--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

USASCII code chart

b_7 b_6 b_5 → / Bits ↓					0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b_4	b_3	b_2	b_1	Row ↓ Column →	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	|
1	1	0	1	13	CR	GS	-	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

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：

$$Q(P,x)=\begin{cases}t, & P(x)\ \text{halts}\\ f, & P(x)\ \text{never halts}\end{cases}$$

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：

$$D(P)=\begin{cases}\text{never halts}, & Q(P,P)==t\\ \text{halts}, & Q(P,P)==f\end{cases}$$

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：

$$D(P)=\begin{cases}\text{never halts}, & P(P)\ \text{halts}\\ \text{halts}, & P(P)\ \text{never halts}\end{cases}$$

Step 3：

所以，

$$D(D)=\begin{cases}\text{never halts}, & D(D)\ \text{halts}\\ \text{halts}, & D(D)\ \text{never halts}\end{cases}$$

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：

```
1
2  class ProductExp : public Expression
3  {
4      virtual void parenPrint();
5      virtual void evaluate();
6  //...
7  }
```

（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，

```
1  virtual void visitProductExp(ProductExp *exp)
```

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。

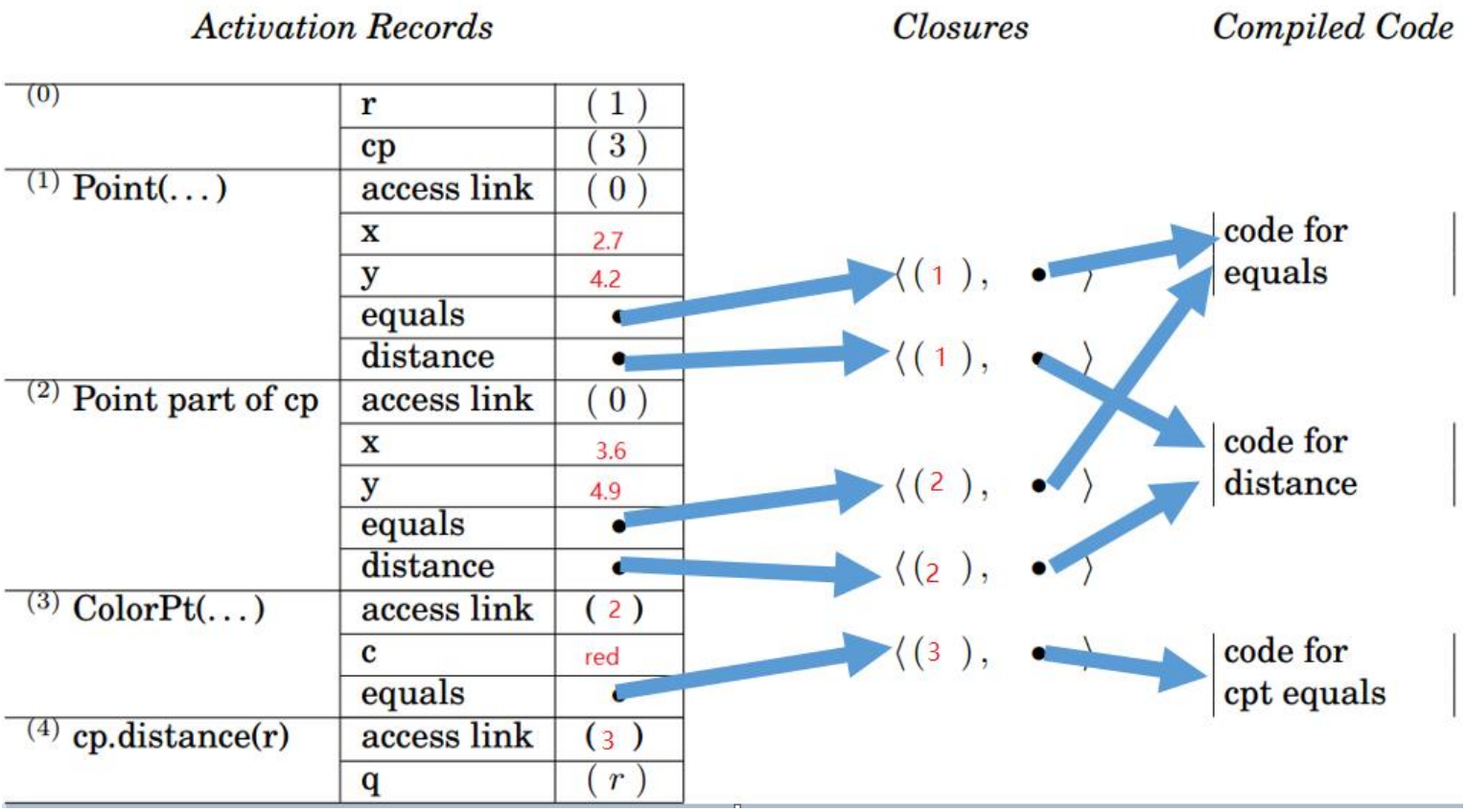
```
1  class ParserVisitor : public Visitor
2  {
3      virtual void visitIntExp(IntegerExp *exp);
4      virtual void visitAddExp(AddExp *exp);
5  }
```

（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：

Algol 60, were needed in Simula because a class returns a pointer to an activation record. In Simula terminology, a pointer is called a ref.

