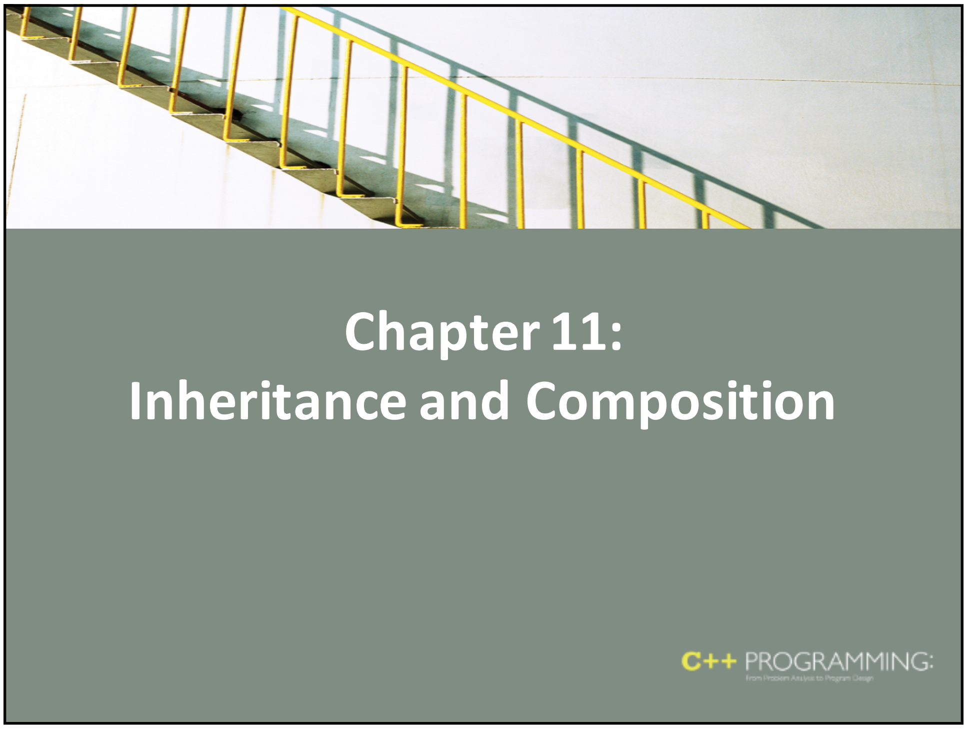
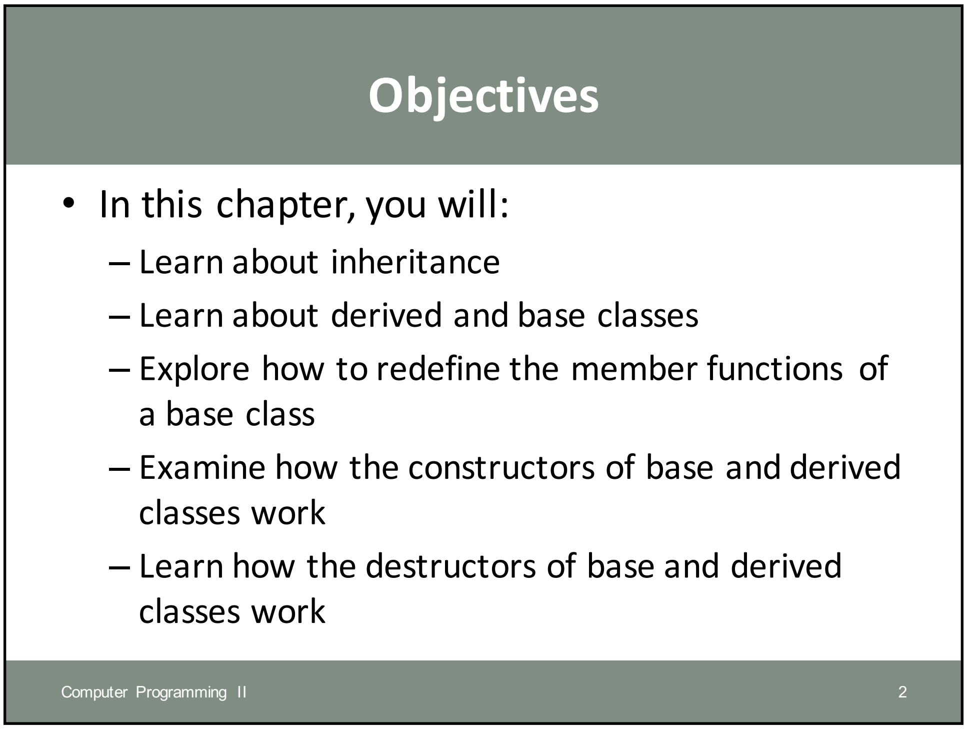


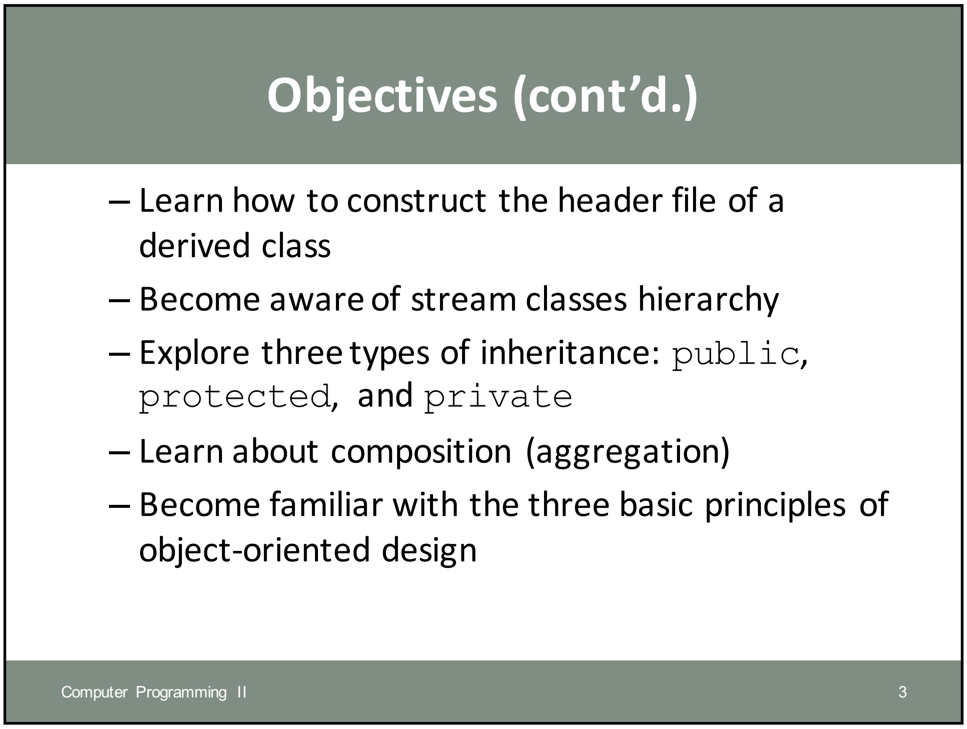
Chapter 11: Inheritance and Composition

C++ PROGRAMMING: From Problem Analysis to Program Design



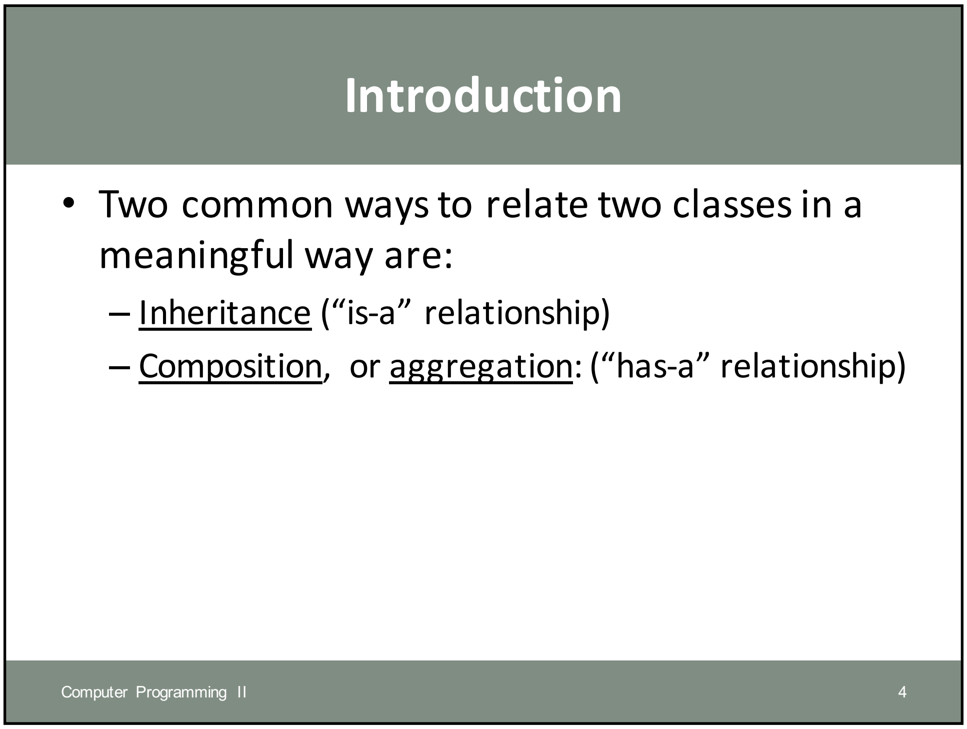
Objectives

- In this chapter, you will:
  - Learn about inheritance
  - Learn about derived and base classes
  - Explore how to redefine the member functions of a base class
  - Examine how the constructors of base and derived classes work
  - Learn how the destructors of base and derived classes work



