(11-2) OOP: Inheritance in C++
D & D Chapter 11

Instructor - Andrew S. O'Fallon
CptS 122 (March 31, 2021)
Washington State University

Key Concepts

- Base and derived classes
- `Protected` members
- Inheritance
  - `public`, `protected`, and `private` accessibility modes
  - *is-a* relationship
  - *Single* and *multiple*
  - *Multilevel, hierarchical,* and *hybrid*
- Software reuse through inheritance

Introduction to Inheritance in OOP (I)

- Inheritance may be viewed as a form of *software reuse* or the process of creating new classes from existing classes
- *Inheritance* allows for the implementation of a class that acquires another class' attributes and operations (its capabilities)
  - The class customizes or enhances the capabilities of the acquired class
- *Software reuse* allows for higher levels of developer production through leveraging tested, quality code

Introduction to Inheritance in OOP (II)

- How inheritance works!
  - When implementing a new class some data members (attributes) and member functions (operations) might be in common between the new class and an existing class – the new class could *inherit* the members of the existing class
    - The existing class is referred to as the *base* class (or *superclass*)
    - The new class, which acquires the members, is referred to as the *derived* class (or *subclass*)
      - Represents a more customized or *specialized* version of objects

Introduction to Inheritance in OOP (III)

- The *is-a* relationship represents inheritance

  For example:

  Let's say we have a base class called Employee and a derived class called Manager – A Manager *is an* Employee (but, note, an Employee is not necessarily a Manager)

- In contrast the *has-a* relationship represents *composition*, where an object contains >= 1 objects of other classes as members

  Some possibilities include:

  – An Employee *has a* "dental plan" (`class` DentalPlan), *has an* "office" (`class` Office), etc.

What is Inherited?

- A derived class inherits every member of a base class *except* its:
  - Constructor(s)
  - Destructor
  - Friend(s)
  - Overloaded assignment operator

Base and Derived Classes

- Base classes tend to be more *general*
- Derived classes tend to be more *specific*
- We've established that every derived class is an object of it's base class so…
  - The set of objects representative of the base class is usually *larger* than the set of objects representative of any of its derived classes
    - An Employee class could be representative of all employee types including managers, supervisors, directors, officers, etc.
    - A Manager class is a *smaller*, more *specific* subset of employees

Protected Members

- The access specifier `protected` provides an intermediate level of protection between `private` and `public`
- Derived classes, and any of its *friends*, have access to `protected` members of a base class, but any *nonmembers* that are *not* friends do not have access

Forms of Inheritance

- There are 5 forms of inheritance
  - Single
  - Multiple
  - Multilevel
  - Hierarchical
  - Hybrid

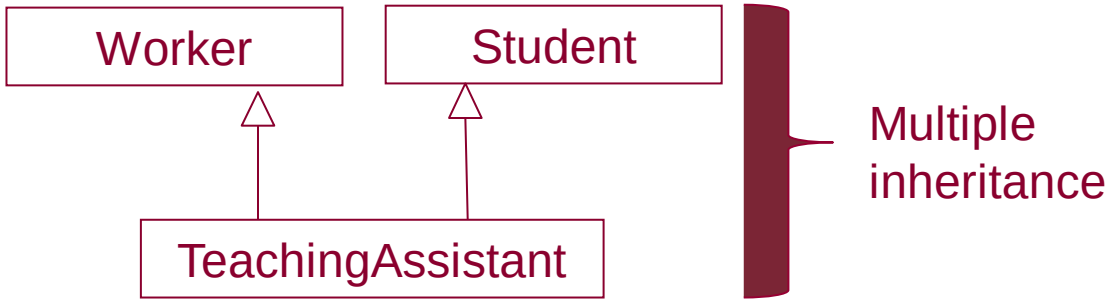
Single Inheritance - Inheritance Structure of Employees of a Business (I)

Single Inheritance - Inheritance Structure of Employees of a Business (II)

- *Single* inheritance
  - One derived class inherits from only one base class
  - A Manager inherits capabilities of an Employee *only*
  - C++ syntax

```
class Manager : public Employee
{
        // class declarations
};
```


