Chapter 11 - Inheritance and Composition

Spring 2022

Objectives (1 of 2)

- 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 (2 of 2)

- In this chapter, you will:
  - 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 (1 of 5)

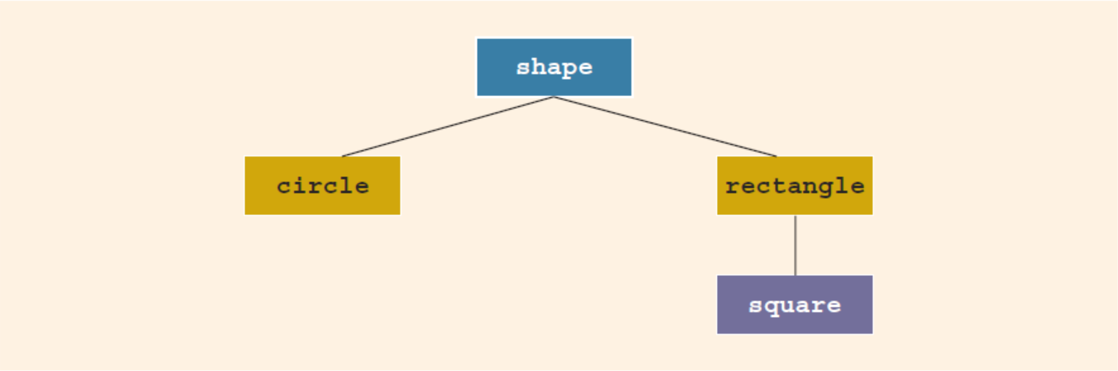
- Inheritance is an “is-a” relationship
  - Example: “every employee is a person”
- Inheritance allows creation of new classes from existing classes
  - **Derived classes**: new classes created from the existing classes
  - **Base class**: the original class
- A derived class inherits the properties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

Inheritance (2 of 5)

- Inheritance helps reduce software development 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 (3 of 5)

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



Inheritance hierarchy

Inheritance (4 of 5)

