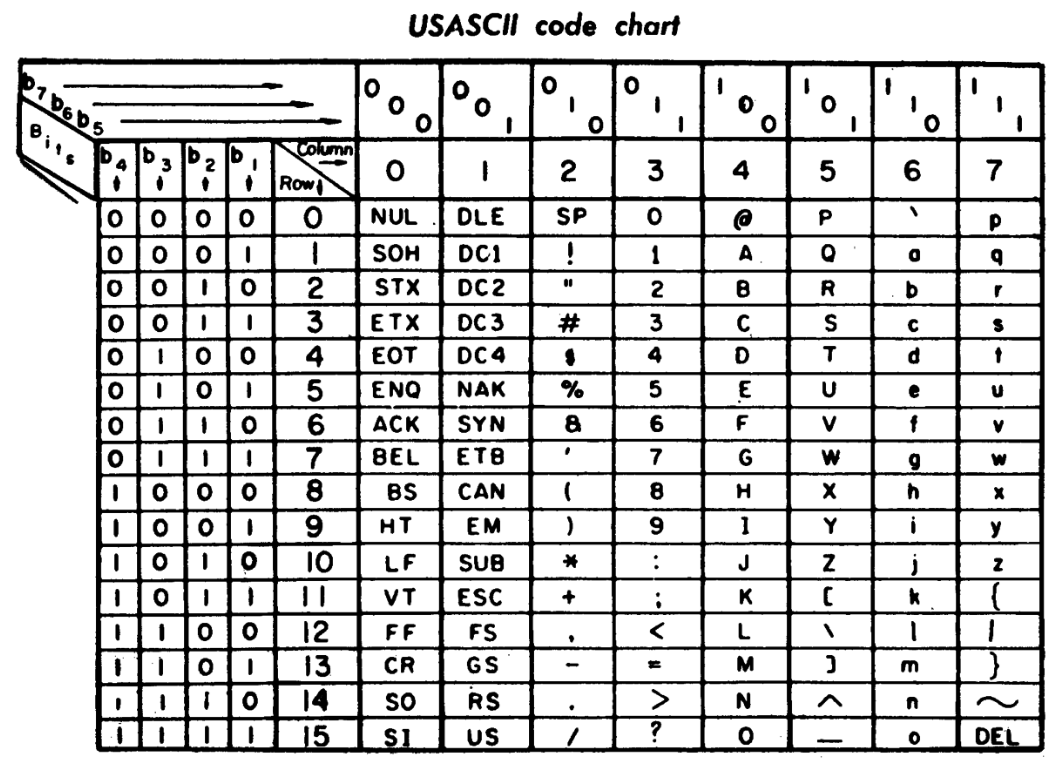
--HW5 for FoPL

--Stu:金泽文 No:PB15111604

**1.**

**Step 0**：

鉴于书上有对Unsovability of halting problem on string input的证明，为了解决Unsovability of halting problem on int input，可以考虑将int 与string联系起来。可以通过一下构造方式，将int与string一一对应起来：

利用1968年版的ASCII码表，将每一个字符与对应的7位二进制数一一对应起来。并且为了标记program结束标志EOF（在这里，是string字符串的‘\0’），我们令int对应的小数点为EOF位。这样，每一个int INT都对应一个二进制数BIN，二者的集合互为双射，而这个BIN在7位对齐之后又可以与单个字符一一对应，二者的集合再次构成双射，结合小数点与EOF的对应关系，我们就能得到string集合与int集合的双射关系。所以，只需要证明Unsovability of halting problem on string input，就可以根据对应关系证明Unsovability of halting problem on int input，也就是本题要证明的结论。下面证明Unsovability of halting problem on string input（跟书上证明一致）：