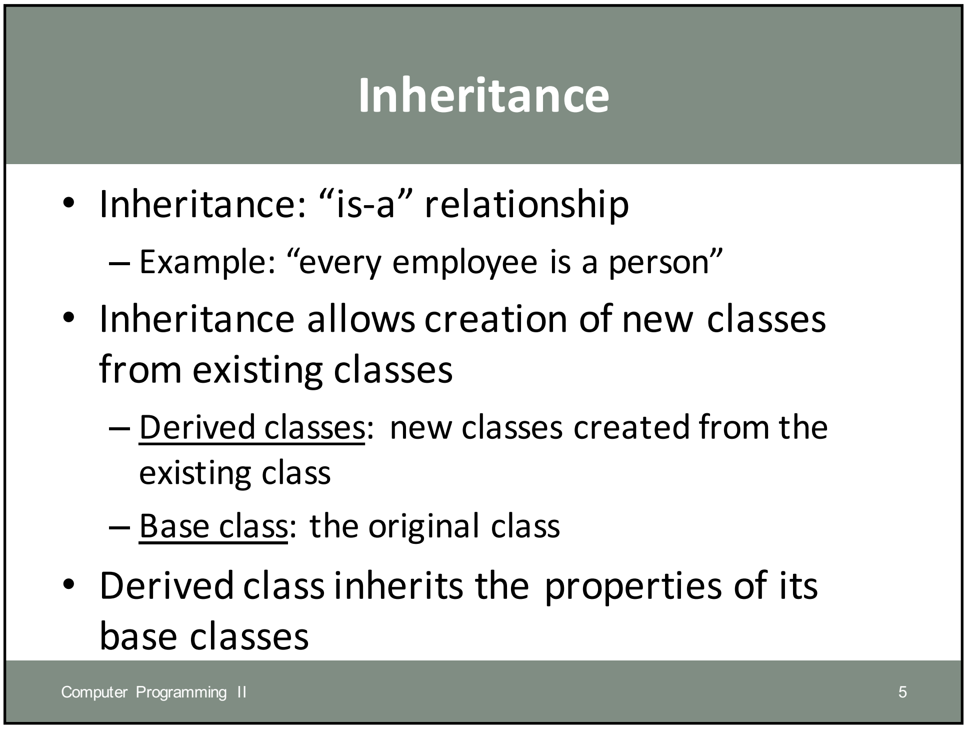
Objectives (cont'd.)

- Learn how to construct the header file of a derived class
- Become aware of stream classes hierarchy
- Explore three types of inheritance: `public`, `protected`, and `private`
- Learn about composition (aggregation)
- Become familiar with the three basic principles of object-oriented design



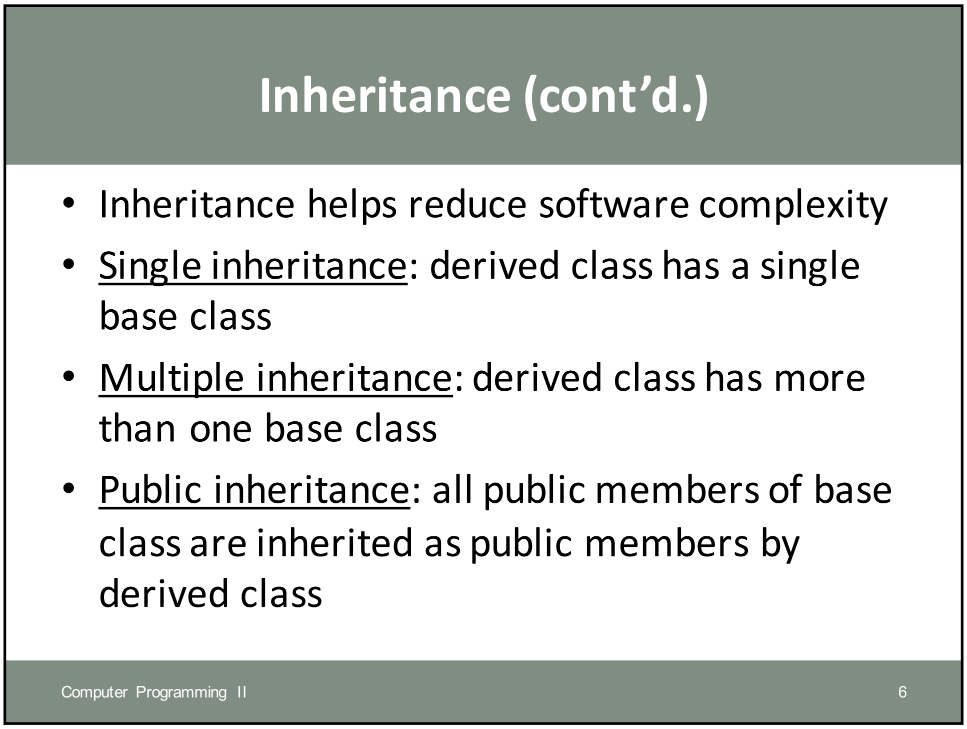
Introduction

- Two common ways to relate two classes in a meaningful way are:
  - Inheritance ("is-a" relationship)
  - Composition, or aggregation: ("has-a" relationship)



Inheritance

- Inheritance: “is-a” relationship
  - Example: “every employee is a person”
- Inheritance allows creation of new classes from existing classes
  - Derived classes: new classes created from the existing class
  - Base class: the original class
- Derived class inherits the properties of its base classes



Inheritance (cont’d.)

- Inheritance helps reduce software complexity
- Single inheritance: derived class has a single base class
- Multiple inheritance: derived class has more than one base class
- Public inheritance: all public members of base class are inherited as public members by derived class

Inheritance (cont'd.)

- Inheritance can be viewed as a tree-like, or hierarchical, structure between the base class and its derived classes

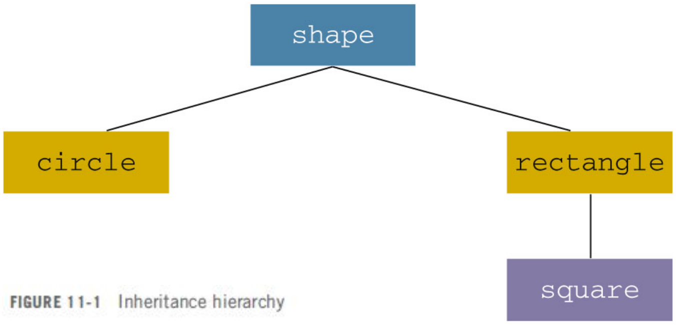


FIGURE 11-1 Inheritance hierarchy

Inheritance (cont'd.)

- Syntax of a derived class:

```
class className: memberAccessSpecifier baseClassName
{
    member list
};
```

  - `memberAccessSpecifier` is `public`, `protected`, or `private` (default)
- `private` members of a base class are `private` to the base class
  - Derived class cannot directly access them

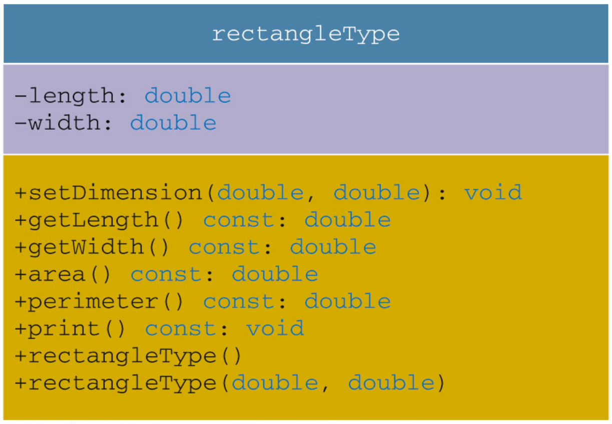
Inheritance (cont'd.)

- public members of base class can be inherited as public or private members
- Derived class can include additional members (data and/or functions)
- Derived class can redefine public member functions of the base class
  - Applies only to the objects of the derived class
- All member variables of the base class are also member variables of the derived class

Redefining (Overriding) Member Functions of the Base Class

- To redefine a public member function:
  - Corresponding function in derived class must have same name/number/types of parameters
- If derived class overrides a public member function of the base class, then to call the base class function, specify:
  - Name of the base class
  - Scope resolution operator (::)
  - Function name with appropriate parameter list

Redefining Member Functions of the Base Class (cont'd.)