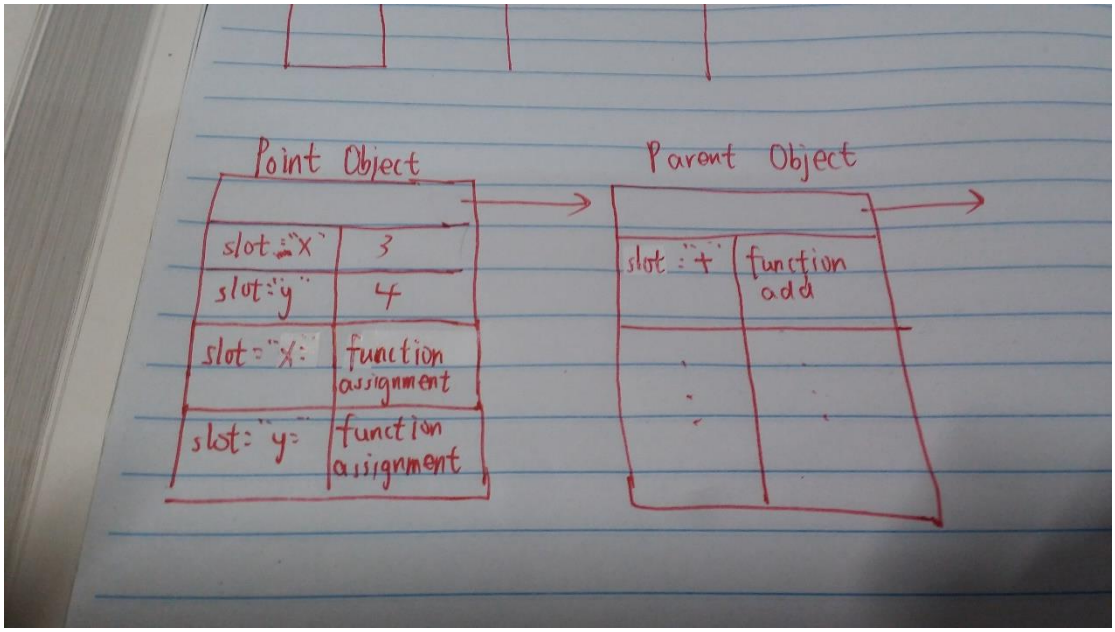
Although the concept of object begins with the idea of leaving activation records on the stack, returning a pointer to an activation record means that the activation record cannot be deallocated until the activation record (object) is no longer used by the program. Therefore, Simula implementations place objects on the heap, not the run-time stack used for procedure calls. Simula objects are deallocated by the garbage collector, which deallocates objects only when they are no longer reachable from the program that created them.

11.2.1 Basic Object-Oriented Features in Simula

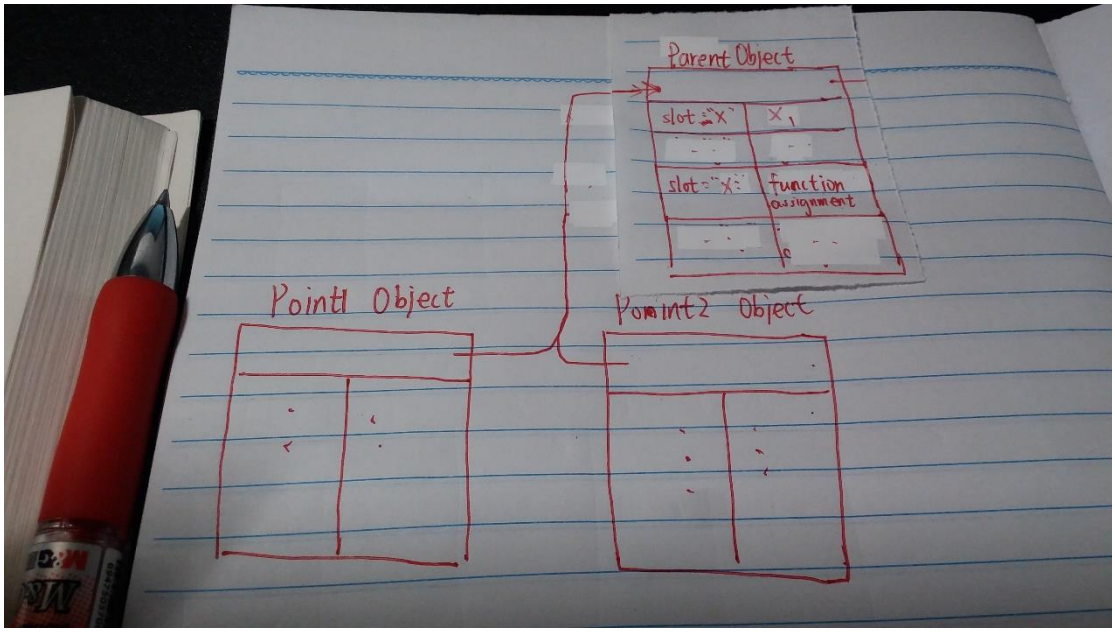
也就是说，由于我们要根据 object 是否再使用来判断是否 deallocate 活动记录，所以我们实现的时候应该把 object 一律放在堆上，而不是运行时栈上。

