1 Multiple Inheritance

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

```pascal
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;
}
```

2 Multiple Inheritance...

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

Rich() should translate into:

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

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

```pascal
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.

4 Multiple Inheritance...

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

![Record Layout Diagram]

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

5 Multiple Inheritance...

- An inefficient implementation might do:

```pascal
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:
6 Multiple Inheritance...

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

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

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

7 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.

9 Multiple Inheritance...

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

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

10 Multiple Inheritance...

```java
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:

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

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

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