LECTURE 42:INHERITANCE

Today

- Reading
  - Weiss Ch. 6
- Objectives
  - Inheritance in C++
    - Slicing
    - Dynamic dispatching
    - Casting

Inheritance in C++

- Syntax of declaring a subclass

```
class Derived : public Base
```

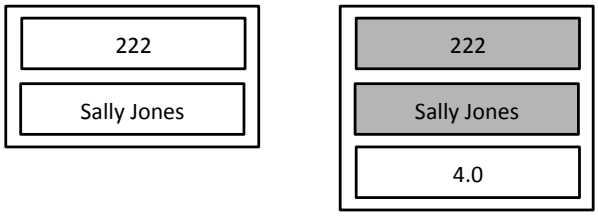
- Public inheritance implements an "isA" relationship

Inheritance in C++

- Derived class inherits
  - All public and protected members
- Derived class does *not* inherit
  - Base class constructors, destructor, operator=
  - private members
  - (any friends)
- Constructor of derived class must call constructor of base class

Slicing

- Slicing – when derived copied into base, only fields from base class are preserved
- Slicing occurs whenever objects are copied
  - For example, call-by-value or return-by-value



Slicing

- When using objects, there is no real way to exploit inheritance
- If you want subtyping, use pointers and references
- Assignment of pointers works as expected!
- E.g., if you want a vector of Person or subclass,

```
vector<Person*> people;
```

Static vs. Dynamic dispatching

- Static dispatching
  - Determine which member function to call by checking type at *compile* time
- Dynamic dispatching
  - Determine which member function to call by checking type at *runtime*

The virtual keyword

- The `virtual` keyword signals that the function uses dynamic dispatching
- Allows this function to be overwritten in subclasses
- If don't use `virtual`, the function called is based on compile-time type not runtime type

Casting in C++

- Type casts in C++ always succeed!
- Downcasting on pointers always succeeds!
- To get checked conversions, use `dynamic_cast`
  - Returns `NULL` if the cast is incorrect
  - `dynamic_cast` does a compile time *and* runtime check to make sure the cast can work
  - requires that the object you're casting has polymorphic type, i.e. has at least one virtual method

Semester in review

Topics

- Pre- and post- conditions, assertions, unit testing, debugger
- Advanced Java topics: generics, graphics, iterators, comparators
- Complexity and correctness: Big-O notation, induction, sorting
- Data structures
  - ArrayList, linked lists, stacks, queues, binary (search) trees, splay trees, heaps, priority queues, maps, graphs
  - Time/space tradeoff
- Parallelism and Concurrency
- C++ and memory management

Final Exam

- Friday May 15th at 9 am
- 3 hours, closed book, closed notes
- ~8-10 questions
- Comprehensive but mostly focused on material after midterm
- C++ for Java Programmers
  - End of chapter problems for Chapters 3 and 4

Questions?