Multiple Inheritance

- Consider the following ~~class~~ record types.

```
type Object = {age: int}
type Vehicle = {age: int, speed: int}
type Machine = {age: int, fuel: String}
type Car = {age: int, speed: int, fuel: String}
```

- Then we have the following subtype relations.

  $\text{Vehicle} \sqsubseteq \text{Object}$

  $\text{Machine} \sqsubseteq \text{Object}$

  $\text{Car} \sqsubseteq \text{Object}, \text{Car} \sqsubseteq \text{Vehicle}, \text{Car} \sqsubseteq \text{Machine}$

- Can extend this to full-fledged classes with code members.
- If the same method is implemented in multiple superclasses, then we need a linguistic mechanism to specify or disambiguate which methods are inherited from which superclass.
- Q: Since (by the definition of $\sqsubseteq$) a `Car` can be used wherever a `Vehicle` is expected, and also wherever a `Machine` is expected, how do we lay out the object record of `Car`?

Step 0: Single Base Class

- Let’s say we have the following classes (simple C++ syntax).

```
class A {int a; void f(int);};
class B : A {int b; void g(int);};
class C : B {int c; void h(int);};
```

- Then the layout of an object of class `C` looks like this.

- We also have the declaration

```
C* pc;
```

- How is the call `pc->g(2)` implemented?

Step 1: Single Base Class, Virtual Functions

- Let’s say we have the following classes.

```
class A {int a;
         virtual void f(int);
         virtual void g(int);
         virtual void h(int)};
class B : A {int b; void g(int);};
class C : B {int c; void h(int);};
```

- Then the layout of an object of class C looks like this.

- We also have the declaration

```
C* pc;
```

- How is the call `pc->g(2)` implemented?

Step 2: Multiple Base Classes

- Let’s say we have the following classes.

```
class A {int a; void f(int);};
class B {int b; void g(int);};
class C : A, B {int c; void h(int);};
```

- Then the layout of an object of class C looks like this.

- We also have the declaration

```
C* pc;
```

- How is the call `pc->f(2)` implemented?
- How about the call `pc->h(2)`?
- How about the call `pc->g(2)`?

Step 3: Multiple Base Classes, Virtual Functions

- Let’s say we have the following classes.

```
class A {virtual void f(int);};
class B {virtual void f(int);
         virtual void g(int);};
class C : A, B {void f(int);};
```

- Then the layout of an object of class C looks like this.

- We also have the declarations

```
A* pa = new C; B* pb = new C; C*pc = new C;
```

- How is the call `pa->f(2)` or `pc->f(2)` implemented?
- How about the call `pb->f(2)`?