Multiple Inheritance - Inheritance Structure of University Members (I)

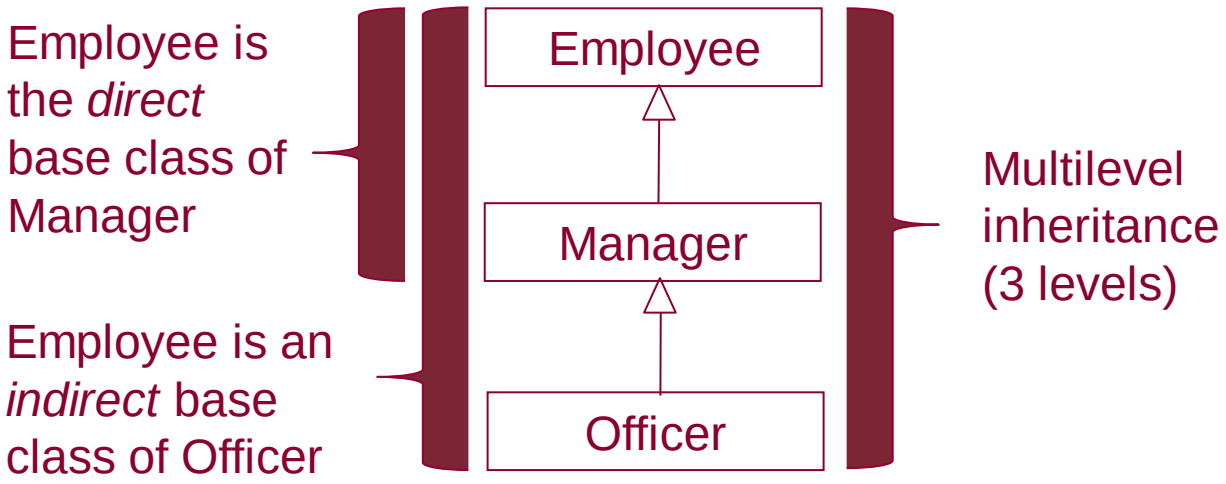


Multiple Inheritance - Inheritance Structure of University Members (II)

- *Multiple* inheritance
  - A derived class inherits from more than one base class
  - A TeachingAssistant inherits capabilities of a Worker *and* Student
  - C++ syntax

```
class TeachingAssistant: public Worker, public Student
{
        // class declarations
};
```

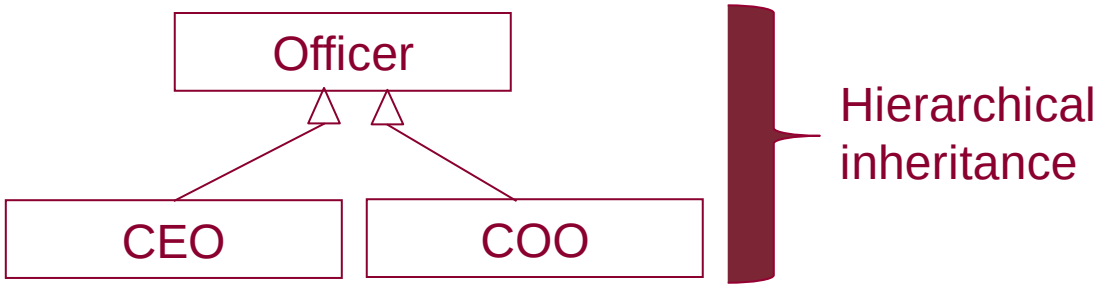

Multilevel Inheritance - Inheritance Structure of Employees of a Business (I)



Multilevel Inheritance - Inheritance Structure of Employees of a Business (II)

- *Multilevel* inheritance
  - A derived class acts as a base class for another derived class
  - An Officer is created from a Manager and a Manager is created from an Employee
    - An Officer is a type of Manager and a Manager is a type of Employee
  - Generally want no more than a few levels

Hierarchical Inheritance - Inheritance Structure of Employees of a Business (I)

