Lecture 30: C++

CSCI 131
Spring, 2011

Kim Bruce

Smalltalk hierarchies

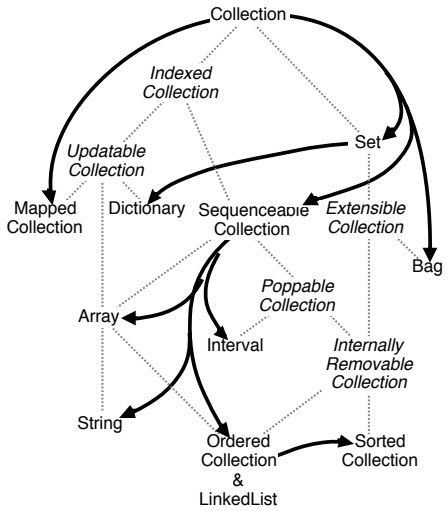


Figure 5: Interfaces versus Inheritance

C++ Design Goals

- Data abstraction & OO features
- Better static type checking
- Backwards compatibility w/ C
- Efficiency: If you do not use a feature, you should not pay for it
- Explicitly hybrid language -- C w/abstraction

Additions to C

- type bool
- reference types & call by reference
- user-defined overloading
- templates
- exceptions
- public or private inheritance

Problems

- Confusing casts and conversions
- Objects allocated on stack
  - what happens w/subtyping? truncation!
- Overloading methods -- see earlier examples!
- Multiple inheritance (*later*)

Casts & Conversions

- Implicit conversions:
  - from short to int
  - class B { public: B (A a) {} }; A a; B b = a;
- Explicit conversions:
  - C c; D* d; d = (D*) &c; d -> DonlyMeth();
- Try to avoid problems by using new casts:
  - static_cast, dynamic_cast, reinterp_cast, const_cast
  - dynamic_cast checks using run-time type info (RTTI)
  - reinterp_cast trusts

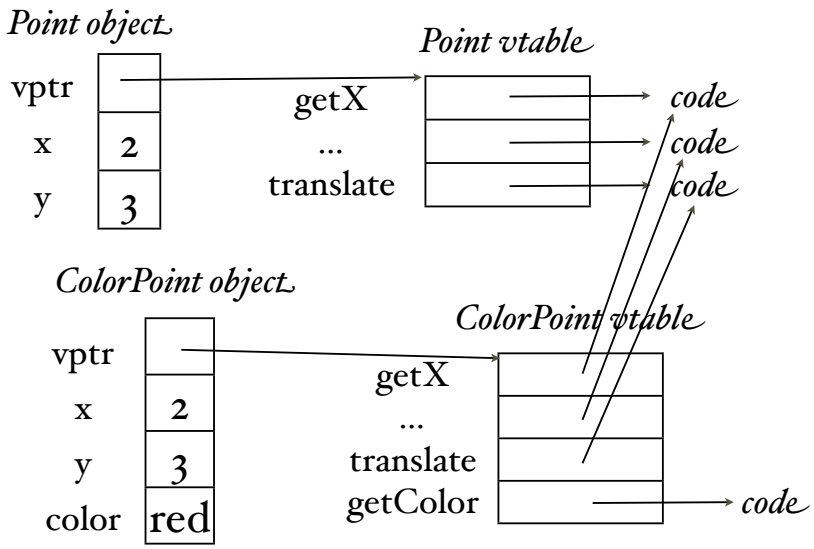
Objects on stack

- Doesn't interact well with subtyping.
- Point p; // allocates point on stack
- ColorPoint cp(3,4,BLUE);
- p = cp; // slices and converts to Point
- Call by value has similar problems
- What about reference parameters to methods?

OO Features

- Visibility
  - Public, protected, private
  - Friends ...
- Virtual vs. nonvirtual functions
  - don't pay the price of dynamic method invocation
- Implemented via vtable
  - no search necessary
  - static typing makes efficient rep possible
  - *efficient iff subtype from inheritance!*

VTable for Virtual methods



C++ vs Smalltalk implementation

- No search in C++ since offset for given method same in base and derived classes
- Smalltalk has no type declaration
  - value not known to be subtype of declared type
  - no idea where method is located

Abstract classes

- Have at least one method undefined
- "Pure" leaves all undefined
- Can't construct, but can inherit from
- Derived subclasses can be used as subtypes of abstract base class.

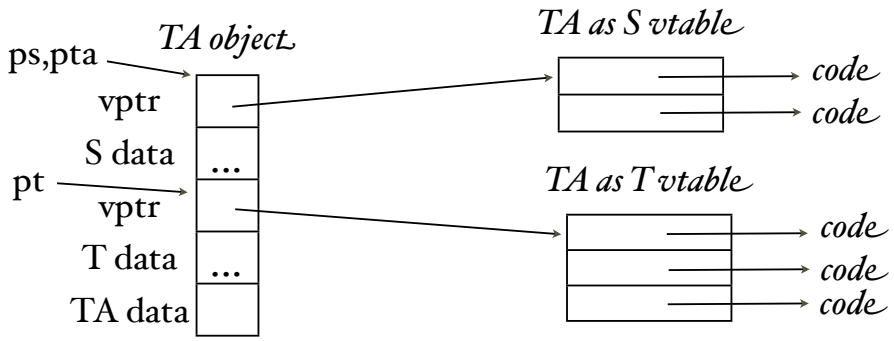
Multiple Inheritance

- Appealing: TA derived from Student and Teacher.
- Added to C++ and Smalltalk. In Eiffel from beginning.
- Problems conceptually and with implementation

MI in C++

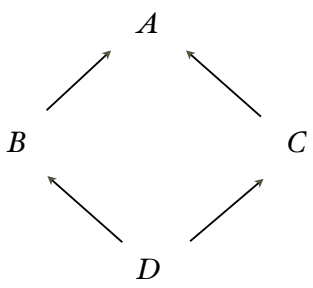
```
class S {...}
class T{...}
class TA: public S, public T
{...}

TA* pta = new TA();
S * ps = pta;
T * pt = pta;
```

Representing MI



What if T and TA both define virtual f?
T methods expect inst vbles starting at pt.
How get access to instance vbles from S?

Conceptual Problems w/ MI



Diamond Inheritance: Suppose A has virtual f
and B and C override it.
Which version is inherited in D?

Java Solution

- Most multiple inheritance in C++ involves pure base classes.
- Java: Single inheritance, but can implement multiple interfaces.
- Avoids problems.
- *Traits (e.g., in Scala) are modern alternative.*

C++ Summary

- One of most complicated languages ever
  - design by accretion
- Meets design goals but very hard to get right
  - C makes it easy to shoot yourself in the foot. In C++ it's harder to shoot yourself in the foot, but when you do, you blow off your whole leg. -- Stroustrup
- Memory management is big problem
- Most programmers learn a subset.

C++ Humor

- C++: Hard to learn and built to stay that way.
- Java is, in many ways, C++--.
- How C++ is like teenage sex:
  1. It is on everyone's mind all the time.
  2. Everyone talks about it all the time.
  3. Everyone thinks everyone else is doing it.
  4. Almost no one is really doing it.
  5. The few who are doing it are:
     A. Doing it poorly.
     B. Sure it will be better next time.
     C. Not practicing it safely.