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。

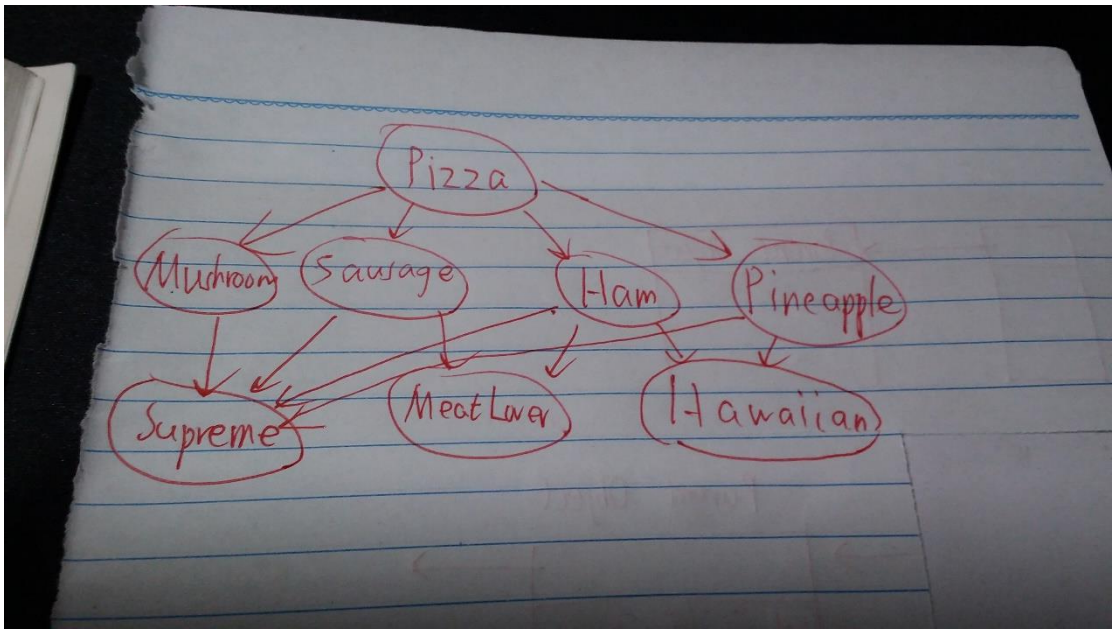
（e）

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.

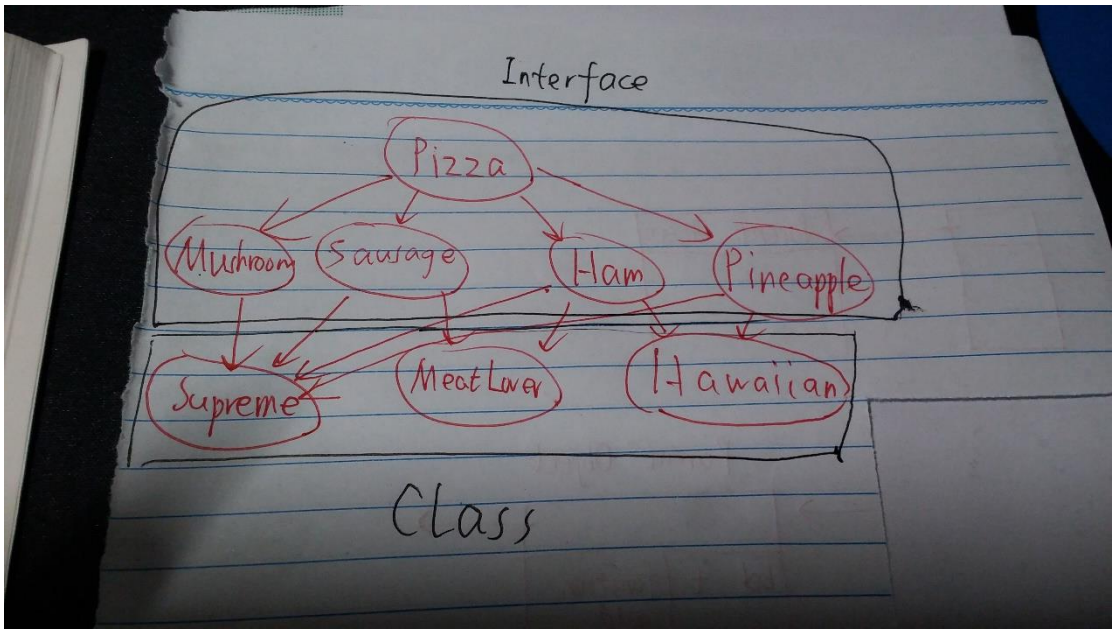
（a）



（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！