# 12

# Object-Oriented Programming: Inheritance



Say not you know another entirely, till you have divided an inheritance with him.

— Johann Kasper Lavater

This method is to define as the number of a class the class of all classes similar to the given class.

— Bertrand Russell

Good as it is to inherit a library, it is better to collect one.

— Augustine Birrell

Save base authority from others' books.

— William Shakespeare



## **OBJECTIVES**

In this chapter you will learn:

- To create classes by inheriting from existing classes.
- How inheritance promotes software reuse.
- The notions of base classes and derived classes and the relationships between them.
- The protected member access specifier.
- The use of constructors and destructors in inheritance hierarchies.
- The differences between public, protected and private inheritance.
- The use of inheritance to customize existing software.



| 12.1 | Introduction  |  |
|------|---|--|
| 12.2 | Base Classes and Derived Classes  |  |
| 12.3 | protected <b>Members</b>  |  |
| 12.4 | Relationship between Base Classes and Derived Classes   |  |
|      | 12.4.1 Creating and Using a CommissionEmployee Class  |  |
|      | 12.4.2 Creating a BasePlusCommissionEmployee Class Without Using Inheritance                    |  |
|      | 12.4.3 Creating a CommissionEmployee-BasePlusCommissionEmployee Inheritance Hierarchy           |  |
|      | 12.4.4 CommissionEmployee-BasePlusCommissionEmployee Inheritance Hierarchy Using protected Data |  |
|      | 12.4.5 CommissionEmployee-BasePlusCommissionEmployee Inheritance Hierarchy Using private Data   |  |
| 12.5 | Constructors and Destructors in Derived Classes   |  |
| 12.6 | public, protected and private Inheritance   |  |
| 12.7 | Software Engineering with Inheritance   |  |
| 12.8 | Wrap-Up   |  |



## 12.1 Introduction

#### • Inheritance

- Software reusability
- Create new class from existing class
  - Absorb existing class' s data and behaviors
  - Enhance with new capabilities
- Derived class inherits from base class
  - Derived class
    - More specialized group of objects
    - Behaviors inherited from base class
      - Can customize
    - Additional behaviors



# 12.1 Introduction (Cont.)

- Class hierarchy
  - Direct base class
    - Inherited explicitly (one level up hierarchy)
  - Indirect base class
    - Inherited two or more levels up hierarchy
  - Single inheritance
    - Inherits from one base class
  - Multiple inheritance
    - Inherits from multiple base classes
      - Base classes possibly unrelated
    - More details in chapter 24



# 12.1 Introduction (Cont.)

- Three types of inheritance
  - public
    - Every object of derived class is also an object of base class
      - Base-class objects are not objects of derived classes
      - Example: All cars are vehicles, but not all vehicles are cars
    - Can access non-private members of base class
      - To access private base-class members
        - Derived class must use inherited non-private member functions
  - private
    - Alternative to composition
    - Chapter 21
  - protected
    - Rarely used



# 12.1 Introduction (Cont.)

- Abstraction
  - Focus on commonalities among objects in system
- "is-a" vs. "has-a"
  - "is-a"
    - Inheritance
    - Derived class object can be treated as base class object
    - Example: Car is a vehicle
      - Vehicle properties/behaviors also apply to a car
  - "has-a"
    - Composition
    - Object contains one or more objects of other classes as members
    - Example: Car has a steering wheel



## **Software Engineering Observation 12.1**

Member functions of a derived class cannot directly access private members of the base class.



## **Software Engineering Observation 12.2**

If a derived class could access its base class' sprivate members, classes that inherit from that derived class could access that data as well. This would propagate access to what should be private data, and the benefits of information hiding would be lost.



## 12.2 Base Classes and Derived Classes

- Base classes and derived classes
  - Object of one class "is an" object of another class
    - Example: Rectangle is quadrilateral
      - Class Rectangle inherits from class Quadrilateral
        - Quadrilateral is the base class
        - Rectangle is the derived class
  - Base class typically represents larger set of objects than derived classes
    - Example:
      - Base class: Vehicle
        - Includes cars, trucks, boats, bicycles, etc.
      - Derived class: Car
        - Smaller, more-specific subset of vehicles



| Base class    | Derived classes  |
|---------------|--|
| Student       | GraduateStudent, UndergraduateStudent  |
| Shape<br>Loan | Circle, Triangle, Rectangle, Sphere, Cube CarLoan, HomeImprovementLoan, MortgageLoan |
| Employee      | Faculty, Staff   |
| Account       | CheckingAccount, SavingsAccount  |

Fig. 12.1 | Inheritance examples.



# 12.2 Base Classes and Derived Classes (Cont.)

- Inheritance hierarchy
  - Inheritance relationships: tree-like hierarchy structure
  - Each class becomes
    - Base class
      - Supplies data/behaviors to other classes

#### **OR**

- Derived class
  - Inherits data/behaviors from other classes

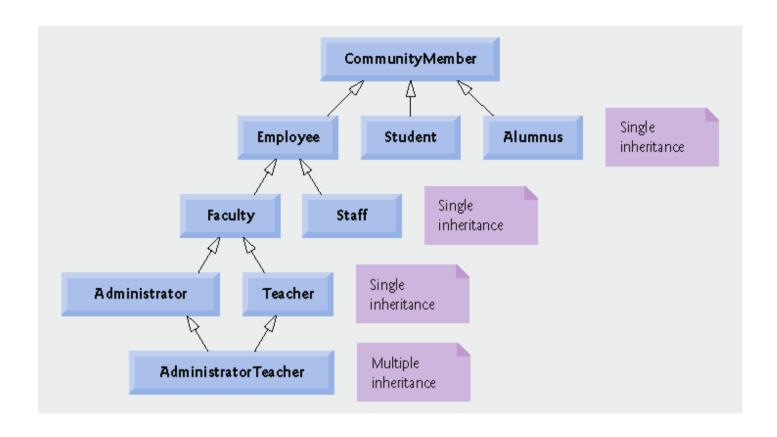


Fig. 12.2 | Inheritance hierarchy for university Community Members.