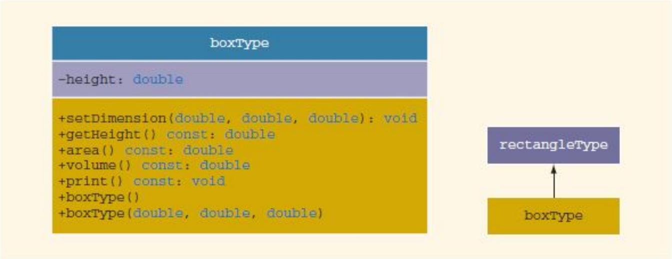


```
rectangleType
-length: double
-width: double
+setDimension(double, double): void
+getLength() const: double
+getWidth() const: double
+area() const: double
+perimeter() const: double
+print() const: void
+rectangleType()
+rectangleType(double, double)
```

FIGURE 11-2 UML class diagram of the **class** rectangleType

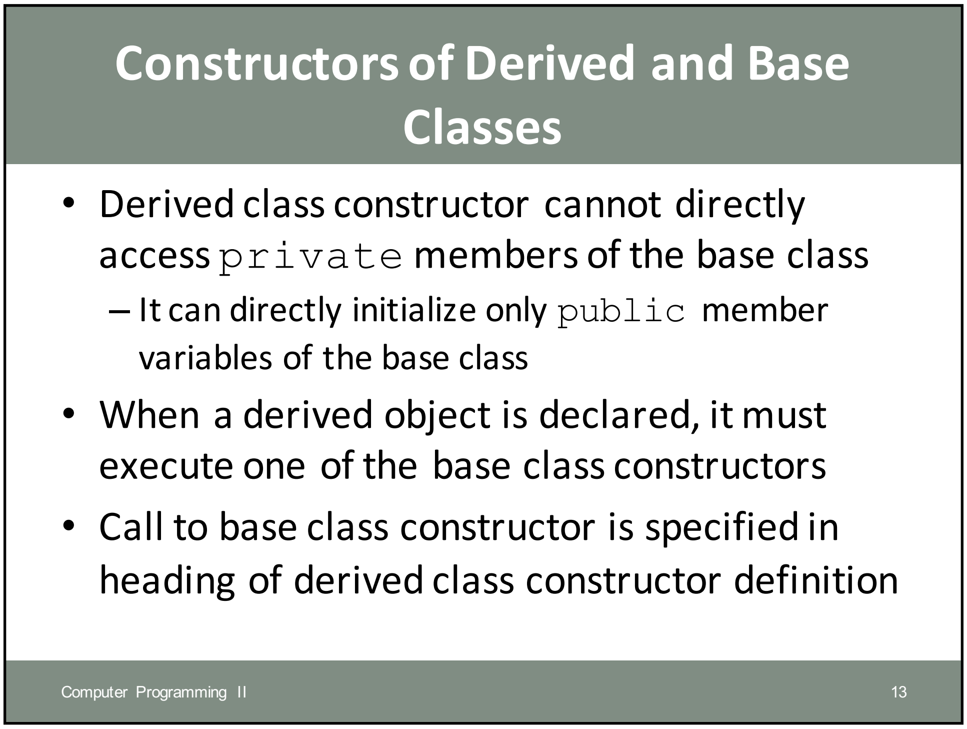
Redefining Member Functions of the Base Class (cont'd.)

- `boxType` is derived from `rectangleType`, and it is a `public` inheritance
  - Also overrides `print` and `area`



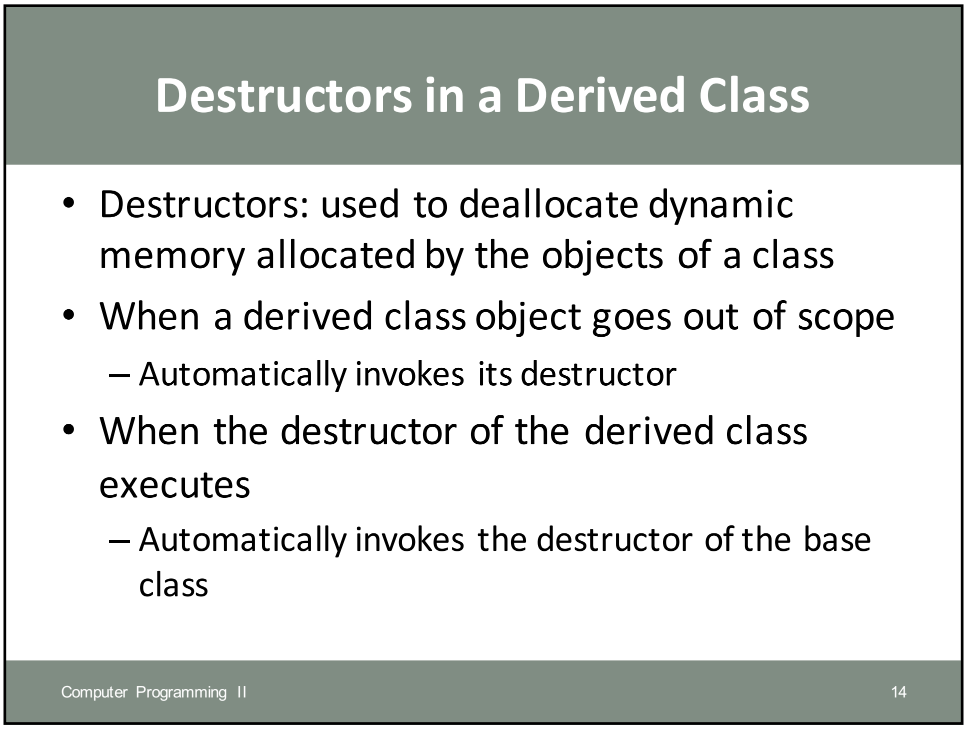
```
boxType
-height: double
+setDimension(double, double, double): void
+getHeight() const: double
+area() const: double
+volume() const: double
+print() const: void
+boxType()
+boxType(double, double, double)
```

FIGURE 11-3 UML class diagram of the **class** boxType and the inheritance hierarchy



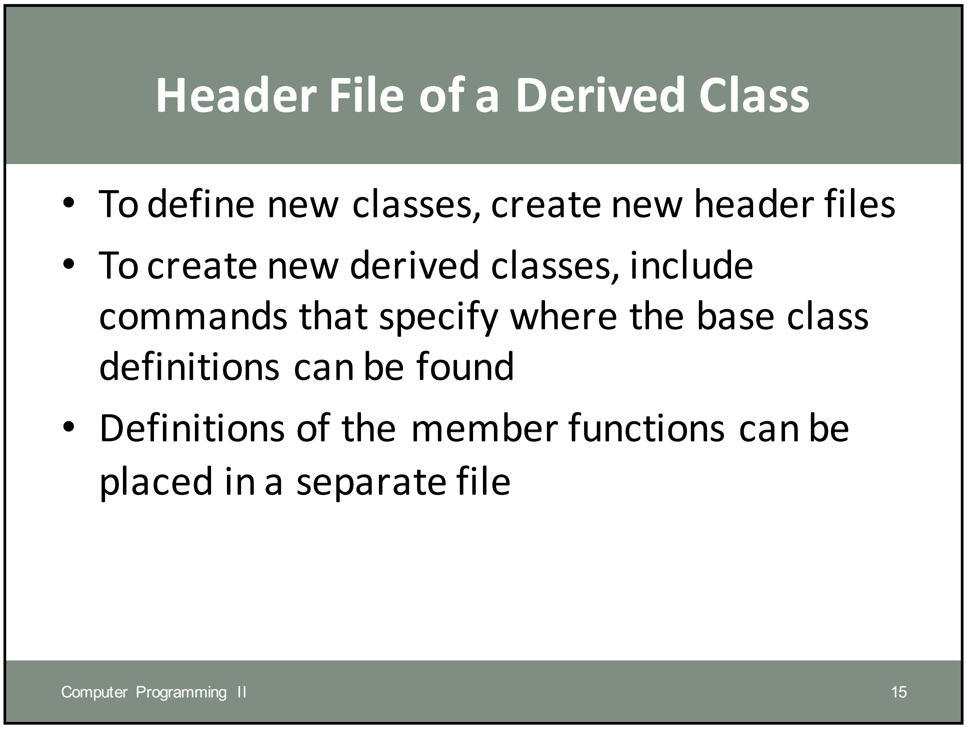
Constructors of Derived and Base Classes

- Derived class constructor cannot directly access `private` members of the base class
  - It can directly initialize only `public` member variables of the base class
- When a derived object is declared, it must execute one of the base class constructors
- Call to base class constructor is specified in heading of derived class constructor definition