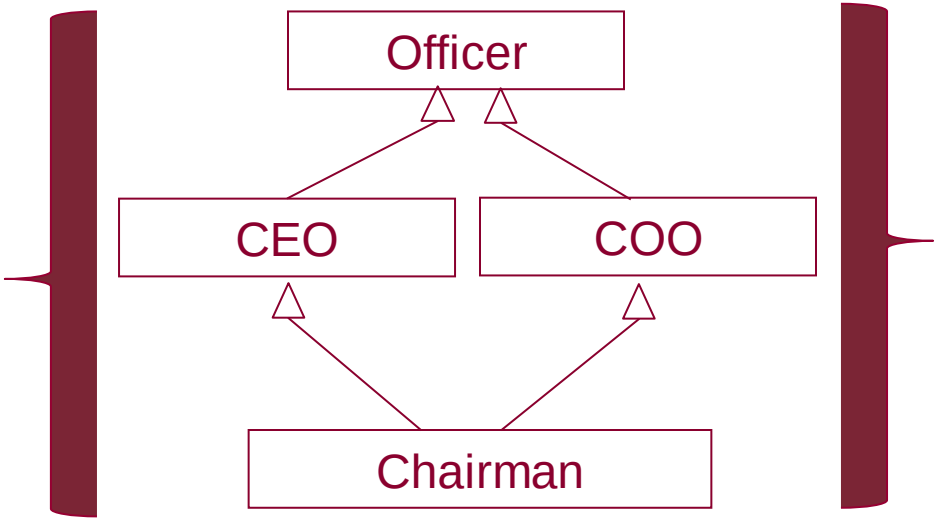


Hierarchical Inheritance - Inheritance Structure of Employees of a Business (II)

- *Hierarchical* inheritance
  - Multiple derived classes inherit from the same base class
  - CEO (Chief Executive Officer) and COO (Chief Operations Officer) have attributes of an Officer, but also have their own unique attributes

Hybrid Inheritance - Inheritance Structure of Employees of a Business (I)

The Diamond Problem!



Hybrid inheritance

Hybrid Inheritance - Inheritance Structure of Employees of a Business (II)

- *Hybrid* inheritance
  - Two or more inheritance forms are combined
  - A Chairman inherits from both CEO (Chief Executive Officer) and COO (Chief Operations Officer) classes, and CEO and COO inherit from Officer – forms a diamond relationship
    - Here the "diamond" problem occurs because CEO and COO inherit from Officer, which have own copies of the data members and methods – Chairman contains two subobjects - there is ambiguity in which members are accessed by Chairman
      - We'll solve this problem with keyword `virtual` – to be explained along with polymorphism later!

Accessibility Modes and Inheritance in C++ (I)

- `public`, `protected`, and `private`
  - X in the table indicates hidden from derived class

		Inheritance Mode		
		public	protected	private
Members in Base Class	public	public	protected	private
	protected	protected	protected	private
	private	X	X	X
		Members in derived class		

- table courtesy of http://www.codingunit.com/cplusplus-tutorial-inheritance

Accessibility Modes and Inheritance in C++ (II)

- `public`
  - C++ syntax

```
class Manager : public Employee
{
        // class declarations
};
```

- `protected`
  - C++ syntax

```
class Manager : protected Employee
{
        // class declarations
};
```

- `private`
  - C++ syntax

```
class Manager : private Employee
{
        // class declarations
};
```


Summary of Inheritance (I)

- Advantages
  - Software reuse
  - Reduces code redundancy
  - Reduces code size
  - Promotes readability
  - Promotes extensibility
    - Extensibility is a software design principle which considers growth of the system – a system's ability to extend the system with new functionality with minimal changes and impact to the existing system's functionality

Summary of Inheritance (II)

- Disadvantages
  - Base classes and derived classes are tightly coupled – a change to the base class could impact all classes derived from it
  - With a class hierarchy, many data members could remain unused, possibly affecting performance

In a Few Lectures…

- Soon we will discuss polymorphism! Let inheritance sink in first!

References

- P.J. Deitel & H.M. Deitel, *C++: How to Program* (9th ed.), Prentice Hall, 2014
- J.R. Hanly & E.B. Koffman, *Problem Solving and Program Design in C (7th Ed.)*, Addison-Wesley, 2013

Collaborators

- Jack Hagemeister