型3，有两个类,BlackCat ,WhiteCat都继承自Cat如下调用class Cat{}class WhiteCat extends Cat{}class BlackCat extends Cat{}public class Person { public void feed(Cat cat){ System.out.println("feed cat"). } public void feed(WhiteCat cat){ System.out.println("feed WhiteCat"). } public void feed(BlackCat cat){ System.out.println("feed BlackCat"). } public static void

main(String[] args) { Cat wc = new WhiteCat(). Cat bc = new BlackCat(). Person p = new Person(). p.feed(wc). p.feed(bc). }}运行结果是:feed catfeed cat这样的结果是因为重载是静态分派，在编译器执行的，取决于变量的声明类型，因为wc ,bc都是Cat所以调用的都是feed(Cat cat)的函数.二，动态分派1，定义：发生在运行期，动态分派，动态的置换掉某个方法。还是上边类似的例子：class Cat{ public void eat(){ System.out.println("cat eat"). }}public class BlackCat extends Cat{ public void eat(){ System.out.println("black cat eat"). } public static void main(String[] args){ Cat cat = new BlackCat(). cat.eat(). }}这个时候的结果是:black cat eat这样的结果是因为在执行期发生了向下转型，就是动态分派了。三，单分派：1，定义：根据一个宗量的类型进行方法的选择四,多分派：1，定义：根据多于一个宗量的类型对方法的选择2，说明：多分派其实是一系列的单分派组成的，区别的地方就是这些但分派不能分割。3,C ,Java都是动态单分派，静态多分派语言多分派的语言有：CLOS Cecil访问差异类型的集合类--visitor模式入门访问差异类型的集合类--visitor模式入门本文对应代码下载这里一，问题提出访问同一类型的集合类是我们最常见的事情了，我们工作中这样的代码太常见了。1 Iterator ie = list.iterator().2 while (ie.hasNext()) {3 Person p = (Person)ie.next().4 p.doWork().5 } 这种访问的特点是集合类中的对象是同一类对象Person，他们拥有功能的方法run,我们调用的恰好是这个共同的方法。在大部份的情况下，这个是可以的，但在一些复杂的情况，如被访问者的继承结构复杂，被访问者的并不是同一类对象，也就是说不是继承自同一个根类。方法名也并

不相同。例如Java GUI中的事件就是一个例子。例如这样的问题，有如下类和方法:类：PA ,方法：runPA().类：PB ,方法：runPB().类：PC ,方法：runPC().类：PD ,方法：runPD().类：PE ,方法：runPE().有一个集合类ListList list = new ArrayList().list.add(new PA()).list.add(new PB()).list.add(new PC()).list.add(new PD()).list.add(new PE()).....