W 9.1 - Inheritance

Outline 9.1 Introduction

- Inhoritance: Base Classes and Derived Classes Protected Members Casting Base-Class Pointers to Derived-Class Pointers
- 9.2 9.3 9.4 9.5 9.6 9.7
- Using Member Functions Overriding Base-Class Members in a Derived Class Public, Protected and Private Inheritance Direct Base Classes and Indirect Base Classes
- Using Constructors and Destructors in Derived Classes Implicit Derived-Class Object to Base-Class Object Conversion Software Engineering with Inheritance
- 9.8 9.9 9.10 9.11
- 9.12
- Composition vs. Inheritance "Uses A" and "Knows A" Relationships Case Study: Point, Circle, Cylinder 9.12 9.13 9.14
- 9.15 **Multiple Inheritance**

© 2000 Prentice Hall, Inc. All rights reserved.

9.1 Introduction

Inheritance

1

- New classes created from existing classes
- Absorb attributes and behaviors
- Derived class
 - · Class that inherits data members and member functions from a previously defined base class

2

- Single inheritance
- · Class inherits from one base class
- Multiple inheritance · Class inherits from multiple base classes
- Types of inheritance

 - public: Derived objects are accessible by the base class objects · private: Derived objects are inaccessible by the base class

 - protected: Derived classes and friends can access protected members of the base class

©2000 Prentice Hall, Inc. All rights reserved.

9.1 Introduction

· Polymorphism

© 2000 Prentice Hall, Inc. All rights reserved.

- Write programs in a general fashion
- Handle a wide variety of existing (and unspecified) related classes

9.2 Inheritance: Base and Derived Classes

· Base and derived classes

- Often an object from a derived class (subclass) is also an object of a base class (superclass)
 - A rectangle is a derived class in reference to a quadrilateral and a base class in reference to a square
- Inheritance examples

| Base class | Derived classes |
|------------|--|
| Student | GraduateStudent UndergraduateStudent |
| Shape | Circle Triangle Rectangle |
| Loan | CarLoan HomeImprovementLoan MortgageLoan |
| Employee | FacultyMember StaffMember |
| Account | CheckingAccount SavingsAccount |

©2000 Prentice Hall, Inc. All rights reserved.

5

9.2 Inheritance: Base and Derived Classes

- Implementation of **public** inheritance
 - class CommissionWorker : public Employee { . . .
 - };
 - Class CommissionWorker inherits from class
 - Employee
 - friend functions not inherited
 - private members of base class not accessible from derived class

9.3 protected Members

• protected access

- Intermediate level of protection between public and private inheritance
- Derived-class members can refer to public and protected members of the base class simply by using the
- member names
- Note that protected data "breaks" encapsulation

© 2000 Prentice Hall, Inc. All rights reserved.

© 2000 Prentice Hall, Inc. All rights reserved.

9.4 Casting Base-Class Pointers to Derived **Class Pointers**

- · Derived classes relationships to base classes
 - Objects of a derived class can be treated as objects of the base class
 - · Reverse not true base class objects cannot be derived-class objects
- Downcasting a pointer
 - Use an explicit cast to convert a base-class pointer to a derivedclass pointer
 - If pointer is going to be dereferenced, the type of the pointer must match the type of object to which the pointer points
 - Format:

derivedPtr = static_cast< DerivedClass * > basePtr;

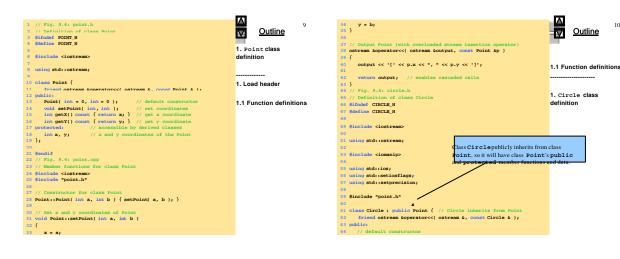
© 2000 Prentice Hall, Inc. All rights reserved.