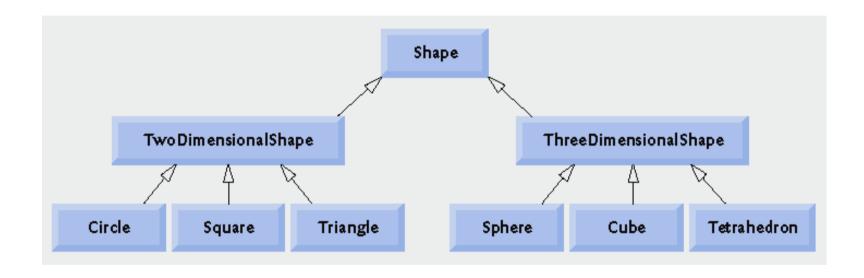


Fig. 12.3 | Inheritance hierarchy for Shapes.



# 12.2 Base Classes and Derived Classes (Cont.)

- public inheritance
  - Specify with:
    - Class TwoDimensionalShape : public Shape
      - Class TwoDimensional Shape inherits from class Shape
  - Base class private members
    - Not accessible directly
    - Still inherited
      - Manipulated through inherited public member functions
  - Base class public and protected members
    - Inherited with original member access
  - friend functions
    - Not inherited



# 12.3 protected Members

## protected access

- Intermediate level of protection between public and private
- protected members are accessible to
  - Base class members
  - Base class friends
  - Derived class members
  - Derived class friends

#### Derived-class members

- Refer to public and protected members of base class
  - Simply use member names
- Redefined base class members can be accessed by using base-class name and binary scope resolution operator (::)



# 12.4 Relationship between Base Classes and Derived Classes

- Base class and derived class relationship
  - Example: CommissionEmployee/
     BasePlusCommissionEmployee inheritance hierarchy
    - CommissionEmployee
      - First name, last name, SSN, commission rate, gross sale amount
    - BasePlusCommissionEmployee
      - First name, last name, SSN, commission rate, gross sale amount
      - And also: base salary



# 12.4.1 Creating and Using a CommissionEmployee Class

- Class CommissionEmployee
  - CommissionEmployee header file
    - Fig. 12.4
    - Specify public services
      - Constructor
      - get and set functions
      - Member functions earnings and print
  - CommissionEmployee source code file
    - Fig. 12.5
    - Specify member-function definitions

#### Outline

Commission

Employee.h

(1 of 2)

2 // CommissionEmployee class definition represents a commission employee. 3 #ifndef COMMISSION\_H #define COMMISSION\_H 5 #include <string> // C++ standard string class using std::string; 8 class CommissionEmployee 10 { 11 public: 12 CommissionEmployee( const string &, const string &, const string &, 13 double = 0.0, double = 0.0); Class CommissionEmployee constructor 14 void setFirstName( const string & ); // set first name 15 16 string getFirstName() const; // return first name 17 18 void setLastName( const string & ); // set last name string getLastName() const; // return last name 19 20 21 void setSocialSecurityNumber( const string & ); // set SSN string getSocialSecurityNumber() const; // return SSN 22 23 24 void setGrossSales( double ); // set gross sales amount 25 double getGrossSales() const; // return gross sales amount 26 void setCommissionRate( double ); // set commission rate (percentage) 27

double getCommissionRate() const; // return commission rate

1 // Fig. 12.4: CommissionEmployee.h

28



```
29
     double earnings() const; // calculate earnings
30
     void print() const; // print CommissionEmployee object
31
32 private:
     string firstName;
33
     string lastName;
                                                      Declare private
34
35
     string socialSecurityNumber;
                                                      data members
     double grossSales; // gross weekly sales
36
     double commissionRate; // commission percentage
37
38 }; // end class CommissionEmployee
39
40 #endif
```

#### Outline

Commission Employee.h

(2 of 2)

```
1 // Fig. 12.5: CommissionEmployee.cpp
2 // Class CommissionEmployee member-function definitions.
                                                                                      Outline
3 #include <iostream>
  using std::cout;
5
  #include "CommissionEmployee.h" // CommissionEmployee class definition
                                                                                     Commission
7
                                                                                      Employee.cpp
  // constructor
  CommissionEmployee::CommissionEmployee(
                                                                                     (1 \text{ of } 4)
10
      const string &first, const string &last, const string &ssn,
11
      double sales, double rate )
12 {
                                                                        Initialize data members
13
     firstName = first; // should validate
     lastName = last; // should validate
14
15
     socialSecurityNumber = ssn; // should validate
     setGrossSales( sales ); // validate and store gross sales
16
     setCommissionRate( rate ): // validate and store commission rate
17
18 } // end CommissionEmployee constructor
19
20 // set first name
21 void CommissionEmployee::setFirstName( const string &first )
22 {
      firstName = first; // should validate
23
24 } // end function setFirstName
25
26 // return first name
27 string CommissionEmployee::getFirstName() const
28
     return firstName;
29
30 } // end function getFirstName
```



```
31
32 // set last name
                                                                                      Outline
33 void CommissionEmployee::setLastName( const string &last )
34 {
     lastName = last; // should validate
35
36 } // end function setLastName
                                                                                     Commission
37
                                                                                      Employee.cpp
38 // return last name
39 string CommissionEmployee::getLastName() const
                                                                                     (2 \text{ of } 4)
40 {
     return lastName:
41
42 } // end function getLastName
43
44 // set social security number
45 void CommissionEmployee::setSocialSecurityNumber( const string &ssn )
46 {
     socialSecurityNumber = ssn; // should validate
47
48 } // end function setSocialSecurityNumber
49
50 // return social security number
51 string CommissionEmployee::getSocialSecurityNumber() const
52 {
     return socialSecurityNumber;
53
                                                        Function setGrossSales
54 } // end function getSocialSecurityNumber
                                                         validates gross sales amount
55
56 // set gross sales amount
57 void CommissionEmployee::setGrossSales( double sales )
58 {
     grossSales = ( sales < 0.0 ) ? 0.0 : sales;
```