**Step 1：**

假设存在读入两个字符串P，x的程序Q，满足：

这里，Q是一个总会halt的程序

**Step 2：**

利用Q，构造读入1个字符串P的程序D，满足：

这里，D是一个会根据情况可能halt，可能never halt 的程序。

于是，我们得到：

**Step 3：**

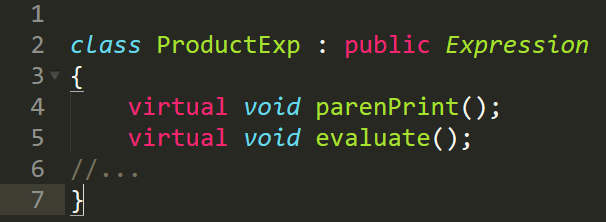
所以，

显然矛盾。

**Step 4：**

综上，Unsovability of halting problem on string input得证，根据Step 1， Unsovability of halting problem on int input得证。证毕。

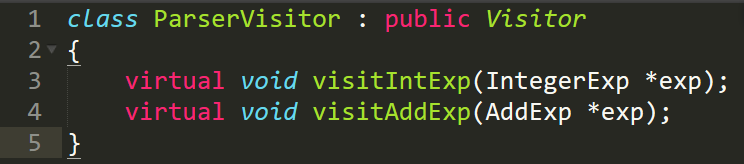
**2.**

（a）1个。形如：

（b）n+1个。每个subclass和本身都需要添加。

（c）m+2个。需要添加一个新的类ProductExp，

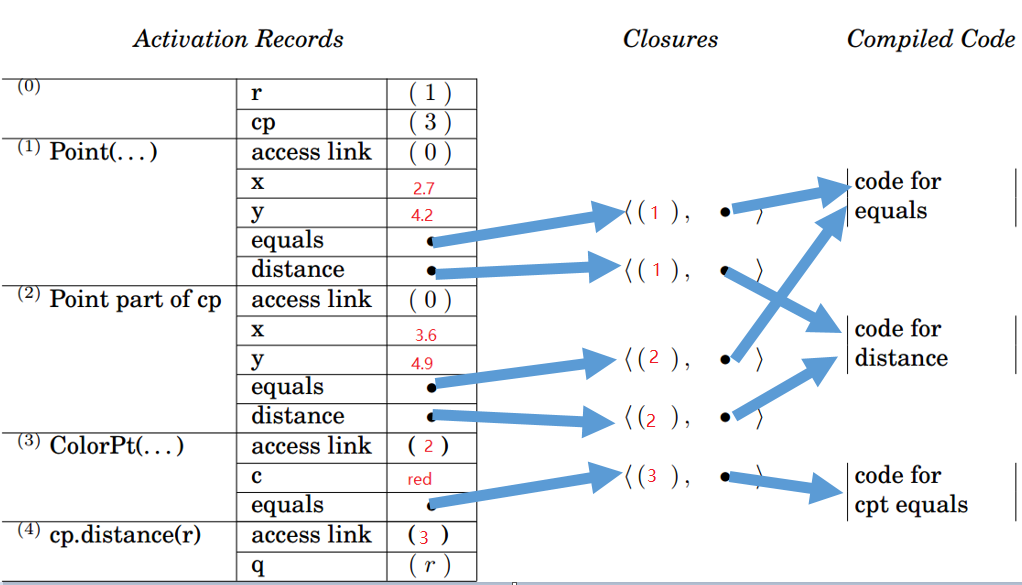
同时在Visitor类及其所有子类中添加方法。

（d）只需要一个Visitor的子类。

（e）在对类的操作比较少，设计开发时对类中method的增减比较少时。

（f）在对类的操作比较多，设计开发时对类中method的增减比较多时。

**3.**

（a）

（b）

沿着cp.distance(r)对应的access link 找到对应的（2）Point part of cp，这里的x就是所求，value为3.6的int。

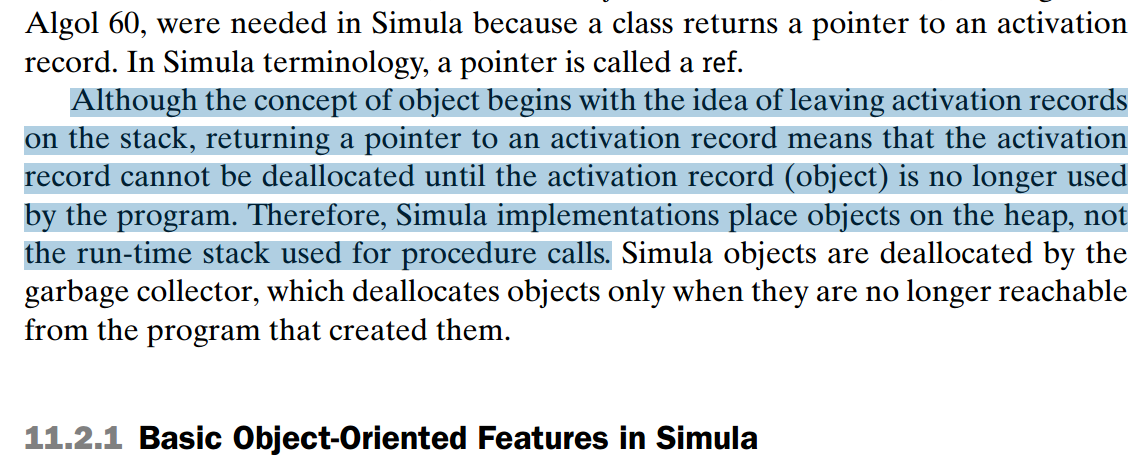
（c）

首先沿着access link找到（3）ColorPt，在这里没有找到x变量，于是继续沿着access link 找到（2）Point part of cp，这里的x就是所求，value为3.6的int。

（d）

因为这一步的distance方法定义在Point部分，而不是在ColorPt部分。先在Point part of the cp里寻找x和y变量。找到，则结束寻找，而无需寻找ColorPt部分。