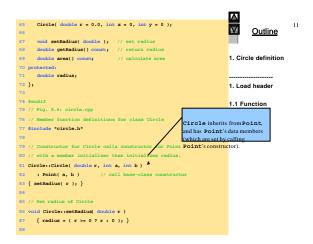
9.4 Casting Base-Class Pointers to Derived-**Class Pointers**

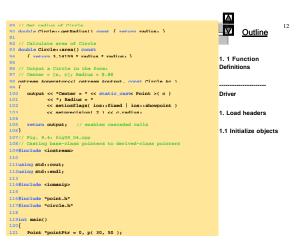
• The following example:

©2000 Prentice Hall, Inc. All rights reserved.

- Demonstrates the casting of base class pointers to derived class pointers
- Class Circle is derived from class Point
- A pointer of type Point is used to reference a Circle object, and a pointer to type Circle is used to reference a Point object







| 122 Circle *circlePtr = 0, c(2.7, 120 | 0, 89); 13 | | | | |
|--|--|--|--|--|--|
| 123 | Outline | | | | |
| 124 cout << "Point p: " << p << "\nCirc | cale c: " << c << '\n'; | | | | |
| 125 Poin | nt p: [30, 50] | | | | |
| 126 // Treat a Circle as a Point Circ | cle c: Center = [120, 89]; Radius = 2.70 objects | | | | |
| <pre>127 pointPtr = &c // assign address</pre> | | | | | |
| 128 cout << "\nCircle o (via *pointPtr | r): " <u>1.2</u> Assign objects | | | | |
| 129 << *pointPtr << '\n. | Circle c (via *pointPtr): [120, 89] | | | | |
| 130 | 2. Function calls | | | | |
| | in ome casting) | | | | |
| | ved-class pointer | | | | |
| | >(pointPtr); | | | | |
| 13 When it is cast to a Circle *, | tr):\n" << * airclePer | | | | |
| 13: circlePtr is really assigned to a | T Circle c (via *circlePtr): | | | | |
| 130 base-class object with no derived- | base-class object with no derived- ; Center = [120, 89]; Radius = 2.70 | | | | |
| 13 class information. This is dangerous. Area of c (via circlePtr): 22.90 | | | | | |
| 138 // DANGEROUS: Theat a Point as a Circle The base class pointer only | | | | | |
| 139 pointPtr = &p X assign address | s of Point to pointPtr "sees" the base-class part of the object it points to. | | | | |
| 140 | of the object it points to. | | | | |
| 141 // cast base-class pointer to derived-class pointer Cast pointPtr into a | | | | | |
| 42 circlePtr = static_cast< Circle * >(pointPtr); Circle *, and assign to | | | | | |
| 143 cout << "\nPoint p (via *circlePtr):\n" << *circlePtr circlePtr. | | | | | |
| 144 << "\nArea of object circlePtr pointe ter " | | | | | |
| <pre>145 << circlePtr->area() << endl;</pre> | | | | | |
| 146 return 0; | Center = [30, 50]; Radius = 0.00 | | | | |
| 147} | Area of object circlePtr points to: 0.00 | | | | |



©2000 Prentice Hall, Inc. All rights reserved.

©2000 Prentice Hall, Inc. All rights reserved.

15

Using Member Functions

Derived class member functions

- Cannot directly access private members of their base
 - class
 - Maintains encapsulation

9.5

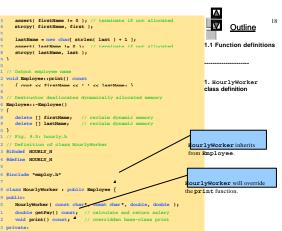
© 2000 Prentice Hall, Inc. All rights reserved.

 Hiding private members is a huge help in testing, debugging and correctly modifying systems