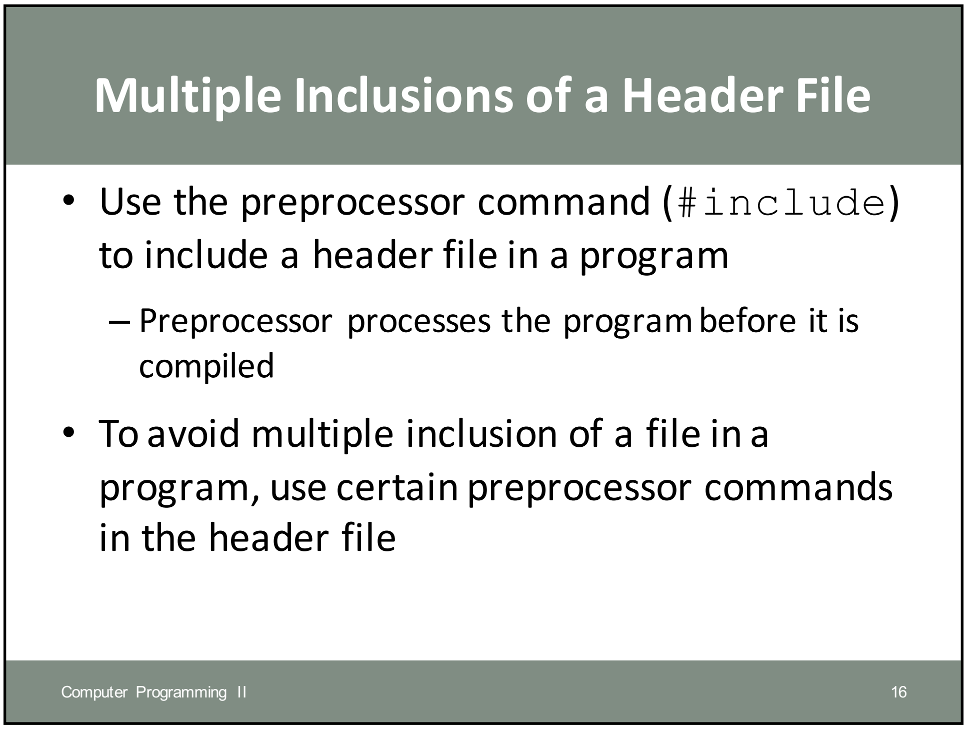
Destructors in a Derived Class

- Destructors: used to deallocate dynamic memory allocated by the objects of a class
- When a derived class object goes out of scope
  - Automatically invokes its destructor
- When the destructor of the derived class executes
  - Automatically invokes the destructor of the base class



Header File of a Derived Class

- To define new classes, create new header files
- To create new derived classes, include commands that specify where the base class definitions can be found
- Definitions of the member functions can be placed in a separate file



Multiple Inclusions of a Header File

- Use the preprocessor command (`#include`) to include a header file in a program
  - Preprocessor processes the program before it is compiled
- To avoid multiple inclusion of a file in a program, use certain preprocessor commands in the header file

C++ Stream Classes

- `ios` is the base class for all stream classes
  - Contains formatting flags and member functions to access/modify the flag settings

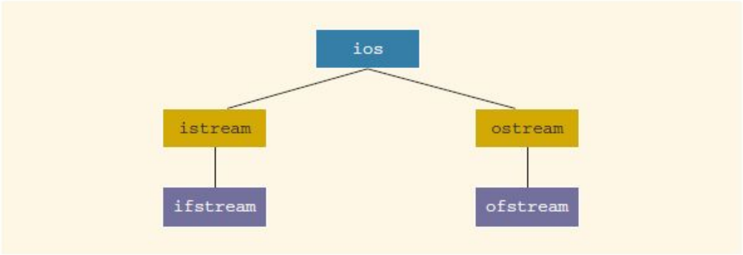
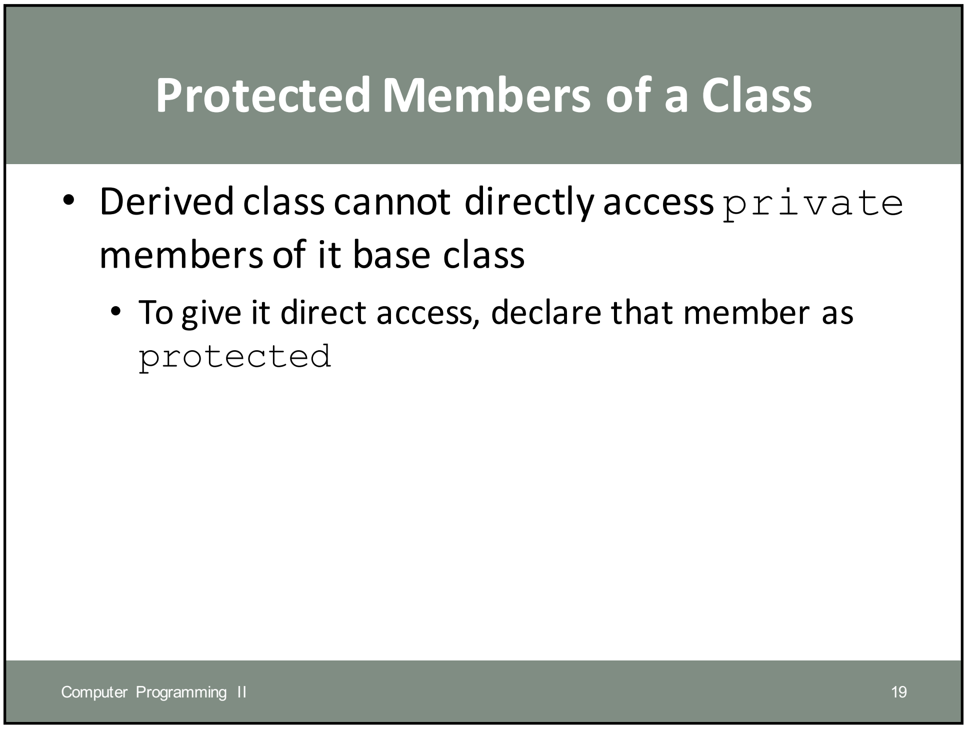


FIGURE 11-6 C++ stream classes hierarchy

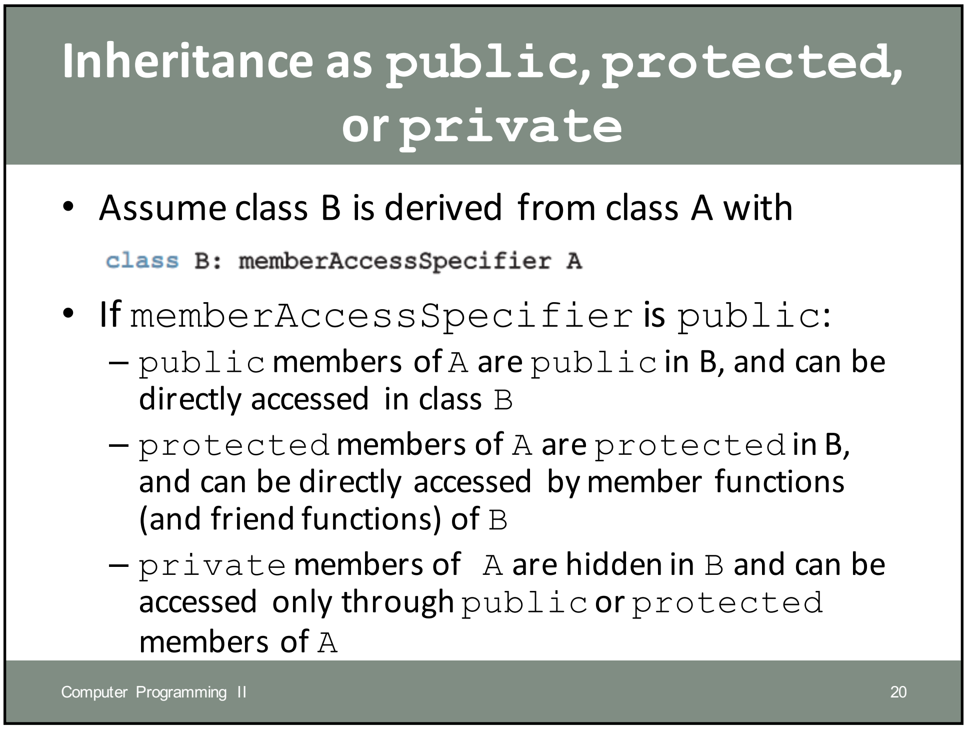
C++ Stream Classes (cont'd.)