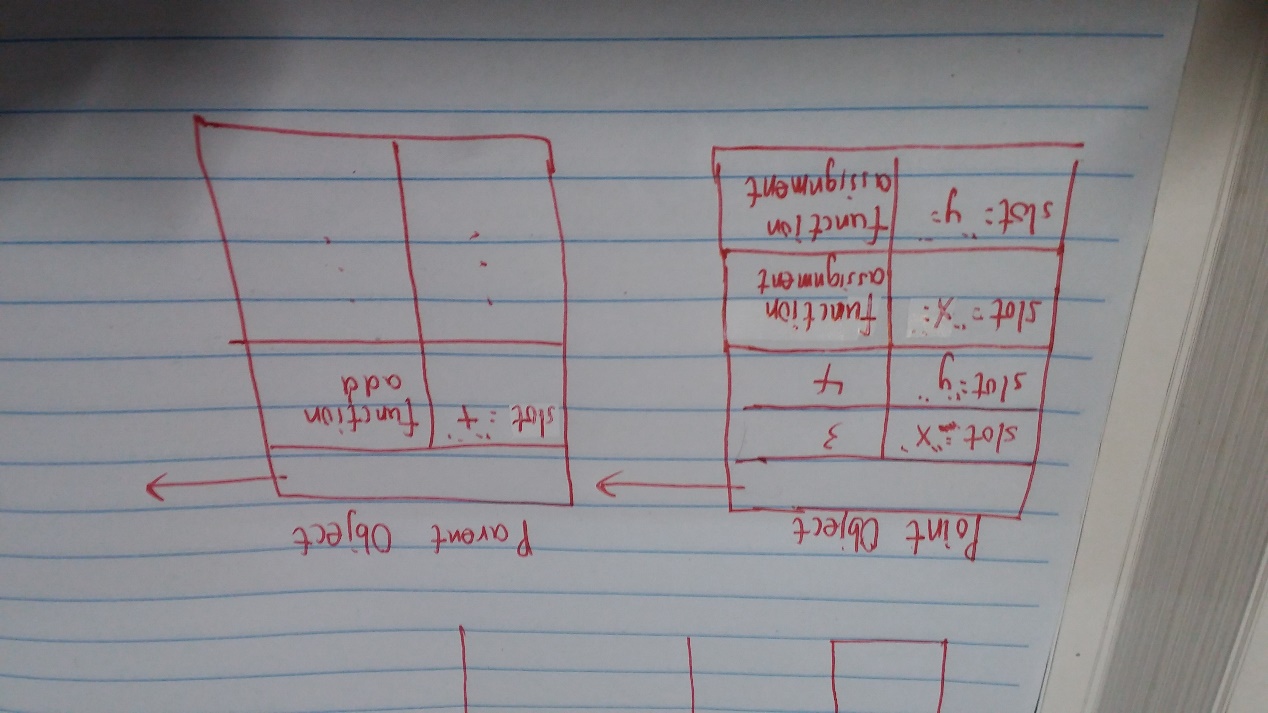
（e）

这一问题的解答可以在书上找到：

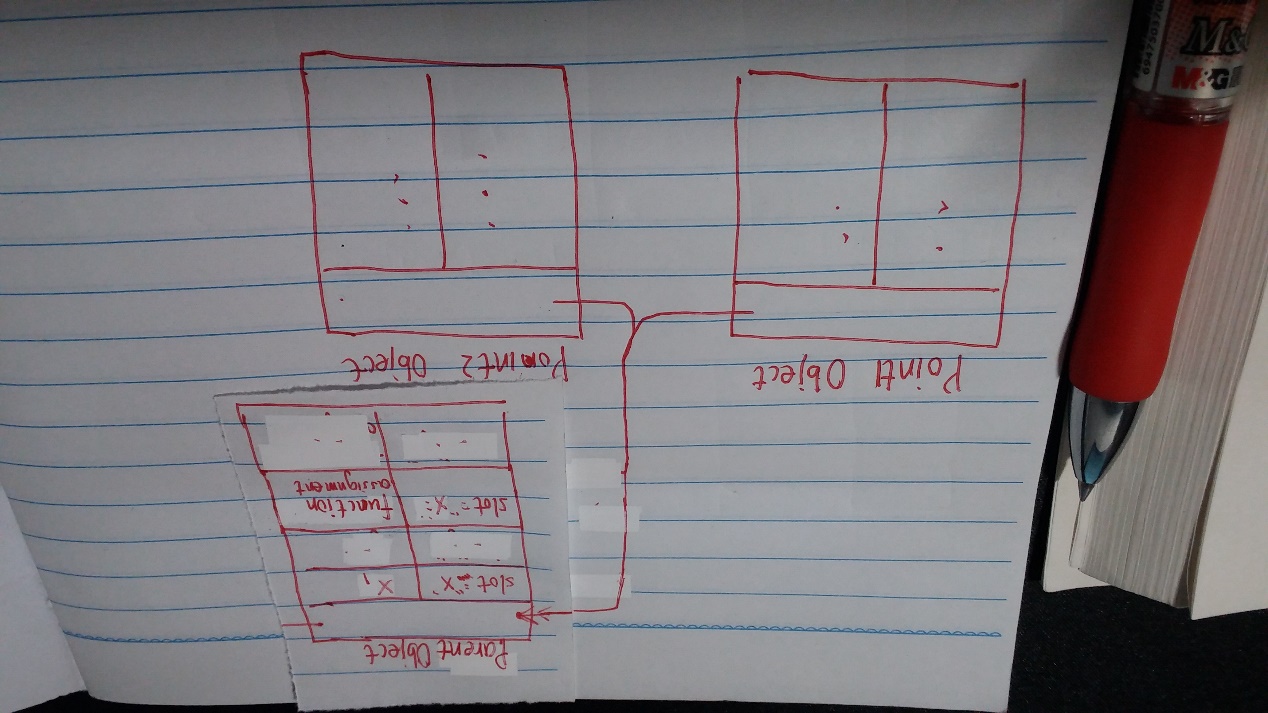
也就是说，由于我们要根据object是否再使用来判断是否deallocate活动记录，所以我们实现的时候应该把object一律放在堆上，而不是运行时栈上。