60 } // end function setGrossSales



срр

```
61
62 // return gross sales amount
                                                                                      Outline
63 double CommissionEmployee::getGrossSales() const
64 {
      return grossSales;
65
                                                             Function setCommissionRate
66 } // end function getGrossSales
                                                             validates commission rate
67
68 // set commission rate
                                                                                      (3 \text{ of } 4)
69 void CommissionEmployee::setCommissionRate( double rate )
70 {
     commissionRate = ( rate > 0.0 \&\& rate < 1.0 ) ? rate : 0.0;
71
72 } // end function setCommissionRate
73
74 // return commission rate
75 double CommissionEmployee::getCommissionRate() const
76 [
77
     return commissionRate;
78 } // end function getCommissionRate
```

```
79
80 // calculate earnings
                                                                                       Outline
81 double CommissionEmployee::earnings() const
                                                                  Function earnings
82 {
                                                                   calculates earnings
83
      return commissionRate * grossSales;
                                                                                       Commission
84 } // end function earnings
                                                                                       Employee.cpp
85
86 // print CommissionEmployee object
                                                                                       (4 \text{ of } 4)
87 void CommissionEmployee::print() const
                                                                Function print displays
88 {
                                                                 CommissionEmployee object
89
     cout << "commission employee: " << firstName << ' ' << l</pre>
         << "\nsocial security number: " << socialSecurityNumber</pre>
90
         << "\ngross sales: " << grossSales
91
         << "\ncommission rate: " << commissionRate;</pre>
92
93 } // end function print
```

```
1 // Fig. 12.6: fig12_06.cpp
2 // Testing class CommissionEmployee.
                                                                                          Outline
3 #include <iostream>
4 using std::cout:
5 using std::endl;
  using std::fixed;
                                                                                          fig12_06.cpp
7
  #include <iomanip>
                                                                                          (1 \text{ of } 2)
  using std::setprecision;
10
11 #include "CommissionEmployee.h" // CommissionEmployee class definition
12
13 int main()
14 {
                                                                Instantiate CommissionEmployee object
      // instantiate a CommissionEmployee object
15
      CommissionEmployee employee(
16
17
         "Sue" "Jones" "222-22-2222" 10000 .06 ):
18
      // set floating-point output formatting
19
20
      cout << fixed << setprecision( 2 );</pre>
21
22
      // get commission employee data
      cout << "Employee information obtained by get functions: \n"</pre>
23
         << "\nFirst name is " << employee.getFirstName()</pre>
24
         << "\nLast name is " << employee.getLastName()</pre>
                                                                       Use
25
         << "\nSocial security number is "</pre>
26
                                                                       CommissionEmployee's
         << employee.getSocialSecurityNumber()</pre>
27
                                                                       get functions to retrieve the
         << "\nGross sales is " << employee.getGrossSales()</pre>
28
                                                                       object's instance variable << end1; values
         << "\nCommission rate is " << employee.getCommissionRate()</pre>
29
```

```
30
31
      employee.setGrossSales( 8000 ); // set gross sales
                                                                                      Outline
32
      employee.setCommissionRate( .1 ); // set commission rate
33
                                                       Use CommissionEmployee's set
      cout << "\nUpdated employee information output</pre>
34
                                                       functions to change the object's instance
35
         << end1;
      employee.print(); // display the new employee informable values
36
                                                                                      (2 \text{ of } 2)
                                                 Call object's print function
37
38
     // display the employee's earnings
                                                 to display employee information
      cout << "\n\nEmployee's earnings: $" << employee.earnings() << endl;</pre>
39
40
41
     return 0:
                                                                     Call object's earnings
42 } // end main
                                                                      function to calculate earnings
Employee information obtained by get functions:
First name is Sue
Last name is Jones
Social security number is 222-22-2222
Gross sales is 10000.00
Commission rate is 0.06
Updated employee information output by print function:
commission employee: Sue Jones
social security number: 222-22-2222
gross sales: 8000.00
commission rate: 0.10
Employee's earnings: $800.00
```

# 12.4.2 Creating a BasePlusCommissionEmployee Class Without Using Inheritance

- Class BasePlusCommissionEmployee
  - Much of the code is similar to CommissionEmployee
    - private data members
    - public methods
    - constructor
  - Additions
    - private data member baseSalary
    - Methods setBaseSalary and getBaseSalary

Outline

BasePlus

(1 of 2)