- `istream` and `ostream` provide operations for data transfer between memory and devices
  - `istream` defines the extraction operator (>>) and functions `get` and `ignore`
  - `ostream` defines the insertion operator (<<) which is used by `cout`
- `ifstream`/`ofstream` objects are for file I/O
  - Header file `fstream` contains the definitions for these



Protected Members of a Class

- Derived class cannot directly access `private` members of it base class
  - To give it direct access, declare that member as `protected`

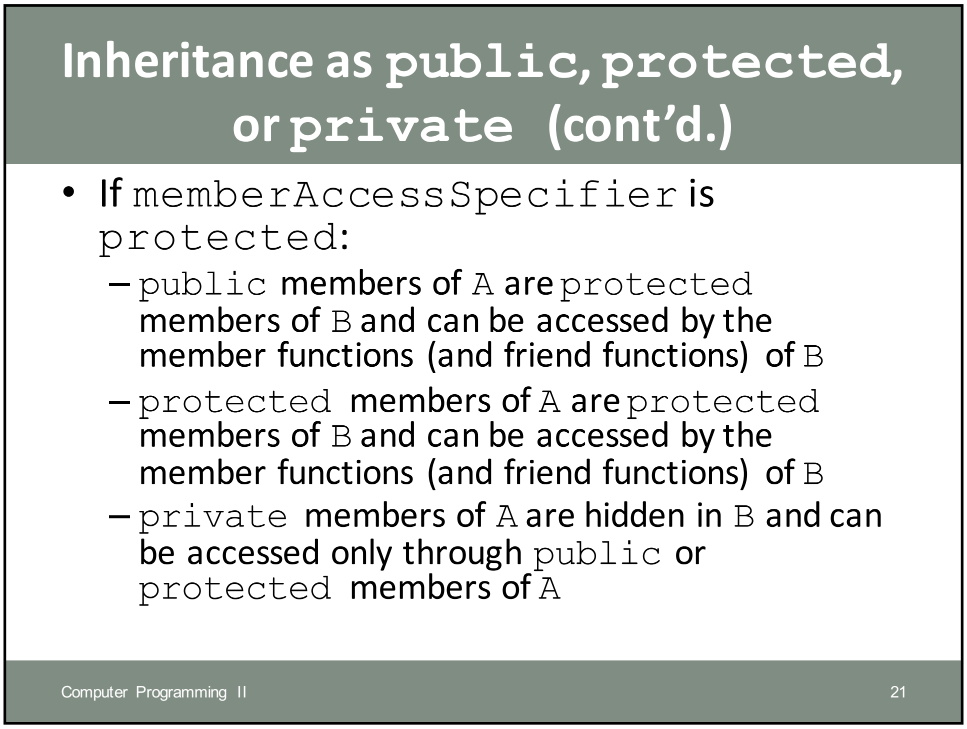


Inheritance as `public`, `protected`, or `private`

- Assume class B is derived from class A with

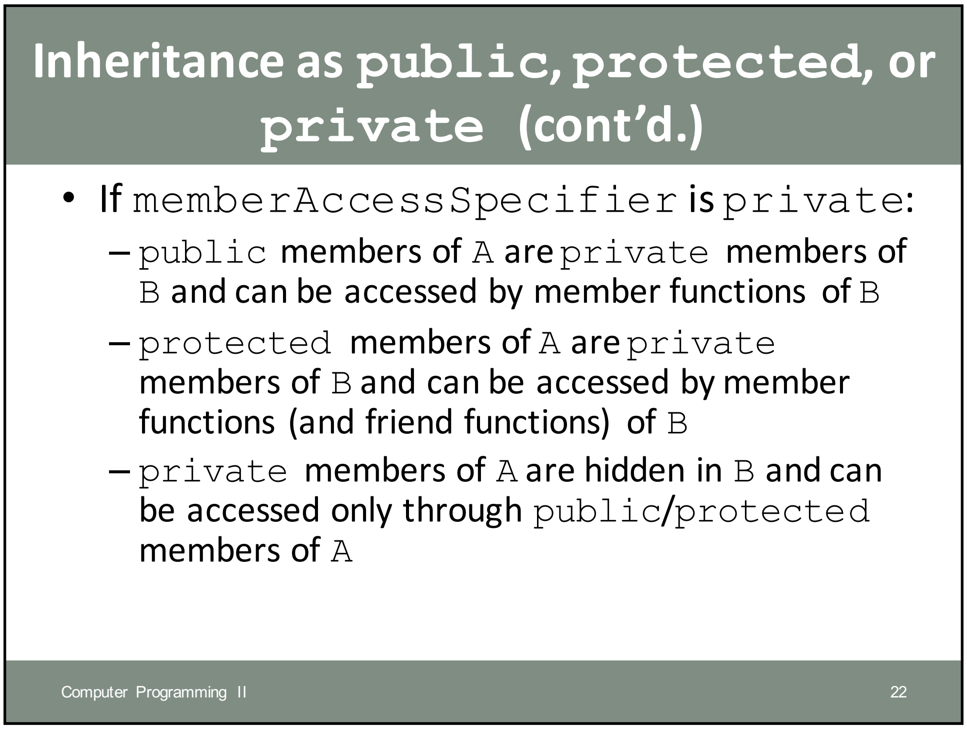
```
class B: memberAccessSpecifier A
```

- If `memberAccessSpecifier` is `public`:
  - `public` members of `A` are `public` in B, and can be directly accessed in class `B`
  - `protected` members of `A` are `protected` in B, and can be directly accessed by member functions (and friend functions) of `B`
  - `private` members of `A` are hidden in `B` and can be accessed only through `public` or `protected` members of `A`



Inheritance as `public`, `protected`, or `private` (cont'd.)

- If `memberAccessSpecifier` is `protected`:
  - `public` members of `A` are `protected` members of `B` and can be accessed by the member functions (and friend functions) of `B`
  - `protected` members of `A` are `protected` members of `B` and can be accessed by the member functions (and friend functions) of `B`
  - `private` members of `A` are hidden in `B` and can be accessed only through `public` or `protected` members of `A`



Inheritance as `public`, `protected`, or `private` (cont'd.)

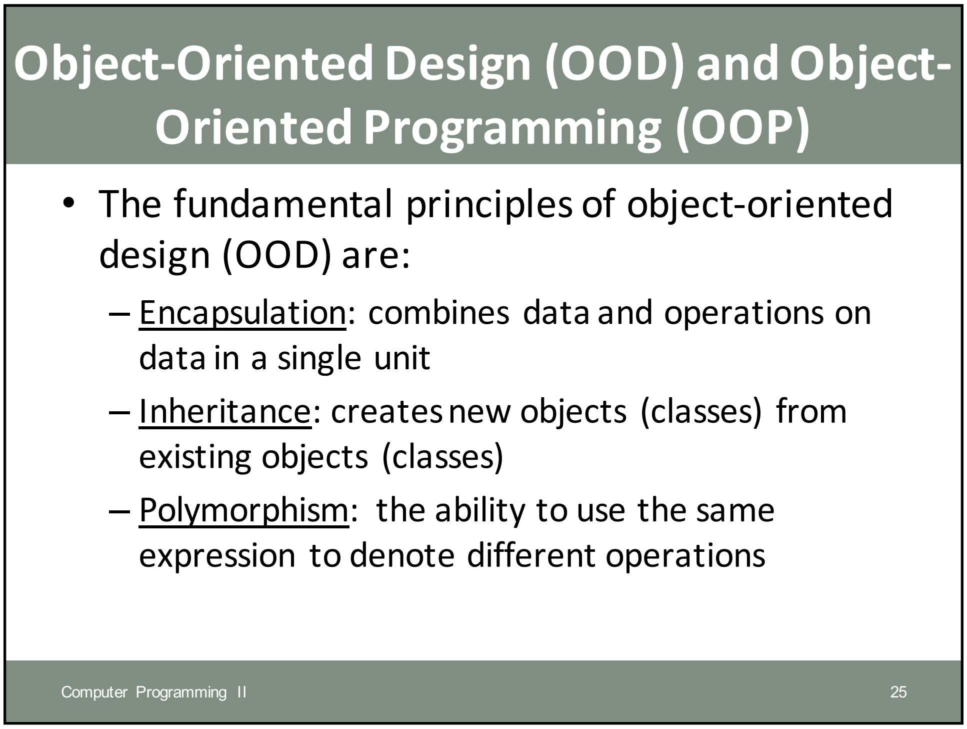
- If `memberAccessSpecifier` is `private`:
  - `public` members of `A` are `private` members of `B` and can be accessed by member functions of `B`
  - `protected` members of `A` are `private` members of `B` and can be accessed by member functions (and friend functions) of `B`
  - `private` members of `A` are hidden in `B` and can be accessed only through `public`/`protected` members of `A`

Composition (Aggregation)

- In composition, one or more member(s) of a class are objects of another class type
- <u>Composition</u> (<u>aggregation</u>): “has-a” relation
- Arguments to the constructor of a member-object are specified in the heading part of the definition of the constructor