9.6 Overriding Base-Class Members in a Derived Class

- To override a base-class member function
 - In the derived class, supply a new version of that function with the same signature
 same function name, different definition
 - When the function is then mentioned by name in the derived class, the derived version is automatically called
 - The scope-resolution operator may be used to access the base class version from the derived class

| | Π | |
|---|--------------------------|---------------|
| 1 // Fig. 9.5: employ.b | 17 | 33 |
| 2 // Definition of class Employee | | 34 |
| 3 #ifndef EMPLOY_H | <u>o duino</u> | 35 |
| 4 #define EMPLOY_H | | 36 |
| 5 | 1. Employee class | 37 |
| 6 class Employee { 7 public: | definition | 38 |
| 8 Employee(const char *, const char *); // constructor | | 30 3 |
| 9 woid print() const: // output first and last name | | 40 41 // |
| 10 ~Employee(); // destructor | | 42 VO |
| 11 private: | 1. Load header | 43 |
| <pre>12 char *firstName; // dynamically allocated string</pre> | | 44 |
| 13 char *lastName: // dvnamically allocated string | 1.1 Function definitions | 45 // |
| 14 }; 15 | | 46 Emg |
| 15 16 #endif | | 47 { |
| 17 // Fig. 9.5: employ.cop | | 48 |
| 18 // Member function definitions for class Employee | | 49 |
| 19 #include <iostream></iostream> | | 50 } |
| 20 | | 51 // |
| 21 using std::cout: | | 52 // |
| 22 | | 53 #11 |
| 23 #include <gstring></gstring> | | 54 #de |
| 24 #include <cassert></cassert> | | 55 |
| 25 #include "employ.h" | | 56 #ir |
| 26 | | 57 |
| 27 // Constructor dynamically allocates space for the | | 58 cla |
| 28 // first and last name and uses stropy to copy | | 59 pub |
| 29 // the first and last names into the object. | | 60 |
| 30 Employee::Employee(const char *first, const char *last) | | 61 |
| 31 { | | 62 |
| <pre>31) 32 firstName = new char[strlen(first) + 1];</pre> | | 63 pri |
| | | to pro |





21

23

9.7 public, private, and protected Inheritance

| Base class member | Type of inheritance | | | |
|----------------------|--|--|---|--|
| access | public | protected | private | |
| specifier | inheritance | inheritance | inheritance | |
| Public | <pre>public in derived class. Can be accessed directly by any non-static member functions, friend functions and non- member functions.</pre> | protected in derived class. Can be accessed directly by all non-static member functions and friend functions. | privatein derived class. Can be accessed directly by all non-static member functions and friend functions. | |
| Protected | protected in derived class. | protected in derived class. | privatein derived class. | |
| | Can be accessed directly by all | Can be accessed directly by all | Can be accesseddirectly by all | |
| | non-static member functions | non-static member functions | non-static member functions | |
| | and friend functions. | and friend functions. | and friend functions. | |
| Private | Hidden in derived class. | Hidden in derived class. | Hidden in derived class. | |
| | Can be accessed by non-static | Can be accessed by non-static | Can be accessed by non-static | |
| | member functions and friend | member functions and <u>friend</u> | member functions and friend | |
| | functions through public or | functions through public or | functions through public or | |
| | protected member functions | protected member functions | protected member functions | |
| | of the base class. | of the base class. | of the base class. | |

© 2000 Prentice Hall, Inc. All rights reserved.

9.9 Using Constructors and Destructors in Derived Classes

- Base class initializer
 - Uses member-initializer syntax
 - Can be provided in the derived class constructor to call the base-class constructor explicitly
 - Otherwise base class's default constructor called implicitly
 Base-class constructors and base-class assignment operators
 - are not inherited by derived classes
 - Derived-class constructors and assignment operators, however, can call base-class constructors and assignment operators



9.8 Direct and Indirect Base Classes

22

24

· Direct base class

- Explicitly listed derived class's header with the colon (:) notation when that derived class is declared
 class HourlyWorker : public Employee
 Employee is a direct base class of HourlyWorker
- Indirect base class
 - Not listed in derived class's header
 - Inherited from two or more levels up the class hierarchy class MinuteWorker : public HourlyWorker
 Employee is an indirect base class of MinuteWorker

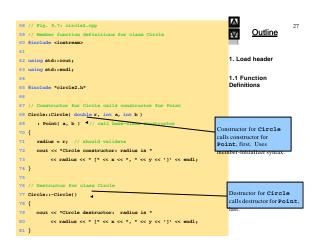
©2000 Prentice Hall, Inc. All rights reserved.