**4.**

（a）



（b）



（c）

先创建一个对象PointA，使得PointA中含有x: ,y: ,move。

再创建PointA的子对象PointB，它含有x, y， 其parent pointer指向PointA。

然后创建对象PointC，copy自PointB。

这样PointA有4条记录，PointB有3条记录，PointC有3条记录，所以一共10条记录。

（d）

因为parent pointer可以被修改，因此不能在compile time就进行所有的类型检查，故性能会有所影响。

**5.**

（a）

二者都不需要。因为调用的时候直接查找vtable就可以找到对应地址。

（b）

2n个。使用多继承要考虑offset域。

（c）

第一种编译器在调用虚函数的时候，还要计算this+offset。

（d）

一方面，条目数目没有变。另一方面，由于多继承的时候要调用thunk，所以所用时间变了。

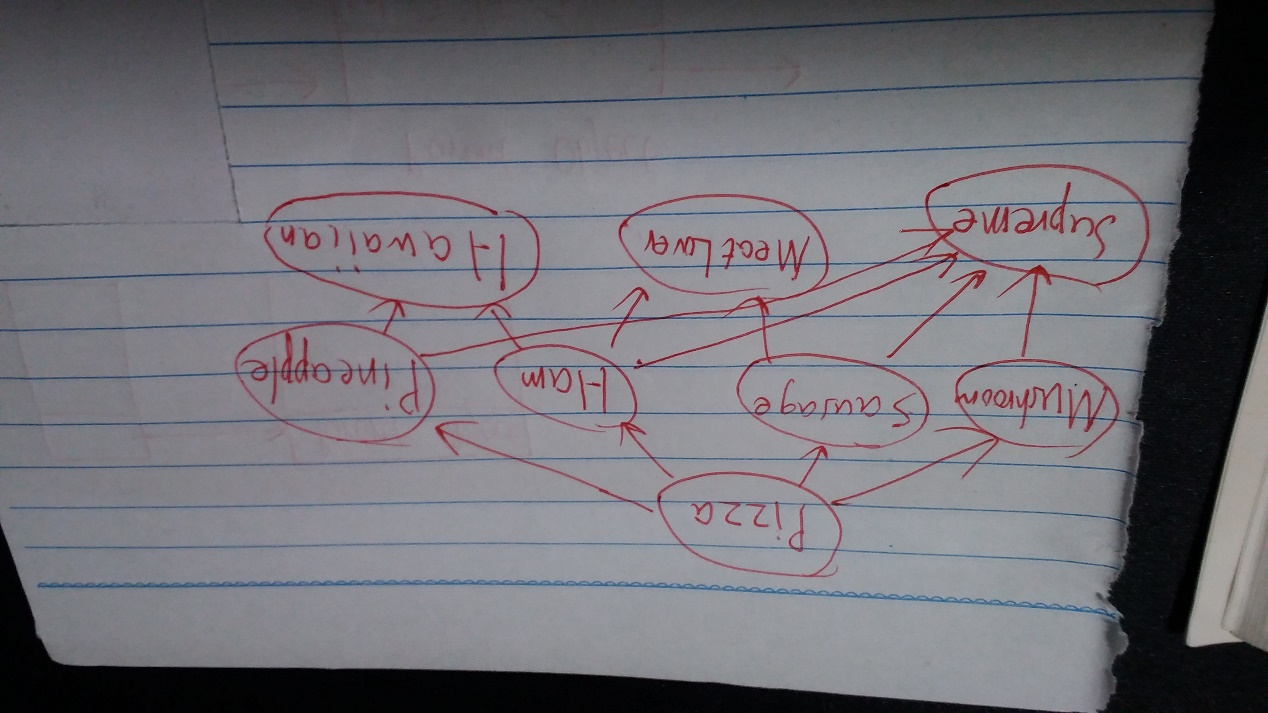
（e）