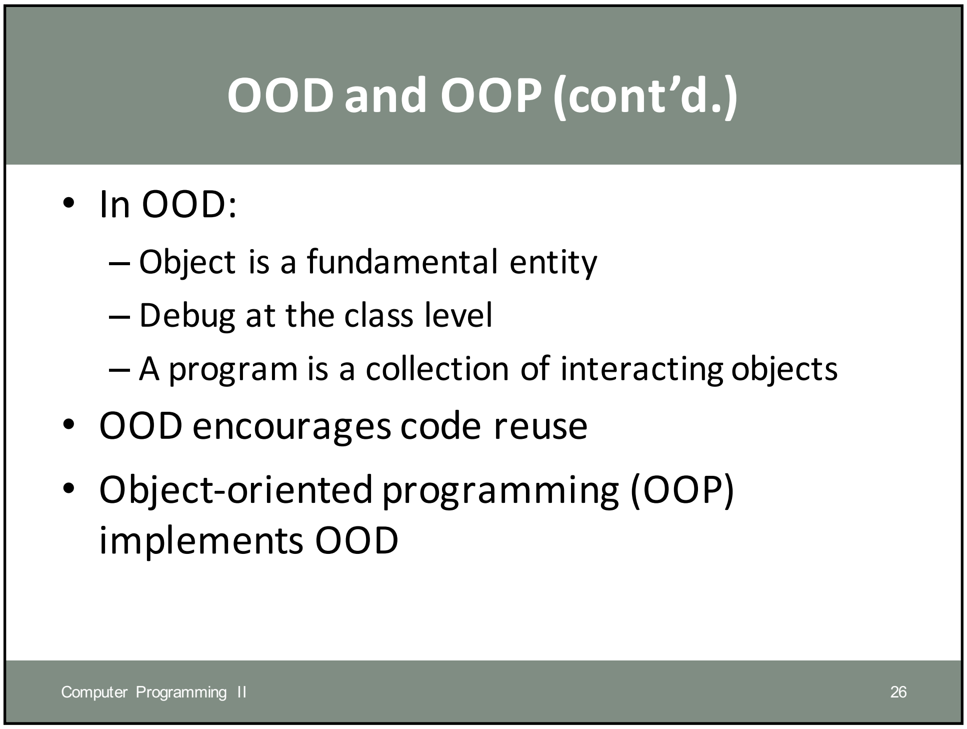
Composition (Aggregation) (cont’d.)

- Member-objects of a class are constructed in the order they are declared
  - Not in the order listed in the constructor’s member initialization list
- They are constructed before the containing class objects are constructed



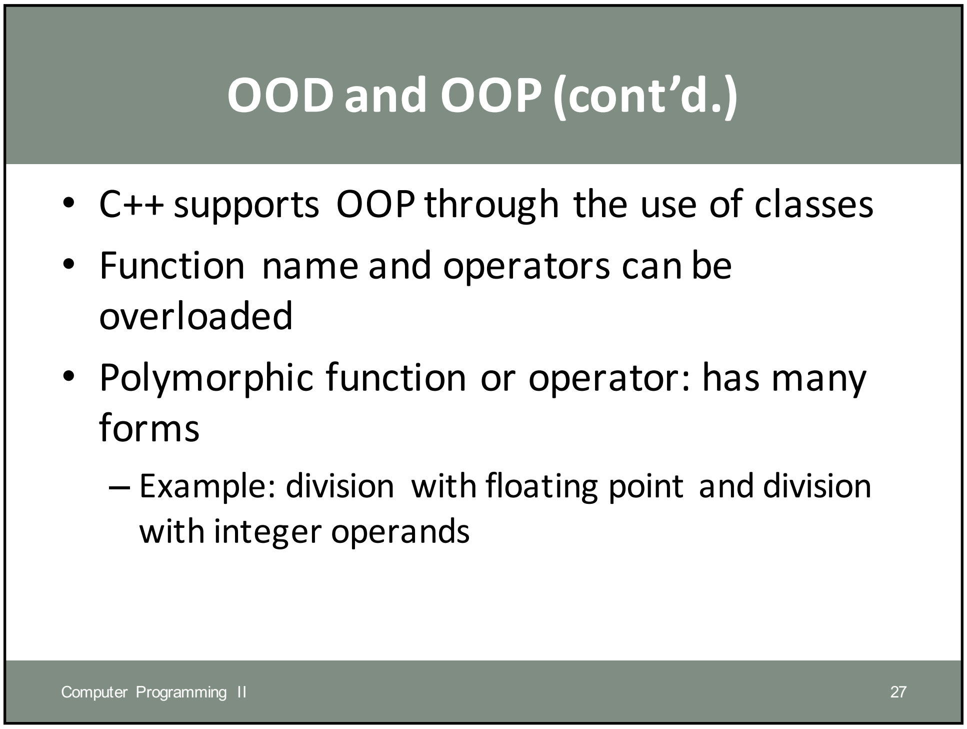
Object-Oriented Design (OOD) and Object-Oriented Programming (OOP)

- The fundamental principles of object-oriented design (OOD) are:
  - Encapsulation: combines data and operations on data in a single unit
  - Inheritance: creates new objects (classes) from existing objects (classes)
  - Polymorphism: the ability to use the same expression to denote different operations



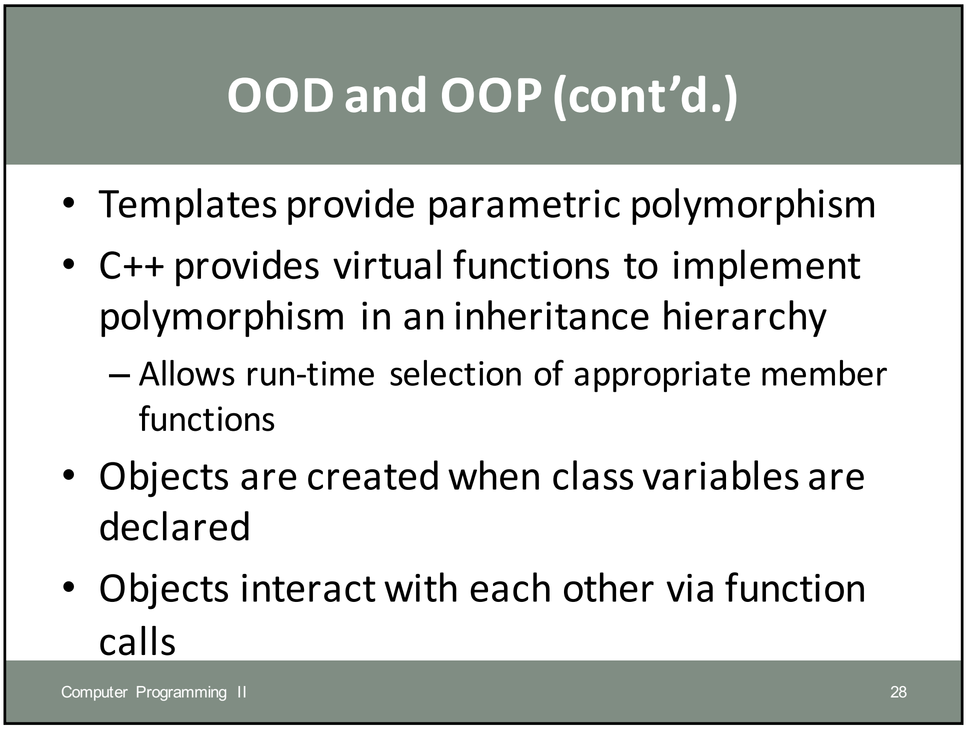
OOD and OOP (cont'd.)

- In OOD:
  - Object is a fundamental entity
  - Debug at the class level
  - A program is a collection of interacting objects
- OOD encourages code reuse
- Object-oriented programming (OOP) implements OOD



OOD and OOP (cont'd.)

- C++ supports OOP through the use of classes
- Function name and operators can be overloaded
- Polymorphic function or operator: has many forms
  - Example: division with floating point and division with integer operands



OOD and OOP (cont'd.)

- Templates provide parametric polymorphism
- C++ provides virtual functions to implement polymorphism in an inheritance hierarchy
  - Allows run-time selection of appropriate member functions
- Objects are created when class variables are declared
- Objects interact with each other via function calls

OOD and OOP (cont’d.)

- Every object has an internal state and external state
  - Private members form the internal state
  - Public members form the external state
- Only the object can manipulate its internal state