- Syntax of a derived class:

```
class className : memberAccessSecifier 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

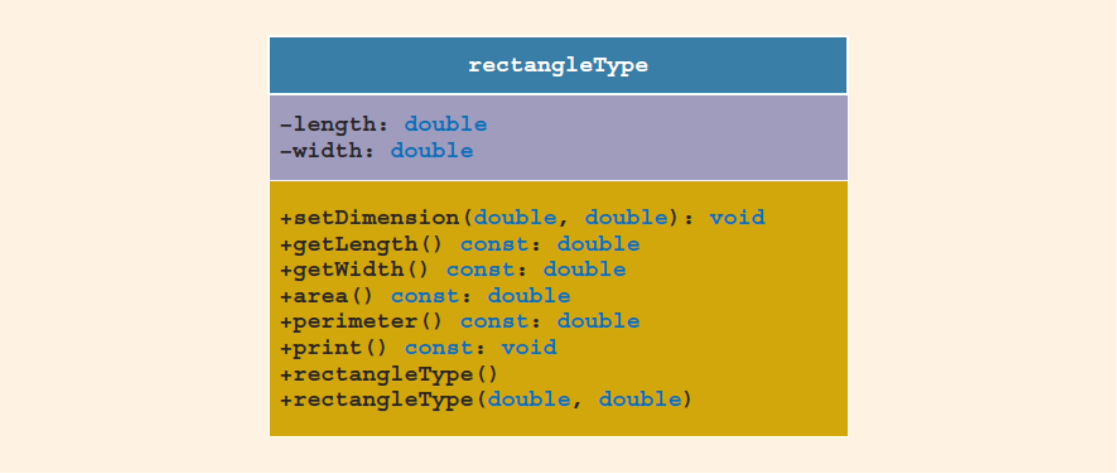
Inheritance (5 of 5)

- `public` members of the base class can be inherited as `public` or `private` members
- The derived class can include additional members (data and/or functions)
- The 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 (1 of 3)

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

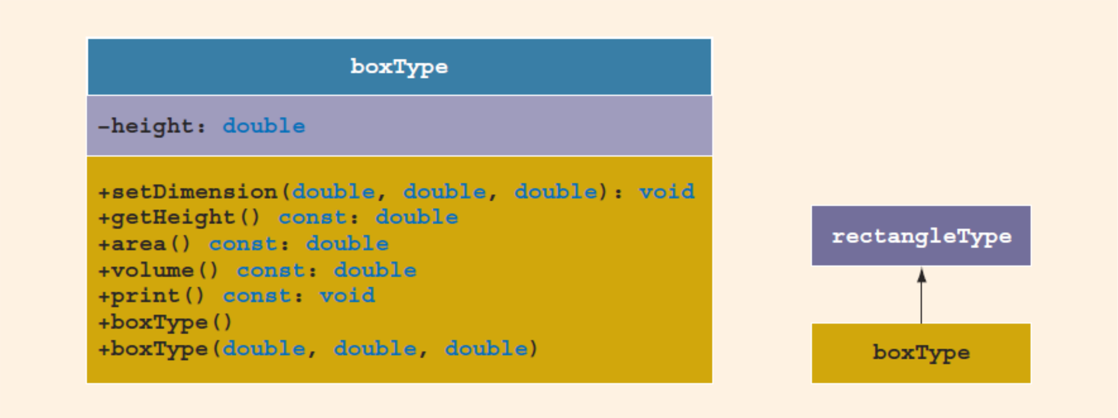
Redefining (Overriding) Member Functions of the Base Class (2 of 3)


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)

UML class diagram of the class rectangleType

Redefining (Overriding) Member Functions of the Base Class (3 of 3)

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



UML class diagram of the class boxType and the inheritance hierarchy

Constructors of Derived and Base Classes

- A derived class constructor cannot directly access private members of the base class
  - 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
- A call to the base class constructor is specified in the heading of the derived class constructor definition

Destructors in a Derived Class

- Destructors 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

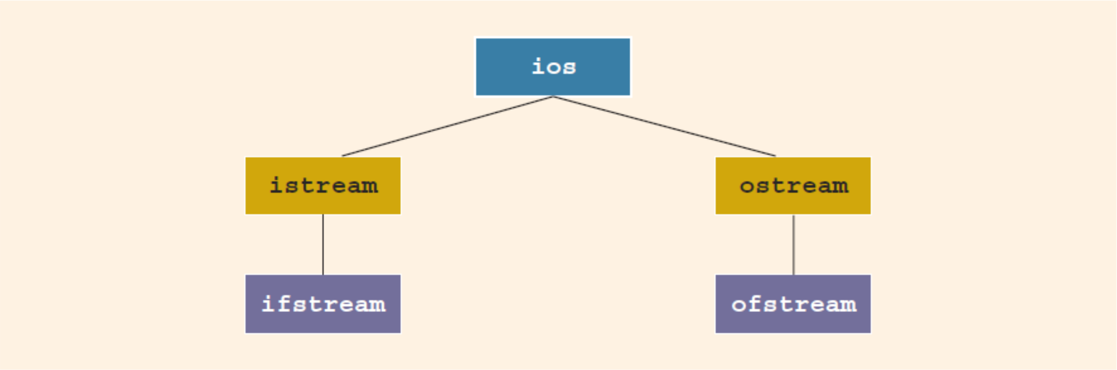
- To define new classes, create new header files
- To create new derived classes, include commands that specify where the base class definitions can be found
- Definitions of the member functions can be placed in a separate file

Multiple Inclusions of a Header File

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

C++ Stream Classes (1 of 2)

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



C++14 stream classes hierarchy

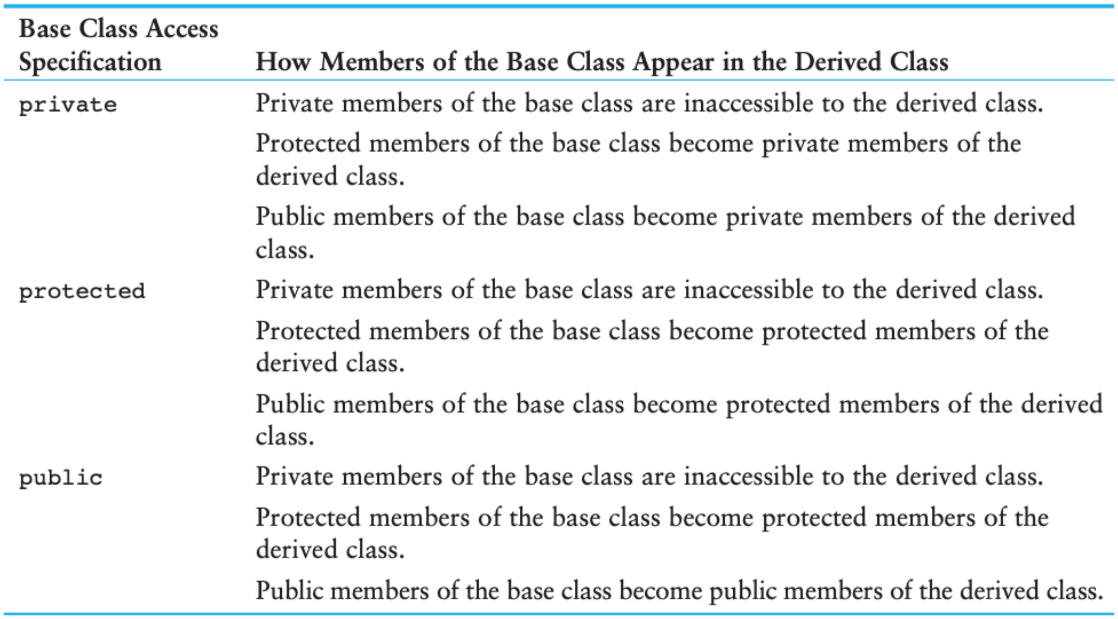
C++ Stream Classes (2 of 2)

- **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** and **ofstream** objects are for file I/O
    - Header file **fstream** contains the definitions for these

Protected Members of a Class

- A 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



Base Class Access Specification	How Members of the Base Class Appear in the Derived Class
`private`	Private members of the base class are inaccessible to the derived class.
	Protected members of the base class become private members of the derived class.