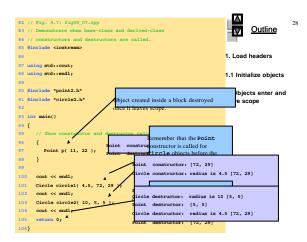
9.9 Using Constructors and Destructors in Derived Classes

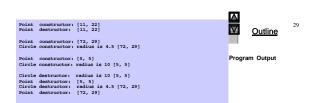
- · A derived-class constructor
 - Calls the constructor for its base class first to initialize its base-class members
 - If the derived-class constructor is omitted, its default constructor calls the base-class' default constructor
- Destructors are called in the reverse order of constructor calls
- So a derived-class destructor is called before its base-class destructor
- Destructor of Class Employee frees up the dynamically assigned arrays[]firstname and[]lastname, avoiding memory leaks (lines 48, 49).

©2000 Prentice Hall, Inc. All rights reserved.

| <pre>1 // Fig. 9.7: point2.h 2 // Definition of class Point 3 % if note FOINT2_H 4 define FOINT2_H 5 6 class Point { 7 public: 8 Point(int = 0, int = 0); // default constructor 9 -Point(); // destructor 10 protected: // accessible by derived classes 11 int x, y; // x and y coordinates of Point 12 }; 13 accessifie 14 accessifie 15 accessifie 15 accessifie 16 accessifie 17 accessifie 18 accessifie 19 accessifie 19 accessifie 19 accessifie 10 accessifie 11 accessifie 11 accessifie 11 accessifie 12 accessifie 13 accessifie 14 accessifie 15 acce</pre> | 25 1. Point definition 1. Load header 1.1 Function definitions | <pre>33 34 // Destructor for class Point 35 Point::-Point() 36 { 37 cout << *Point destructor: * 38</pre> | 26 20 1.1 Function definitions 1.1 Load header 1.1 Circle Definition |
|--|---|---|--|
| <pre>33 34 #endif 35 // Fig. 9.7: point2.cpp 15 // Kenber Function definitions for class Point 17 #include <logtream> 28 39 20 using std:recut; 20 using std:recut; 20 using std:recut; 21 #include "point2.h" 22 #include "point2.h" 23 // Constructor for class Point 25 Point:rboint(int e, int b) 26 (27 / x = e; </logtream></pre> | | | |
| <pre>23 y = b; 23 30 cout << "Point constructor; " 31 c< '[' << x << ', * << y << ']' << endl; 32 }</pre> | | 55 }; 56 57 fendi£ | |







9.10 Implicit Derived-Class Object to Base-Class Object Conversion

• Assignment of derived and base classes

- Derived-class type and base-class type are different
 Derived-class object can be treated as a base-class object
- Derived-class object can be included as a base-class object
 Derived class has members corresponding to all of the base class's members
- Derived-class has more members than the base-class objectBase-class can be assigned a derived-class
- Base-class object cannot be treated as a derived-class object
 Would leave additional derived class members undefined
 - Derived-class cannot be assigned a base-class
 - Assignment operator can be overloaded to allow such an assignment

©2000 Prentice Hall, Inc. All rights reserved.

9.10 Implicit Derived-Class Object to Base-**Class Object Conversion**

31

33

35

- · Mixing base and derived class pointers and objects - Referring to a base-class object with a base-class pointer
 - · Allowed (straightforward) - Referring to a derived-class object with a derived-class pointer
 - Allowed (straightforward) - Referring to a derived-class object with a base-class pointer
 - · Possible syntax error · Code can only refer to base-class members, or syntax error
 - Referring to a base-class object with a derived-class pointer Svntax error
 - The derived-class pointer must first be cast to a base-class pointer
- · Need way to resolve base-class Vs derived-class routines using base-class pointers (Virtual functions)

© 2000 Prentice Hall, Inc. All rights reserved.

9.11 Software Engineering With Inheritance

32

34

- · Classes are often closely related
 - "Factor out" common attributes and behaviors and place these in a base class - Use inheritance to form derived classes
- · Modifications to a base class
 - Derived classes do not change as long as the public and protected interfaces are the same
 - Derived classes may need to be recompiled
- Use Inheritance sparingly, often times complexity is introduced needlessly. Can make for bad engineering thus hard to understand systems.

