

Polymorphism III

CS2704: Object-Oriented Software
Design and Construction

Constantinos Phanouriou
Department of Computer Science
Virginia Tech

CS2704 (Spring 2000)

1

Pure Virtual Methods

- A *pure virtual method* has a null definition
virtual void draw(Canvas&) = 0;
- *Abstract class* - class in which **at least one** method is pure virtual
 - Cannot be instantiated (no objects)
 - Can have fields and other methods
- *Pure abstract class* - defines an interface

CS2704 (Spring 2000)

2

Why Abstract Classes?

- Abstract class corresponds to general concept that is too general to actually have its own instances (must have derived class)
- Examples:
 - Geometric shape
 - Number
 - Mammal

CS2704 (Spring 2000)

3

Example: Geometric Shapes

- General class for geometric shapes

```
class Shape {  
public:  
    virtual void draw (Canvas&) const = 0;  
    virtual void print (ostream&) const = 0;  
    virtual void scale (Point center, double s);  
    virtual void move (int x, int y);  
};
```

- Better style to not include data

CS2704 (Spring 2000)

4

Protected Data

- If want data to be accessible to derived class, can declare as protected.

```
class BaseClass {  
public:  
    //inherited, visible from outside  
protected:  
    //inherited, private from outside  
private:  
    //private in base class and outside  
};
```

CS2704 (Spring 2000)

5

Private & Protected Inheritance

- Have used public inheritance
class DerivedClass : public BaseClass
- Can replace public with private or protected
 - Public - all public members made private in derived class
 - Protected - all public members made protected

CS2704 (Spring 2000)

6

Design Observations

- A base class should contain common operations and fields
 - Means you should not have to “hide” inherited methods
 - If you do, consider alternative designs
 - Using aggregation
 - A different hierarchy

CS2704 (Spring 2000)

7

Design Observations (2)

- Use fields to keep track of state, methods to keep track of behavior
 - Ex. Base: Vehicle; Derived: Car, Truck
Cars have speed limit of 65, trucks 55
 - Awkward if Car and Truck define own max_speed methods
 - Solution:
 - Add _max_speed, and max_speed accessor to Vehicle class
 - Car and Truck classes set value in constructor

CS2704 (Spring 2000)

8

Design Observations (3)

- Methods of derived-class must preserve assumptions of base-class
 - Should not change state of inherited data to violate assumptions of base class
 - Not a worry if using public inheritance
 - Could be a problem with protected inheritance

CS2704 (Spring 2000)

9

Design Observations (4)

- Objects of derived class should be preserved by inherited methods
 - Ex. Deriving Stack from List
 - List methods may insert anywhere in stack
 - Violates property of being a stack
 - Situation where would have to hide methods

CS2704 (Spring 2000)

10

Design Observations (5)

- Polymorphism should be used where you would use type information
 - Ex. Code of the form

```
if (x is of type 1) do_this();
else if (x is of type 2) do_that();
```
 - Can be replaced by virtual methods
 - Virtual methods easier to maintain

CS2704 (Spring 2000)

11

Design Observations (6)

- Move common behavior to the base class
 - Some methods may be slightly different for different derived classes
 - Try moving method to base class, but define it using simpler virtual methods that derived classes can define easily
 - Prevents having to define more complex methods for all derived classes

CS2704 (Spring 2000)

12

Design Observations (7)

- Don't use protected data
 - Concern is that representation of class will change over time
 - If protected data changes, all derived classes change
 - If methods of base class maintain some property of implementation, better to not provide access to derived classes (ex. Sorted list)

CS2704 (Spring 2000)

13

Composition Strategies (P. Coad)

- Composition = aggregation and association
- *Composition Strategy*: Use composition to extend responsibilities by delegating work to other classes.
- Prefer composition over inheritance

CS2704 (Spring 2000)

14

Inheritance Strategy (Coad)

- Inheritance is used to extend attributes and methods
- Use should be restricted, because the relationship between base and derived classes leads to a weak form of encapsulation

CS2704 (Spring 2000)