Identifying Classes, Objects, and Operations

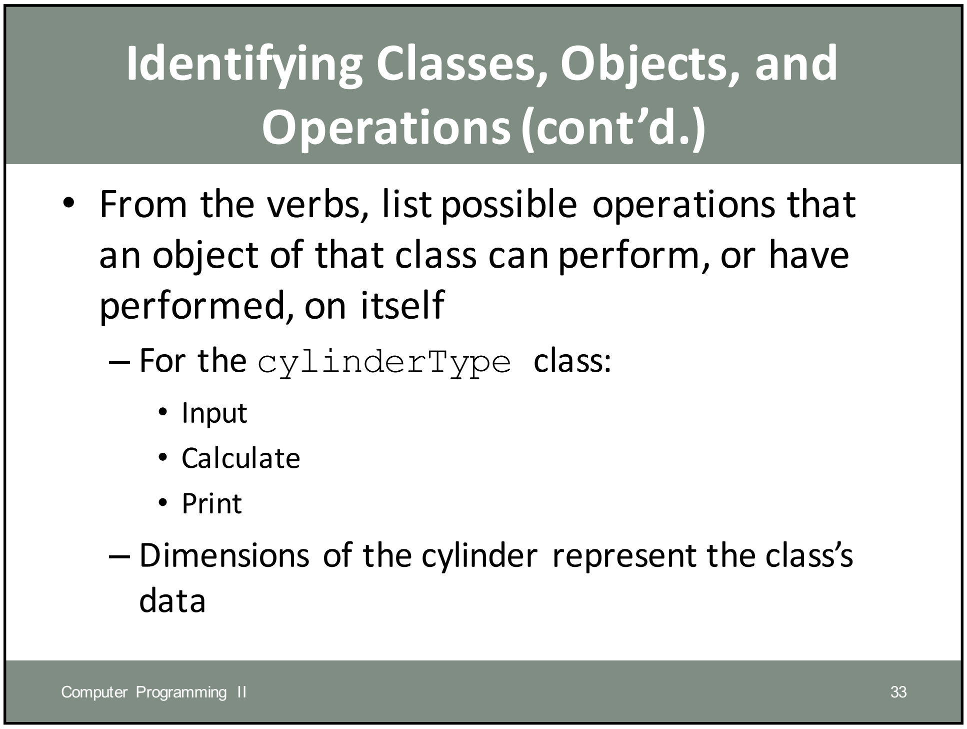
- To find classes: begin with a problem description and identify all nouns and verbs
  - From the list of nouns choose the classes
  - From the list of verbs choose the operations
- Suppose we want to write a program that calculates and prints the volume and surface area of a cylinder

Identifying Classes, Objects, and Operations (cont'd.)

- State this problem as follows:
  - *Write* a **program** to *input* the **dimensions** of a **cylinder** and *calculate* and *print* the **surface area** and **volume**
  - Nouns are bold and verbs are italic
  - From the list of nouns, can visualize a cylinder as a class (`cylinderType`) from which we can create many cylinder objects of various dimensions

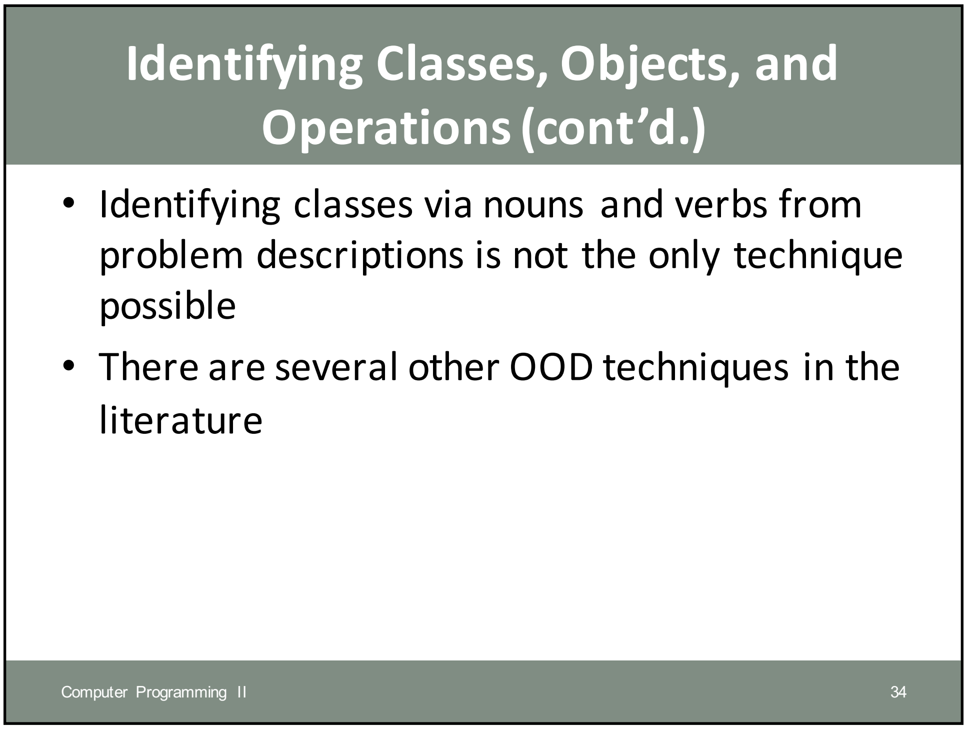
Identifying Classes, Objects, and Operations (cont'd.)

- These nouns are characteristics of a cylinder, so they will not be classes:
  - Dimensions
  - Surface area
  - Volume
- Next, determine three pieces of information about this class:
  - Operations that an object can perform
  - Operations that can be performed on an object
  - Information that an object must maintain



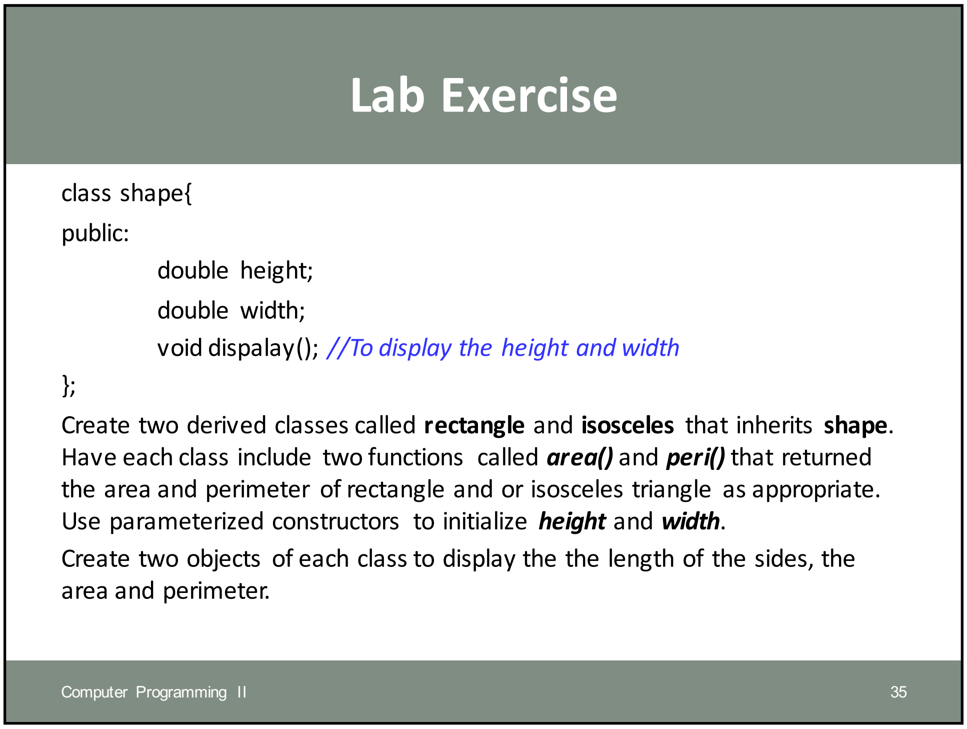
Identifying Classes, Objects, and Operations (cont’d.)

- From the verbs, list possible operations that an object of that class can perform, or have performed, on itself
  - For the `cylinderType` class:
    - Input
    - Calculate
    - Print
  - Dimensions of the cylinder represent the class’s data



Identifying Classes, Objects, and Operations (cont’d.)

- Identifying classes via nouns and verbs from problem descriptions is not the only technique possible
- There are several other OOD techniques in the literature

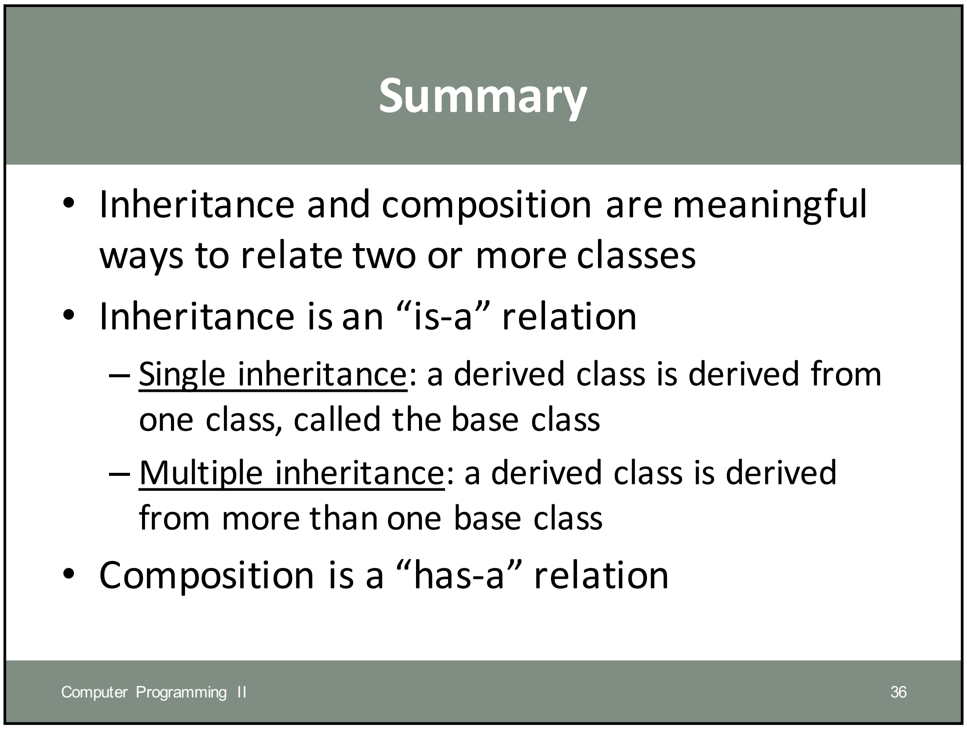


Lab Exercise

```
class shape{
public:
        double height;
        double width;
        void dispalay(); //To display the height and width
};
```