	Public members of the base class become private members of the derived class.
`protected`	Private members of the base class are inaccessible to the derived class.
	Protected members of the base class become protected members of the derived class.
	Public members of the base class become protected members of the derived class.
`public`	Private members of the base class are inaccessible to the derived class.
	Protected members of the base class become protected members of the derived class.
	Public members of the base class become public members of the derived class.

How Base Class Members are Inherited

Composition (Aggregation) (1 of 2)

- In composition, one or more member(s) of a class are objects of another class type
- Composition (aggregation) is a "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) (2 of 2)

- 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) (1 of 5)

- 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 (2 of 5)

- 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 (3 of 5)

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

OOD and OOP (4 of 5)

- 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 (5 of 5)

- Every object has an internal state and an 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 (1 of 5)

- 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 (2 of 5)

- 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, one can visualize a cylinder as a class (**cylinderType**) from which we can create many cylinder objects of various dimensions

Identifying Classes, Objects, and Operations (3 of 5)

- 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 (4 of 5)

- 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 (5 of 5)

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

Quick Review (1 of 4)

- 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

Quick Review (2 of 4)

- **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
- A derived class can redefine function members of a base class
  - Redefinition applies only to objects of derived class

Quick Review (3 of 4)

- 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):
  - A class member is an object of another class
  - A call to constructor of member objects is specified in heading of the definition of class's constructor

Quick Review (4 of 4)

- 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

Questions?