Commission

Employee.h

```
1 // Fig. 12.7: BasePlusCommissionEmployee.h
2 // BasePlusCommissionEmployee class definition represents an employee
3 // that receives a base salary in addition to commission.
4 #ifndef BASEPLUS H
  #define BASEPLUS H
6
7 #include <string> // C++ standard string class
  using std::string;
9
10 class BasePlusCommissionEmployee
11
12 public:
     BasePlusCommissionEmployee( const string &, const string &,
13
        const string &, double = 0.0, double = 0.0, double = 0.0);
14
15
     void setFirstName( const string & ); // set first name
16
                                                                  Constructor takes one more argument,
17
     string getFirstName() const; // return first name
                                                                   which specifies the base salary
18
19
     void setLastName( const string & ); // set last name
20
     string getLastName() const; // return last name
21
22
     void setSocialSecurityNumber( const string & ); // set SSN
     string getSocialSecurityNumber() const; // return SSN
23
24
25
     void setGrossSales( double ); // set gross sales amount
     double getGrossSales() const; // return gross sales amount
26
27
     void setCommissionRate( double ); // set commission rate
28
     double getCommissionRate() const; // return commission rate
29
```



```
30
                                                                     Define get and set functions for
31
     void setBaseSalary( double ); // set base salary
     double getBaseSalary() const; // return base salary
32
                                                                      data member baseSalary
33
34
     double earnings() const; // calculate earnings
                                                                                     BasePlus
     void print() const; // print BasePlusCommissionEmployee object
35
                                                                                     Commission
36 private:
                                                                                     Employee.h
     string firstName;
37
     string lastName;
38
                                                                                     (2 \text{ of } 2)
     string socialSecurityNumber;
39
     double grossSales; // gross weekly sales
40
     double commissionRate; // commission percentage
41
42
     double baseSalary; // base salary
43 }; // end class BasePlusCommissionEmployee
                                                       Add data member baseSalary
44
45 #endif
```

```
1 // Fig. 12.8: BasePlusCommissionEmployee.cpp
2 // Class BasePlusCommissionEmployee member-function definitions.
                                                                                     Outline
  #include <iostream>
  using std::cout:
                                                                                     BasePlus
  // BasePlusCommissionEmployee class definition
                                                                                     Commission
  #include "BasePlusCommissionEmployee.h"
                                                                                     Employee.cpp
8
  // constructor
                                                                                     (1 \text{ of } 4)
10 BasePlusCommissionEmployee::BasePlusCommissionEmployee(
     const string &first, const string &last, const string &ssn,
11
     double sales, double rate, double salary )
12
13 {
                                                        Constructor takes one more argument,
     firstName = first; // should validate
14
                                                         which specifies the base salary
     lastName = last; // should validate
15
     socialSecurityNumber = ssn; // should validate
16
17
     setGrossSales( sales ); // validate and store gross sales
     setCommissionRate( rate ); // validate and store commission rate
18
     setBaseSalary( salary ); // validate and store base salary
19
20 } // end BasePlusCommissionEmployee constructor
21
                                                       Use function setBaseSalary to validate data
22 // set first name
23 void BasePlusCommissionEmployee::setFirstName( const string &first )
24 {
     firstName = first; // should validate
25
26 } // end function setFirstName
```



#### 27 28 // return first name 29 string BasePlusCommissionEmployee::getFirstName() const **30** [ return firstName; 31 32 } // end function getFirstName 33 34 // set last name 35 void BasePlusCommissionEmployee::setLastName( const string &last ) 36 { lastName = last; // should validate 37 38 } // end function setLastName 39 40 // return last name 41 string BasePlusCommissionEmployee::getLastName() const 42 { return lastName; 43 44 } // end function getLastName 45 46 // set social security number 47 void BasePlusCommissionEmployee::setSocialSecurityNumber( const string &ssn ) 48 49 { socialSecurityNumber = ssn; // should validate 50 51 } // end function setSocialSecurityNumber 52

#### Outline

BasePlus Commission Employee.cpp

(2 of 4)



#### 53 // return social security number 54 string BasePlusCommissionEmployee::getSocialSecurityNumber() const 55 return socialSecurityNumber; 56 57 } // end function getSocialSecurityNumber 58 59 // set gross sales amount 60 void BasePlusCommissionEmployee::setGrossSales( double sales ) 61 62 grossSales = (sales < 0.0)? 0.0 : sales; 63 } // end function setGrossSales 64 65 // return gross sales amount 66 double BasePlusCommissionEmployee::getGrossSales() const 67 return grossSales; 68 69 } // end function getGrossSales 70 71 // set commission rate 72 void BasePlusCommissionEmployee::setCommissionRate( double rate ) 73 { commissionRate = ( rate > 0.0 && rate < 1.0 ) ? rate : 0.0;74 75 } // end function setCommissionRate 76 77 // return commission rate 78 double BasePlusCommissionEmployee::getCommissionRate() const 79 { return commissionRate; 80 81 } // end function getCommissionRate 82

#### Outline

BasePlus Commission Employee.cpp

(3 of 4)



```
83 // set base salary
84 void BasePlusCommissionEmployee::setBaseSalary( double salary )
                                                                                     Outline
85 {
     baseSalary = (salary < 0.0)? 0.0 : salary;
86
                                                        Function setBaseSalary validates data
87 } // end function setBaseSalary
                                                        and sets instance variable baseSalary
88
89 // return base salary
                                                                                     Commission
90 double BasePlusCommissionEmployee::getBaseSalary() const
                                                                                     Employee.cpp
91 {
92
     return baseSalary;
                                                            Function getBaseSalary returns the
93 } // end function getBaseSalary
94
                                                             value of instance variable baseSalary
95 // calculate earnings
96 double BasePlusCommissionEmployee::earnings()
97 {
98
     return baseSalary + ( commissionRate * grossSales );
99 } // end function earnings
                                                         Update function earnings to calculate the
100
                                                         earnings of a base-salaried commission employee
101// print BasePlusCommissionEmployee object
102void BasePlusCommissionEmployee::print() const
103 {
     cout << "base-salaried commission employee: " << firstName << ' '</pre>
104
        << lastName << "\nsocial security number: " << socialSecurityNumber
105
        << "\ngross sales: " << grossSales
106
        << "\ncommission rate: " << commissionRate</pre>
107
                                                                Update function print
        << "\nbase salary: " << baseSalary; +</pre>
108
                                                                 to display base salary
109} // end function print
```



```
1 // Fig. 12.9: fig12_09.cpp
2 // Testing class BasePlusCommissionEmployee.
                                                                                      Outline
3 #include <iostream>
  using std::cout;
  using std::endl;
                                                                                      fig12_09.cpp
  using std::fixed;
7
                                                                                      (1 \text{ of } 3)
  #include <iomanip>
  using std::setprecision;
10
11 // BasePlusCommissionEmployee class definition
12 #include "BasePlusCommissionEmployee.h"
13
14 int main()
15 {
16
     // instantiate BasePlusCommissionEmployee object
     BasePlusCommissionEmployee
17
         employee( "Bob", "Lewis", "333-33-3333", 5000, .04, 300 );
18
19
     // set floating-point output formatting
20
21
     cout << fixed << setprecision( 2 );</pre>
22
                                                Instantiate BasePlusCommissionEmployee object
```



```
// get commission employee data
23
24
     cout << "Employee information obtained by get functions: \n"</pre>
                                                                                 Outline
        << "\nFirst name is " << employee.getFirstName()</pre>
25
        << "\nLast name is " << employee.getLastName()</pre>
26
                                                         Use
        << "\nSocial security number is "</pre>
27
                                                          BasePlusCommissionEmployee's
        << employee.getSocialSecurityNumber() <</pre>
28
                                                          get functions to retrieve the object's
        << "\nGross sales is " << employee.getGrossSales()</pre>
29
        30
31
        << "\nBase salary is " << employee.getBaseSalary() << endl:</pre>
32
     employee.setBaseSalary( 1000 ); // set base salary
33
34
                                                      Use BasePlusCommissionEmployee's
     cout << "\nUpdated employee information output by</pre>
35
                                                      setBaseSalary function to set base salary
        << end1;
36
     employee.print(); // display the new employee information
37
                                                              Call object's print function
38
     // display the employee's earnings
                                                              to display employee information
39
     cout << "\n\nEmployee's earnings: $" << employee.earnings() << endl;</pre>
40
41
42
     return 0:
43 } // end main
                                                                Call object's earnings
                                                                 function to calculate employee's
                                                                 earnings
```