©2000 Prentice Hall, Inc. All rights reserved.

9.12 Composition vs. Inheritance

• "Is a" relationships

- Inheritance
- · Relationship in which a class is derived from another class
- "Has a" relationships
 - Composition

© 2000 Prentice Hall, Inc. All rights reserved.

- · Relationship in which a class contains other classes as members
- Has a, is a composition. Is a kind of, is inheritance.
- Interchangeable! Careful.

9.13 "Uses A" And "Knows A" Relationships

• "Uses a"

- One object issues a function call to a member function of another object. Limited!
- "Knows a"

© 2000 Prentice Hall, Inc. All rights reserved.

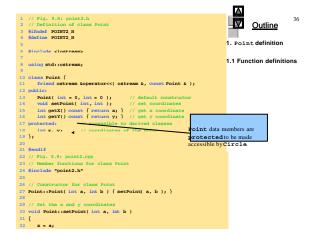
- One object is aware of another
 - · Contains a pointer or handle to another object
- Has access to all public stuff. - Also called an association

| 9.14 | Case St | udy: Po | oint, Circ | le, Cylind | er |
|------|---------|---------|------------|------------|----|

- · Point, circle, cylinder hierarchy
- Point class is base class

© 2000 Prentice Hall, Inc. All rights reserved.

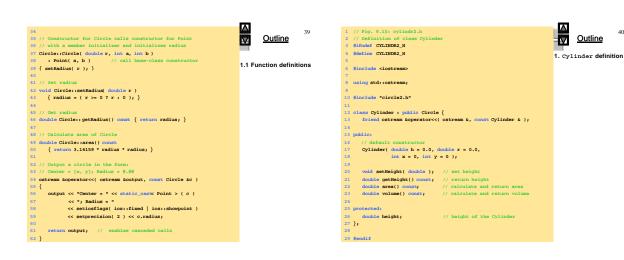
- Circle class is derived from Point class
- Cylinder class is derived from Circle class

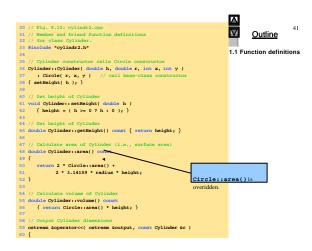


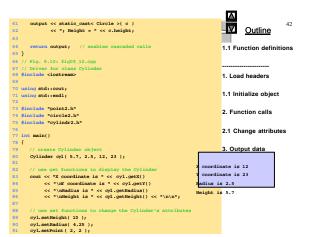
| 33 | y = b; | 37 |
|----|--|--------------------------|
| 34 | } | Outline |
| 35 | | ∇ |
| 36 | // Output the Point | 1.1 Function definitions |
| 37 | ostream &operator<<(ostream &output, const Point &p) | |
| 38 | { | |
| 39 | output << '[' << p.x << ", " << p.y << ']'; | |
| 40 | | |
| 41 | return output; // enables cascading | |
| 42 | } | |

| | // Fig. 9.9: circle2.h // Definition of class Circle #inder CIRCLE2 H Martine CIRCLE2 H Einclus circle2 H | 1 | Out | tline | 38 |
|----------------------------------|--|------|--|-------------|----|
| 7 8 9 10 | using std::ostream; #include "point2.h" | 1 | .1 Function | n definitio | ns |
| 13 14 15 16 17 18 | <pre>class Circle : while Point { friend ostream &perstor<(ostream &, const Circle &); public; // default constructor Circle(double r = 0.0, int x = 0, int y = 0); void setRadius(double); // return radius double getRadius() const; // return radius</pre> | | | | |
| 21 22 23 24 | double radius: // accessible to derived classes double radius: // radius of the Circle }; | ecte | ta members a dto be made yCylinder | | |
| 26 27 28 | // Member function definitions for class Circle #include <iomanip></iomanip> | | | | |
| 31 32 | using std::setiosflags; using std::setprecision; Binclude "circle2.h" | | | | |

© 2000 Prentice Hall, Inc. All rights reserved.