如果第二个编译器使用offsets，那就不能link，因为二者所对应的vtable格式不一致。

而如果第二个编译器使用thunks，那就能够link，因为二者所对应的vtable格式是一致的。

**6.**

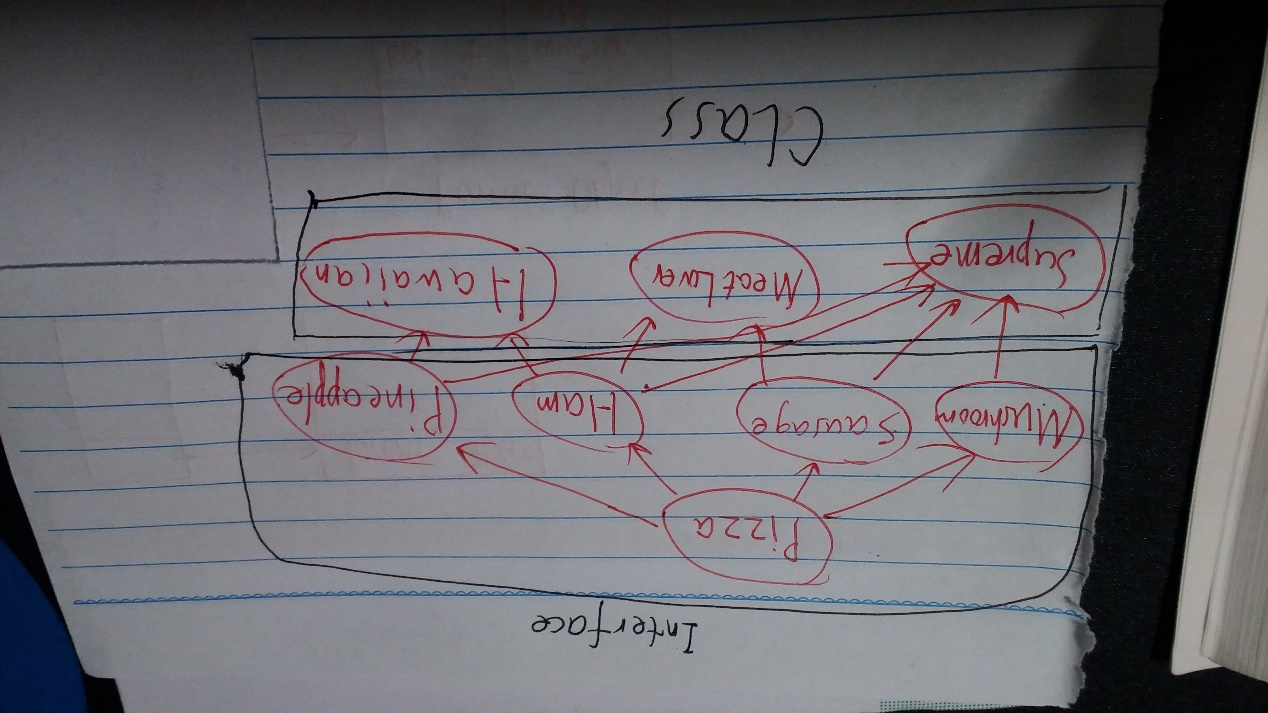
（a）



（b）

Sausage和Ham都是Pizza的subclass，但同时它们又都是MeatLover的superclass，这就导致了Diamond Inheritance，对这种状况下的继承的处理也就相当复杂。并且，Pizza, Ham, Pineapple, Hawaiian四者也构成了Diamond inheritance。

（c）



（d）

C++的类可以多继承，并且无需查表，所以编写代码十分方便同时程序效率也会很高，而且子类不用重新实现父类的方法，但是编译器难以处理Diamond inheritance的问题。

Java只能通过interface实现多继承，这样写比较清晰，避免了Diamond inheritance问题，但是需要写较多的重复代码。

**感谢助教的认真批改！**