#### Employee information obtained by get functions:

First name is Bob Last name is Lewis Social security number is 333-33-3333 Gross sales is 5000.00 Commission rate is 0.04 Base salary is 300.00

Updated employee information output by print function:

base-salaried commission employee: Bob Lewis

social security number: 333-33-3333

gross sales: 5000.00 commission rate: 0.04 base salary: 1000.00

Employee's earnings: \$1200.00

## Outline

fig12\_09.cpp

(3 of 3)

# **Software Engineering Observation 12.3**

Copying and pasting code from one class to another can spread errors across multiple source code files. To avoid duplicating code (and possibly errors), use inheritance, rather than the "copy-and-paste" approach, in situations where you want one class to "absorb" the data members and member functions of another class.



# **Software Engineering Observation 12.4**

With inheritance, the common data members and member functions of all the classes in the hierarchy are declared in a base class. When changes are required for these common features, software developers need to make the changes only in the base class—derived classes then inherit the changes. Without inheritance, changes would need to be made to all the source code files that contain a copy of the code in question.



## 12.4.3 Creating a CommissionEmployee-BasePlusCommissionEmployee Inheritance Hierarchy

# Class BasePlusCommissionEmployee

- Derived from class CommissionEmployee
  - Is a Commission Employee
  - Inherits all public members
- Constructor is not inherited
  - Use base-class initializer syntax to initialize base-class data member
- Has data member baseSalary



```
1 // Fig. 12.10: BasePlusCommissionEmployee.h
2 // BasePlusCommissionEmployee class derived from class
                                                                                     Outline
  // CommissionEmployee.
  #ifndef BASEPLUS H
  #define BASEPLUS H
                                                                                     BasePlus
                                                                                            sion
  #include <string> // C++ standard string class
                                                          Include the base-class header file
                                                                                            ree.h
  using std::string:
                                                            in the derived-class header file
9
                                                                                     (1 \text{ of } 1)
10 #include "CommissionEmployee.h" // CommissionEmployee class declaration
11
12 class BasePlusCommissionEmployee : public CommissionEmployee
13 €
14 public:
     BasePlusCommissionEmployee( const string &,
15
                                                   Class BasePlusCommissionEmployee derives
        const string &, double = 0.0, double = 0.
16
                                                   publicly from class CommissionEmployee
17
     void setBaseSalary( double ); // set base salary
18
     double getBaseSalary() const; // return base salary
19
20
     double earnings() const; // calculate earnings
21
     void print() const; // print BasePlusCommissionEmployee object
22
23 private:
     double baseSalary; // base salary
24
25 }; // end class BasePlusCommissionEmployee
26
27 #endif
```

BasePlus

(1 of 4)

Commission

Employee.cpp

```
1 // Fig. 12.11: BasePlusCommissionEmployee.cpp
2 // Class BasePlusCommissionEmployee member-function definitions.
3 #include <iostream>
  using std::cout;
5
  // BasePlusCommissionEmployee class definition
  #include "BasePlusCommissionEmployee.h"
8
  // constructor
10 BasePlusCommissionEmployee::BasePlusCommissionEmployee(
     const string &first, const string &last, const string &ssn,
11
     double sales, double rate, double salary )
12
13
     // explicitly call base-class constructor
14
     : CommissionEmployee(first, last, ssn, sales, rate)
15 {
     setBaseSalary( salary ); // validate and store base salary
16
17 } // end BasePlusCommissionEmployee constructor
                                                    Initialize base class data member by calling the base
18
                                                    -class constructor using base-class initializer syntax
19 // set base salary
20 void BasePlusCommissionEmployee::setBaseSalary( double salary )
21 {
22
     baseSalary = (salary < 0.0)? 0.0 : salary;
23 } // end function setBaseSalary
24
25 // return base salary
26 double BasePlusCommissionEmployee::getBaseSalary() const
27 {
     return baseSalary;
28
29 } // end function getBaseSalary
```



BasePlus

Commission

Compiler generates errors because base class's data member commissionRate and grossSales

// derived class cannot access the base class's private data

42 cout << "base-salaried commission employee: " << firstName << ' '

// derived class cannot access the base class's private data

return baseSalary + ( commissionRate \* grossSales );