| <pre>92 cout << "The new location, radius, and height of cyl are:\n" 93 << cyl << '\n'; 94 Outline</pre> | | | | | | |
|--|--|--|--|--|--|--|
| 94 | | The new location, radius, and height of cyl | | | | |
| 95 | cout << "The area of cyl is:\n" | are: | | | | |
| 96 | << cyl.area() << '\n'; | Center = [2, 2]; Radius = 4.25; Height = 10.00 | | | | |
| 97 98 | | The area of cyl is: | | | | |
| 98 | <pre>// display the Cylinder as a Point Point &pRef = cyl; // pRef "thinks</pre> | 380.53 | | | | |
| 100 | cout << "\nCylinder printed as a Poi | | | | | |
| 101 | << pRef << "\n\n". | | | | | |
| 102 | | Cylinder printed as a Point is: [2, 2] | | | | |
| 103 | // display the Cylinder as a Circle | | | | | |
| 104 | Circle &circleRef = cyl; // circleR | | | | | |
| 105 | cout << "Cylinder printed as a Circl | e is:\n | | | | |
| 106 | << "\nArea: " << OircleRef.area | Cylinder printed as a Circle is: | | | | |
| 107 | | Center = [2, 2]; Radius = 4.25 | | | | |
| 108 | return 0; | | | | | |
| 103} | | Area: 56.74 | | | | |
| X coordinate is 12 Circle, so it prints as one. | | | | | | |
| | rdinate is 23 | | | | | |
| | s is 2.5 | | | | | |
| Height is 5.7 | | | | | | |
| The new location, radius, and height of cyl are: | | | | | | |
| Center = [2, 2]; Radius = 4.25; Height = 10.00 | | | | | | |
| The area of cyl is: 380.53 | | | | | | |
| Cylinder printed as a Point is: [2, 2] | | | | | | |
| Cylinder printed as a Circle is: | | | | | | |
| Center = [2, 2]; Radius = 4.25 | | | | | | |
| Area: 56.74 | | | | | | |
| | | | | | | |

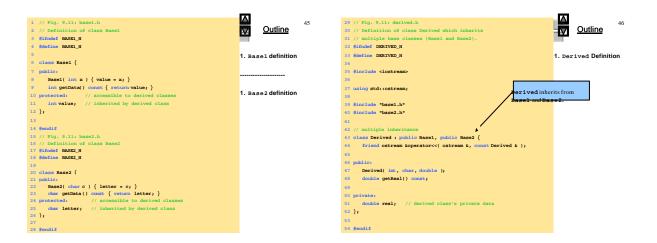
9.15 Multiple Inheritance

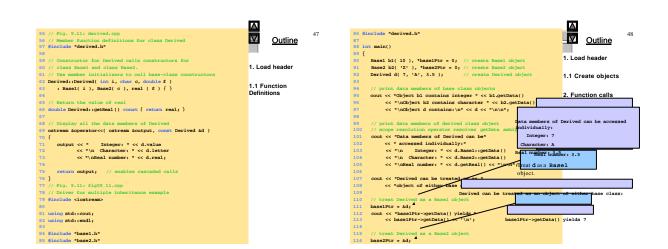
44

• Multiple Inheritance

©2000 Prentice Hall, Inc. All rights reserved.

- Derived-class inherits from multiple base-classes
- Encourages software reuse, but can create ambiguities





| <pre>117 cout << "base2Ptr->getData() yields " 118 << base2Ptr->getData() << endl; 119</pre> | a() yields A Outline |
|--|----------------------|
| 120 return 0; 121} | 3. Output data |
| Object bl contains integer 10 Object b2 contains character Z Object d contains character Z Datager: Character: A Real number: 3.5 | Program Output |
| Data members of Derived can be accessed individually: Integer: 7 Character: A Real number: 3.5 Derived can be treated as an object of either base class: baselPt:-ysetData() yields 7 base2Ptr-ygetData() yields A | |

Graded Exercises

50

- Read the summary of Ch 9 (pp 618..622)
- Do Self-Review exercises Ch 9 Deitel & Deitel
- Do following Exercises

49

9.2 (make a diagram like on Fig 9.2, put in folder),
9.12 (on paper & put in folder)

© 2000 Prentice Hall, Inc. All rights reserved.

©2000 Prentice Hall, Inc. All rights reserved.