Multiple Inheritance

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

```java
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...

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

Rich() should translate into:

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

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

```java
PROCEDURE Rich (SELF : Teacher) : BOOLEAN; 
RETURN SELFˆ.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.

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:

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:

```java
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.

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

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? */
```

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

```eiffel
class Tutor extends Student,
    Teacher rename Name => TName {}
```

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

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