<< lastName << "\nsocial security number: " << socialSecurityNumber

<< "\ngross sales: " << grossSales</pre>

32 double BasePlusCommissionEmployee::earnings() const

45 << "\ncommission rate: " << commissionRate

46 << "\nbase salary; " << baseSalary;</pre>

38 // print BasePlusCommissionEmployee object

39 void BasePlusCommissionEmployee::print() const

47 } // end function print

31 // calculate earnings

36 } // end function earnings

30

33 {

34

35

37

40 {

41

43

44

Compiler generates errors because the base class's data members firstName, lastName, socialSecurityNumber, grossSales and commissionRate are private

```
C:\cpphtp5_examples\ch12\Fig12_10_11\BasePlusCommission-Employee.cpp(35):
    error C2248: 'CommissionEmployee::commissionRate' :
    cannot access private member declared in class 'CommissionEmployee'
        C:\cpphtp5_examples\ch12\Fig12_10_11\CommissionEmployee.h(37) :
    see declaration of 'CommissionEmployee::commissionRate'
       C:\cpphtp5e_examples\ch12\Fig12_10_11\CommissionEmployee.h(10) : see declaration of 'CommissionEmployee'
C:\cpphtp5_examples\ch12\Fig12_10_11\BasePlusCommission-Employee.cpp(35):
    error C2248: 'CommissionEmployee::grossSales' :
    cannot access private member declared in class 'CommissionEmployee'
        C:\cpphtp5_examples\ch12\Fig12_10_11\CommissionEmployee.h(36) :
    see declaration of 'CommissionEmployee::grossSales'
        C:\cpphtp5_examples\ch12\Fig12_10_11\CommissionEmployee.h(10) :
    see declaration of 'CommissionEmployee'
C:\cpphtp5_examples\ch12\Fig12_10_11\BasePlusCommission-Employee.cpp(42):
    error C2248: 'CommissionEmployee::firstName' :
    cannot access private member declared in class 'CommissionEmployee'
       C:\cpphtp5_examples\ch12\Fig12_10_11\CommissionEmployee.h(33) :
    see declaration of 'CommissionEmployee::firstName'
       C:\cpphtp5_examples\ch12\Fig12_10_11\CommissionEmployee.h(10) :
    see declaration of 'CommissionEmployee'
```

BasePlus Commission Employee.cpp

(3 of 4)

```
C:\cpphtp5_examples\ch12\Fig12_10_11\BasePlusCommission-Employee.cpp(43):
 error C2248: 'CommissionEmployee::lastName':
 cannot access private member declared in class 'CommissionEmployee'
       C:\cpphtp5_examples\ch12\Fig12_10_11\CommissionEmployee.h(34):
           see declaration of 'CommissionEmployee::lastName'
       C:\cpphtp5_examples\ch12\Fig12_10_11\CommissionEmployee.h(10) : see declaration of 'CommissionEmployee'
C:\cpphtp5_examples\ch12\Fig12_10_11\BasePlusCommission-Employee.cpp(43):
 error C2248: 'CommissionEmployee::socialSecurity-Number':
 cannot access private member declared in class 'CommissionEmployee'
       C:\cpphtp5_examples\ch12\Fig12_10_11\CommissionEmployee.h(35):
           see declaration of 'CommissionEmployee::socialSecurityNumber'
       C:\cpphtp5_examples\ch12\Fig12_10_11\CommissionEmployee.h(10) : see declaration of 'CommissionEmployee'
C:\cpphtp5_examples\ch12\Fig12_10_11\BasePlusCommission-Employee.cpp(44) :
 error C2248: 'CommissionEmployee::grossSales':
 cannot access private member declared in class 'CommissionEmployee'
       C:\cpphtp5_examples\ch12\Fig12_10_11\CommissionEmployee.h(36) :
           see declaration of 'CommissionEmployee::grossSales'
       C:\cpphtp5_examples\ch12\Fig12_10_11\CommissionEmployee.h(10) : see declaration of 'CommissionEmployee'
C:\cpphtp5_examples\ch12\Fig12_10_11\BasePlusCommission-Employee.cpp(45):
 error C2248: 'CommissionEmployee::commissionRate':
 cannot access private member declared in class 'CommissionEmployee'
       C:\cpphtp5_examples\ch12\Fig12_10_11\CommissionEmployee.h(37):
           see declaration of 'CommissionEmployee::commissionRate'
       C:\cpphtp5_examples\ch12\Fig12_10_11\CommissionEmployee.h(10) : see declaration of 'CommissionEmployee'
```

BasePlus Commission Employee.cpp

(4 of 4)



# **Common Programming Error 12.1**

A compilation error occurs if a derived-class constructor calls one of its base-class constructors with arguments that are inconsistent with the number and types of parameters specified in one of the base-class constructor definitions.



# **Performance Tip 12.1**

In a derived-class constructor, initializing member objects and invoking base-class constructors explicitly in the member initializer list prevents duplicate initialization in which a default constructor is called, then data members are modified again in the derived-class constructor's body.



# 12.4.3 Creating a CommissionEmployee-BasePlusCommissionEmployee Inheritance Hierarchy (Cont.)

- Including the base class header file
  - Base class header file must be included in derived class header file for three reasons, the compiler must
    - Know that base class exists
    - Know size of inherited data members
    - Ensure that inherited class members are used properly

# 12.4.4 CommissionEmployee-BasePlusCommissionEmployee Inheritance Hierarchy Using protected Data

- Use protected data
  - Enable class BasePlusCommissionEmployee to directly access base class data members
  - Base class's protected members are inherited by all derived classes of that base class

# **Good Programming Practice 12.1**

Declare public members first, protected members second and private members last.



```
1 // Fig. 12.12: CommissionEmployee.h
2 // CommissionEmployee class definition with protected data.
3 #ifndef COMMISSION_H
4 #define COMMISSION_H
6 #include <string> // C++ standard string class
7 using std::string;
8
  class CommissionEmployee
10 [
11 public:
     CommissionEmployee( const string &, const string &, const string &,
12
13
        double = 0.0, double = 0.0);
14
     void setFirstName( const string & ); // set first name
15
     string getFirstName() const; // return first name
16
17
     void setLastName( const string & ); // set last name
18
     string getLastName() const; // return last name
19
20
     void setSocialSecurityNumber( const string & ); // set SSN
21
     string getSocialSecurityNumber() const; // return SSN
22
23
```



