CSc 553

Principles of Compilation

17 : OO Languages — Multiple Inheritance

Department of Computer Science
University of Arizona

collberg@gmail.com

Copyright © 2011 Christian Collberg

Multiple Inheritance

Multiple Inheritance

- In some languages (C++, Eiffel) a class can have more than one superclass.

```
class Person { Name : STRING; }
class Student extends Person {
   Advisor : Teacher;}
class Teacher extends Person {
   Salary : INTEGER;
   method Rich () : BOOLEAN;
      return Salary > 50000;}
class Tutor extends Student, Teacher {
   Boss : Teacher;}
```

Multiple Inheritance...

```
class Teacher extends Person {
   Salary : INTEGER;
   method Rich () : BOOLEAN;
      return Salary > 50000;}
```

`Rich()` should translate into:

```
PROCEDURE Rich (
   SELF : Teacher) : BOOLEAN;
   RETURN SELF^.Salary > 50000;
```

Multiple Inheritance...

- We'd like to be able to call `m.Rich()` for any `Teacher` object, including a `Tutor`:

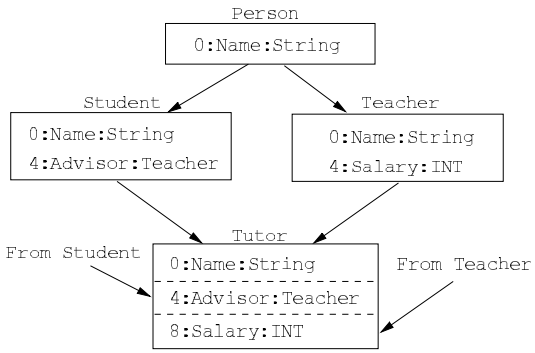
```
PROCEDURE Rich (
   SELF : Teacher) :  BOOLEAN;
   RETURN SELF^.Salary > 50000;

Teacher Knuth = new Teacher;
Tutor Lucy = new Tutor;
boolean k = Knuth.Rich()
boolean l = Lucy.Rich()
```

- In order for this to work, the `Salary` field in a `Tutor` record must be at the same offset as the `Salary` field in the `Teacher` record.

Multiple Inheritance...

- But, if our record layout uses simple concatenation of parent classes (like with single inheritance), we get:



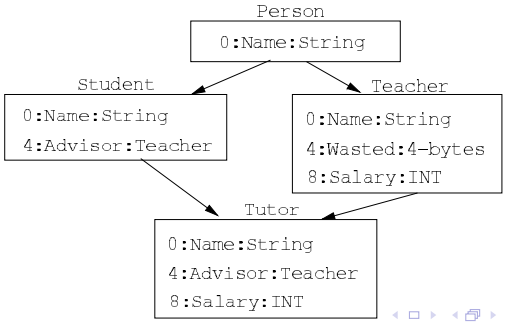
- The `Salary` field in a `Teacher` record is at offset 4, but the `Salary` field in the `Tutor` record is at offset 8.

Multiple Inheritance...

- An inefficient implementation might do:

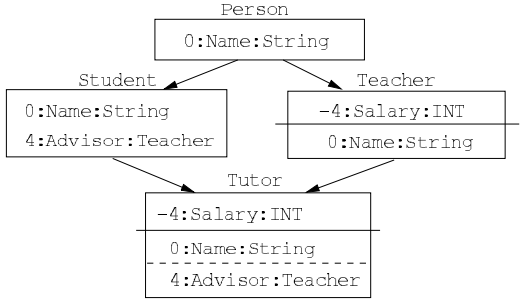
```
PROCEDURE Rich (SELF : Teacher) :  BOOLEAN;
   RETURN IF ISTYPE(SELF,Teacher)
      THEN (SELF-4)^>50000 ELSE (SELF+8)^>50000;
```

- Or we could insert extra space to align the fields properly:

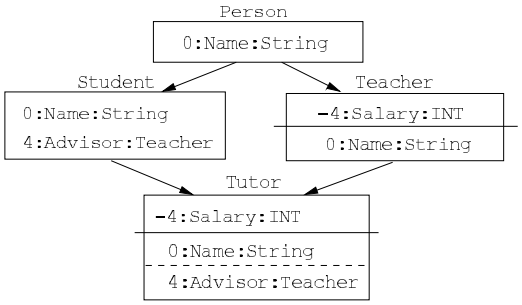


Multiple Inheritance...

- With *multi-directional* layouts, we place variables at both positive and negative offsets:



Multiple Inheritance...



- The Salary-field is always at the same offset, regardless of what type of object:

```
PROCEDURE Rich (
   SELF : Teacher) :  BOOLEAN;
   RETURN (SELF-4)^>50000;
```

Multiple Inheritance...

- How does the language deal with the same field inherited through more than one path? A Tutor inherits Name twice, once from Student and once from Teacher:

```
class Person { Name :  STRING; }
class Student extends Person {···}
class Teacher extends Person {···}
class Tutor extends Student,Teacher {···}
```

- Should Tutor have one or two copies of Name?
- In Trellis/Owl you always get just one copy of Name.
- In C++ you can choose. If you declare a superclass *virtual*, Tutor only gets one copy of Name, otherwise two.

Multiple Inheritance...

- How does the language deal with different fields/methods with the same type/signature inherited from different classes?

```
class Student {Name :  STRING; ··· }
class Teacher {Name :  STRING; ··· }
class Tutor extends Student,Teacher {···}
Tutor T = new Tutor();
T.Name = "Knuth"; /* Which Name?  */
```

Multiple Inheritance...

```
class Student {Name :  STRING; ··· }
class Teacher {Name :  STRING; ··· }
class Tutor extends Student,Teacher {···}
Tutor T = new Tutor();
T.Name = "Knuth"; /* Which Name?  */
```

- In Eiffel, the programmer has to rename fields until there are no more conflicts, using a **rename** clause:

```
class Tutor extends Student,
   Teacher rename Name⇒TName {···}
```

- In C++, conflicts are resolved when the field/method is used:

```
Tutor T = new Tutor();
Teacher::T.Name = "Knuth";
```

Readings and References

- Read Scott: 146-CD-157-CD.
- For information on constructing layouts for multiple inheritance, see
  - William Pugh and Grant Weddell: "Two-directional record layout for multiple inheritance."