15

When to Use Inheritance

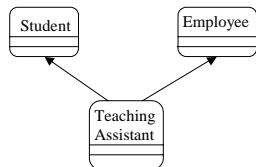
- Inheritance relationship must satisfy:
 1. Represents "is a special kind of", and not "is a role of"
 2. An object of one class in hierarchy never needs to transmute to another class
 3. Derived class extends rather than overriding or nullifying base class
 4. Does not derive for the purpose of copying useful capabilities
 5. If classes from problem domain, represents special kinds of roles, transactions or devices

CS2704 (Spring 2000)

16

Multiple Inheritance

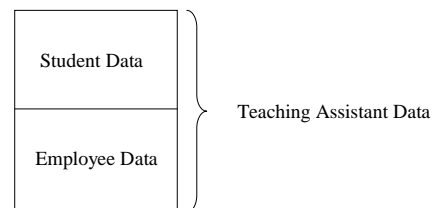
```
class TA : public Student, public Employee {  
    ...  
};
```



CS2704 (Spring 2000)

17

Objects and Multiple Inheritance

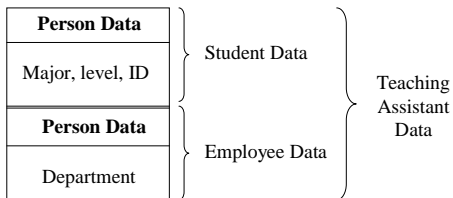


CS2704 (Spring 2000)

18

A Problem with Data

- Don't forget rest of inheritance hierarchy



- A TA could have two names!

CS2704 (Spring 2000)

19

C++ Solution

- Change declarations of base classes


```
class Student : virtual public Person {...};
class Employee : virtual public Person {...};
```
- If inherit both

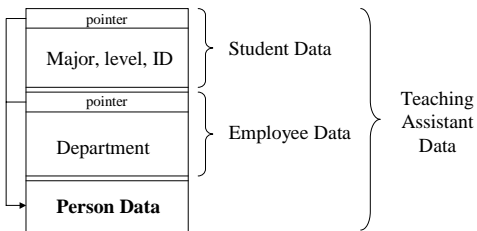

```
class TA : public Student, public Employee { ... };
```
- TA object contains pointers to Person object(s)

CS2704 (Spring 2000)

20

Virtual Inheritance

Hypothetical data layout - compiler may do something else



CS2704 (Spring 2000)

21

Casting & Virtual Inheritance

- Ordinarily casting pointers “does nothing”
- The cast


```
static_cast<Person*> (studentp)
```

 follows pointer
- The cast


```
dynamic_cast<Student*> (st_person)
```

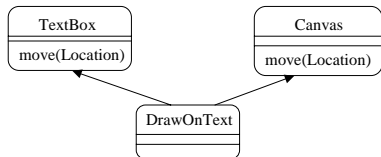
 works if virtual base class has virtual function (usually destructor)

CS2704 (Spring 2000)

22

Methods and Multiple Inheritance

- Both Parents have a move method



CS2704 (Spring 2000)

23

A Problem with Methods

- Can't tell which base class to get method from

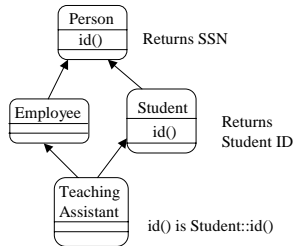

```
DrawOnText y;
y.move(lowerleft);
```
- Could be `TextBox::move(-)` or `Canvas::move(-)`
- Redefine methods with name clash

CS2704 (Spring 2000)

24

Method Dominance

For virtual inheritance



Ambiguous in non-virtual case

CS2704 (Spring 2000)

25

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

CS2704 (Spring 2000)

26

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

CS2704 (Spring 2000)

27

Inheritance Overview

- Useful for coding polymorphism
 - Virtual methods
 - Containers of heterogeneous objects
- Design so that derived class objects are a "special kind of" base class object
- Need to use multiple inheritance is rare
- Consider association and aggregation first

CS2704 (Spring 2000)

28