Commission Employee.h

(1 of 2)

```
void setGrossSales( double ); // set gross sales amount
24
25
      double getGrossSales() const; // return gross sales amount
                                                                                      Outline
26
27
     void setCommissionRate( double ); // set commission rate
      double getCommissionRate() const; // return commission rate
28
                                                                                      Commission
29
                                                                                      Employee.h
30
      double earnings() const; // calculate earnings
31
     void print() const; // print CommissionEmployee object
                                                                                      (2 \text{ of } 2)
32 protected:
     string firstName;
33
                                                          Declare protected data
     string lastName;
34
35
     string socialSecurityNumber;
36
     double grossSales; // gross weekly sales
     double commissionRate; // commission percentage
37
38 }; // end class CommissionEmployee
39
40 #endif
```

#### 1 // Fig. 12.13: CommissionEmployee.cpp 2 // Class CommissionEmployee member-function definitions. 3 #include <iostream> 4 using std::cout; 5 #include "CommissionEmployee.h" // CommissionEmployee class definition 7 // constructor CommissionEmployee::CommissionEmployee( 10 const string &first, const string &last, const string &ssn, 11 double sales. double rate ) 12 { firstName = first; // should validate 13 lastName = last; // should validate 14 socialSecurityNumber = ssn; // should validate 15 setGrossSales( sales ); // validate and store gross sales 16 setCommissionRate( rate ); // validate and store commission rate 17 18 } // end CommissionEmployee constructor 19 20 // set first name 21 void CommissionEmployee::setFirstName( const string &first ) 22 { firstName = first; // should validate 23 24 } // end function setFirstName 25 26 // return first name 27 string CommissionEmployee::getFirstName() const 28 { return firstName; 29 30 } // end function getFirstName

## Outline

Commission Employee.cpp

(1 of 4)



#### 31 32 // set last name 33 void CommissionEmployee::setLastName( const string &last ) 34 { lastName = last; // should validate 35 36 } // end function setLastName 37 38 // return last name 39 string CommissionEmployee::getLastName() const 40 { return lastName: 41 42 } // end function getLastName 43 44 // set social security number 45 void CommissionEmployee::setSocialSecurityNumber( const string &ssn ) 46 { socialSecurityNumber = ssn; // should validate 47 48 } // end function setSocialSecurityNumber 49 50 // return social security number 51 string CommissionEmployee::getSocialSecurityNumber() const 52 { return socialSecurityNumber; 53 54 } // end function getSocialSecurityNumber 55 56 // set gross sales amount 57 void CommissionEmployee::setGrossSales( double sales ) **58** { grossSales = ( sales < 0.0 ) ? 0.0 : sales;60 } // end function setGrossSales

## Outline

Commission Employee.cpp

(2 of 4)



#### 61 62 // return gross sales amount 63 double CommissionEmployee::getGrossSales() const 64 { return grossSales; 65 66 } // end function getGrossSales 67 68 // set commission rate 69 void CommissionEmployee::setCommissionRate( double rate ) 70 { 71 commissionRate = ( rate > 0.0 && rate < 1.0 ) ? rate : 0.0;72 } // end function setCommissionRate 73 74 // return commission rate 75 double CommissionEmployee::getCommissionRate() const 76 { return commissionRate; 77 78 } // end function getCommissionRate 79 80 // calculate earnings 81 double CommissionEmployee::earnings() const 82 { return commissionRate \* grossSales; 83 84 } // end function earnings

## Outline

Commission Employee.cpp

(3 of 4)

```
85
86 // print CommissionEmployee object
                                                                                          Outline
87 void CommissionEmployee::print() const
88 {
      cout << "commission employee: " << firstName << ' ' << lastName</pre>
89
                                                                                         Commission
         << "\nsocial security number: " << socialSecurityNumber</pre>
90
                                                                                         Employee.cpp
91
         << "\ngross sales: " << grossSales
         << "\ncommission rate: " << commissionRate;</pre>
92
                                                                                         (4 \text{ of } 4)
93 } // end function print
```

BasePlus

(1 of 1)

Commission

Employee.h

```
1 // Fig. 12.14: BasePlusCommissionEmployee.h
2 // BasePlusCommissionEmployee class derived from class
3 // CommissionEmployee.
4 #ifndef BASEPLUS H
  #define BASEPLUS H
  #include <string> // C++ standard string class
  using std::string;
9
10 #include "CommissionEmployee.h" // CommissionEmployee class declaration
11
12 class BasePlusCommissionEmployee : public CommissionEmployee
13 {
                                                                  BasePlusCommissionEmployee
14 public:
                                                                     still inherits publicly from
15
      BasePlusCommissionEmployee( const string &, const string &,
                                                                     CommissionEmployee
        const string &, double = 0.0, double = 0.0, double = 0.0 \downarrow
16
17
18
     void setBaseSalary( double ); // set base salary
19
      double getBaseSalary() const; // return base salary
20
      double earnings() const; // calculate earnings
21
      void print() const; // print BasePlusCommissionEmployee object
22
23 private:
      double baseSalary; // base salary
24
25 }; // end class BasePlusCommissionEmployee
26
27 #endif
```



```
1 // Fig. 12.15: BasePlusCommissionEmployee.cpp
2 // Class BasePlusCommissionEmployee member-function definitions.
3 #include <iostream>
4 using std::cout;
5
  // BasePlusCommissionEmployee class definition
7 #include "BasePlusCommissionEmployee.h"
8
9 // constructor
10 BasePlusCommissionEmployee::BasePlusCommissionEmployee(
     const string &first, const string &last, const string &ssn,
11
12
     double sales, double rate, double salary )
     // explicitly call base-class constructor
13
     : CommissionEmployee(first, last, ssn, sales, rate)
14
15 {
16
     setBaseSalary( salary ); // validate and store base salary
17 } // end BasePlusCommissionEmployee constructor
18
19 // set base salary
20 void BasePlusCommissionEmployee::setBaseSalary( double salary )
21 {
     baseSalary = (salary < 0.0)? 0.0: salary;
22
23 } // end function setBaseSalary
24
25 // return base salary
26 double BasePlusCommissionEmployee::getBaseSalary() const
27 {
     return baseSalary;
28
29 } // end function getBaseSalary
```



