CS11600: Introduction to Computer Programming (C++)

Lecture 14

Svetlozar Nestorov

University of Chicago

Outline

- Inheritance
- Polymorphism
- Derived classes
- Static vs. dynamic binding
- Virtual functions

Inheritance

- Inheritance – essential feature of OO.
- Implements i-sa hierarchies.
  - Derived class is-a Base class.
- When to use inheritance?
  - Different behavior vs. different value.
- Examples.

Polymorphism

- Parametric: in ML.
- "Ad-hoc": overloading.
- Object-oriented:
  - Invoking methods on an object without knowing its exact type (at compile time) results in correct behavior.
- Examples.

Initialization Rules

- Constructors
  - Order: base, then derived.

```
Derived : Base() {…}
```

- Destructors
  - Order: derived, then base.
  - Will be called by the compiler.

Access Rules for Derived Classes

- Public derivation
  - Inherit private and public members (data and functions)
  - Access *only* public members
- Protected members are *private* to the outside world but *public* to derived classes.
- Examples.

Substitution Rules

- An object of the derived class may be used where an object of the base type is expected.
- An object of the derived class can always be converted to an object of the base type.
- There is no conversion from base class to derived class.

Dominance

- A function f in the derived class *dominates* (hides) all f functions in the base class even if they have different signatures.
  - Except for *virtual* functions.
- So, no overloading for base and derived classes per se.
- But, there is dynamic binding with virtual functions…

Static and Dynamic Binding

- Static binding
  - Object type is known at compile time, so a function is chosen at compile time.
- Problem: object type at run time may be different from the type we can determine at compile time. How?
- Dynamic binding
  - Perform binding at run time, based on the actual object.

Virtual Functions

- A dynamic binding mechanism in C++.
- Virtual function are bound dynamically (based on the object type at *run time*).
- Virtual destructors
- Virtual constructors?!?