CS 2704

Topic:
Multiple Inheritance

Multiple Inheritance

class TA : public Student, public Employee {
    ...
};

Objects and Multiple Inheritance

Student Data

Employee Data

Teaching Assistant Data

A Problem with Data

• Don’t forget rest of inheritance hierarchy

• A TA could have two names!

C++ Solution

• Change declarations of base classes
  class Student : virtual public Person {...};
  class Employee : virtual public Person {...};
• If inherit both
  class TA : public Student, public Employee {...};
• TA object contains pointers to Person object(s)

Virtual Inheritance

Hypothetical data layout - compiler may do something else
Casting & Virtual Inheritance
- Ordinarily casting pointers “does nothing”
- The cast
  `static_cast<Person*>(studentp)`
  follows pointer
- The cast
  `dynamic_cast<Student*>(st_person)`
  works if virtual base class has virtual function (usually destructor)

Methods and Multiple Inheritance
- Both Parents have a move method

A Problem with Methods
- Can’t tell which base class to get method from
  `DrawOnText y; y.move(lowerleft);`
- Could be `TextBox::move(-)` or `Canvas::move(-)`
- Redefine methods with name clash

Method Dominance
For virtual inheritance
- `Person::id()` Returns SSN
- `Employee::id()`
- `Student::id()` Returns Student ID
- `TeachingAssistant::id()` (is Student::id)
  Ambiguous in non-virtual case

On Multiple Inheritance
- Much disagreement on whether multiple inheritance necessary
- Difficult to come up with examples that couldn’t be done equally well or better some other way
- Multiple inheritance increases complexity

When Multiple Inheritance?
- Disjoint base classes
  – No common base class
  – No method name clashes
- Want to enforce some protocol
  – Ex. MFC persistence functionality
- Conclusion: don’t go out of your way to use it, but recognize could be useful