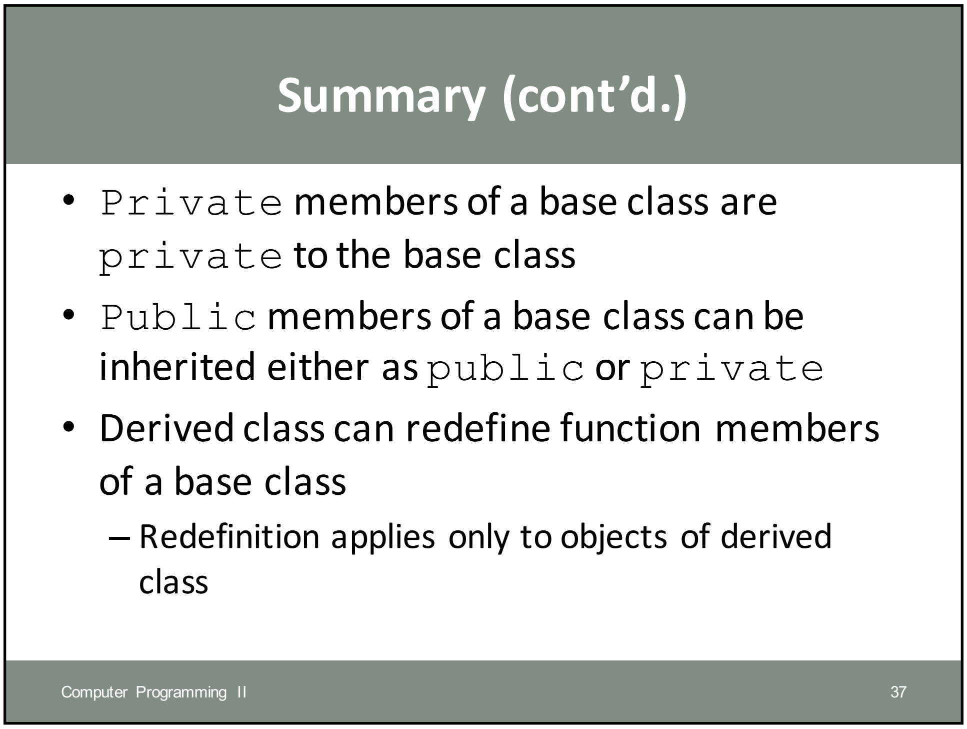
Create two derived classes called **rectangle** and **isosceles** that inherits **shape**. Have each class include two functions called ***area()*** and ***peri()*** that returned the area and perimeter of rectangle and or isosceles triangle as appropriate. Use parameterized constructors to initialize ***height*** and ***width***.

Create two objects of each class to display the the length of the sides, the area and perimeter.



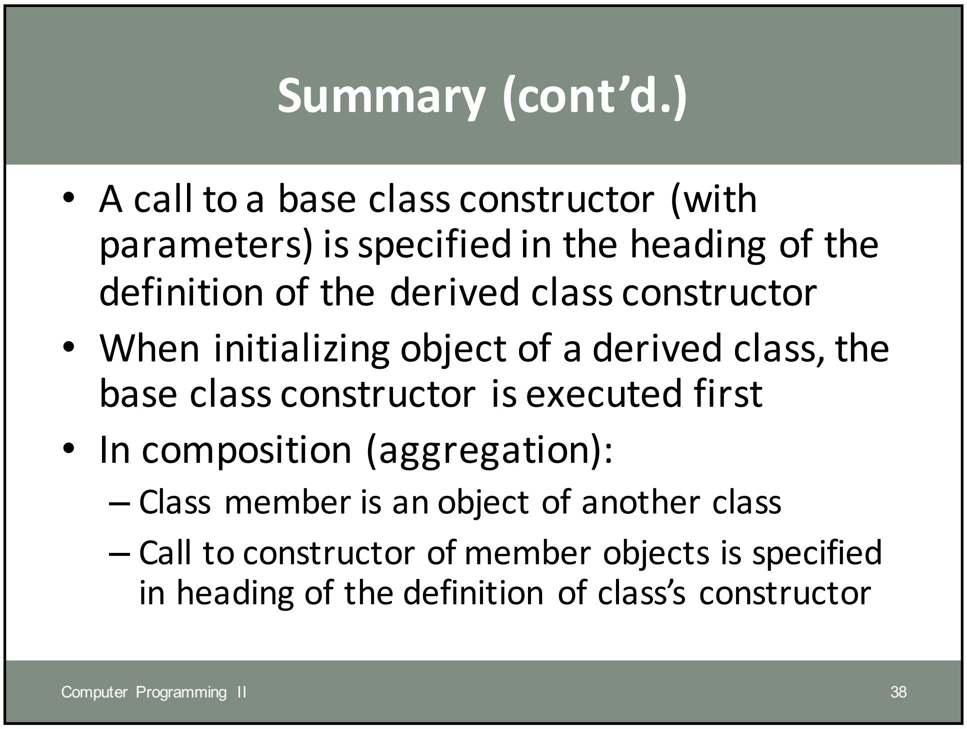
Summary

- Inheritance and composition are meaningful ways to relate two or more classes
- Inheritance is an “is-a” relation
  - Single inheritance: a derived class is derived from one class, called the base class
  - Multiple inheritance: a derived class is derived from more than one base class
- Composition is a “has-a” relation



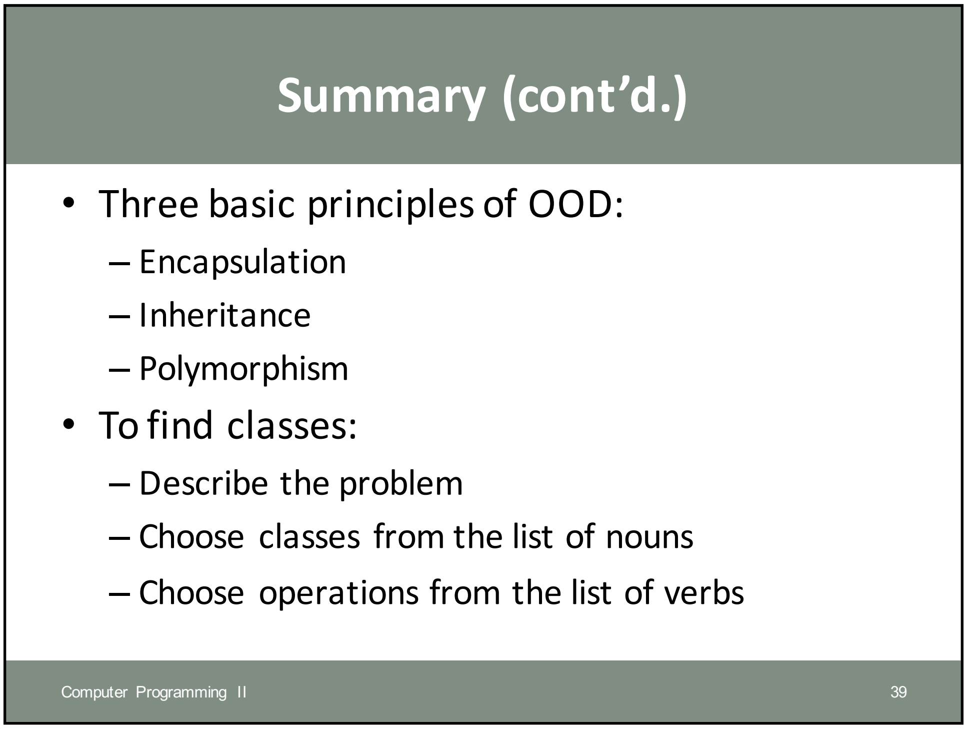
Summary (cont'd.)

- `Private` members of a base class are `private` to the base class
- `Public` members of a base class can be inherited either as `public` or `private`
- Derived class can redefine function members of a base class
  - Redefinition applies only to objects of derived class



Summary (cont'd.)

- A call to a base class constructor (with parameters) is specified in the heading of the definition of the derived class constructor
- When initializing object of a derived class, the base class constructor is executed first
- In composition (aggregation):
  - Class member is an object of another class
  - Call to constructor of member objects is specified in heading of the definition of class's constructor



Summary (cont’d.)

- Three basic principles of OOD:
  - Encapsulation
  - Inheritance
  - Polymorphism
- To find classes:
  - Describe the problem
  - Choose classes from the list of nouns
  - Choose operations from the list of verbs