BasePlus Commission Employee.cpp

Call base-class constructor using base-class initializer syntax



```
30
31 // calculate earnings
                                                                                       Outline
32 double BasePlusCommissionEmployee::earnings() const
33 {
     // can access protected data of base class
34
                                                                                       BasePlus
      return baseSalary + ( commissionRate * grossSales );
35
                                                                                       Commission
36 } // end function earnings
                                                                                       Employee.cpp
37
38 // print BasePlusCommissionEmployee object
                                                                           Directly access base
39 void BasePlusCommissionEmployee::print() const
                                                                            class' s protected
40 {
                                                                            data
41
     // can access protected data of base class
      cout << "base-salaried commission employee: " << firstName << ' '</pre>
42
         << lastName << "\nsocial security number: " << socialSecurityNumber
43
         << "\ngross sales: " << grossSales</pre>
44
         << "\ncommission rate: " << commissionRate</pre>
45
         << "\nbase salary: " << baseSalary;</pre>
46
47 } // end function print
```

```
1 // Fig. 12.16: fig12_16.cpp
2 // Testing class BasePlusCommissionEmployee.
3 #include <iostream>
4 using std::cout;
5 using std::endl;
  using std::fixed;
8 #include <iomanip>
9 using std::setprecision;
10
11 // BasePlusCommissionEmployee class definition
12 #include "BasePlusCommissionEmployee.h"
13
14 int main()
15 {
16
     // instantiate BasePlusCommissionEmployee object
     BasePlusCommissionEmployee
17
         employee( "Bob", "Lewis", "333-33-3333", 5000, .04, 300 );
18
19
     // set floating-point output formatting
20
     cout << fixed << setprecision( 2 );</pre>
21
22
```

Fig12\_16.cpp

(1 of 3)

```
// get commission employee data
23
24
      cout << "Employee information obtained by get functions: \n"</pre>
         << "\nFirst name is " << employee.getFirstName()</pre>
25
         << "\nLast name is " << employee.getLastName()</pre>
26
         << "\nSocial security number is "</pre>
27
         << employee.getSocialSecurityNumber()</pre>
28
29
         << "\nGross sales is " << employee.getGrossSales()</pre>
         << "\nCommission rate is " << employee.getCommissionRate()</pre>
30
         << "\nBase salary is " << employee.getBaseSalary() << endl;</pre>
31
32
33
      employee.setBaseSalary( 1000 ); // set base salary
34
      cout << "\nUpdated employee information output by print function: \n"</pre>
35
         << end1;
36
37
      employee.print(); // display the new employee information
38
39
      // display the employee's earnings
      cout << "\n\nEmployee's earnings: $" << employee.earnings() << endl;</pre>
40
41
      return 0:
42
43 } // end main
```

Fig12\_16.cpp

(2 of 3)

#### Employee information obtained by get functions:

First name is Bob Last name is Lewis Social security number is 333-33-3333 Gross sales is 5000.00 Commission rate is 0.04 Base salary is 300.00

Updated employee information output by print function:

base-salaried commission employee: Bob Lewis

social security number: 333-33-3333

gross sales: 5000.00 commission rate: 0.04 base salary: 1000.00

Employee's earnings: \$1200.00

## **Outline**

Fig12\_16.cpp

(3 of 3)

# 12.4.4 CommissionEmployee-BasePlusCommissionEmployee Inheritance Hierarchy Using protected Data (Cont.)

- Using protected data members
  - Advantages
    - Derived class can modify values directly
      - Avoid set/get method call overhead
        - Slight increase in performance
  - Disadvantages
    - No validity checking
      - Derived class can assign illegal value
    - Implementation dependent
      - Derived class functions more likely dependent on base class implementation
      - Base class implementation changes may result in derived class modifications
        - Fragile (brittle) software



# **Software Engineering Observation 12.5**

It is appropriate to use the protected access specifier when a base class should provide a service (i.e., a member function) only to its derived classes (and friends), not to other clients.



# **Software Engineering Observation 12.6**

Declaring base-class data members private (as opposed to declaring them protected) enables programmers to change the base-class implementation without having to change derived-class implementations.



# **Error-Prevention Tip 12.1**

When possible, avoid including protected data members in a base class. Rather, include non-private member functions that access private data members, ensuring that the object maintains a consistent state.



# 12.4.5 CommissionEmployee-BasePlusCommissionEmployee Inheritance Hierarchy Using private Data

- Reexamine hierarchy
  - Use the best software engineering practice
    - Declare data members as private
    - Provide public get and set functions
    - Use *get* method to obtain values of data members

```
1 // Fig. 12.17: CommissionEmployee.h
2 // CommissionEmployee class definition with good software engineering.
3 #ifndef COMMISSION_H
  #define COMMISSION_H
5
  #include <string> // C++ standard string class
  using std::string;
8
  class CommissionEmployee
10
11 public:
12
     CommissionEmployee( const string &, const string &, const string &,
        double = 0.0, double = 0.0);
13
14
     void setFirstName( const string & ); // set first name
15
     string getFirstName() const; // return first name
16
17
18
     void setLastName( const string & ); // set last name
     string getLastName() const; // return last name
19
20
     void setSocialSecurityNumber( const string & ); // set SSN
21
22
     string getSocialSecurityNumber() const; // return SSN
23
     void setGrossSales( double ); // set gross sales amount
24
     double getGrossSales() const; // return gross sales amount
25
26
27
     void setCommissionRate( double ); // set commission rate
28
     double getCommissionRate() const; // return commission rate
```

Commission Employee.h

(1 of 2)



```
29
     double earnings() const; // calculate earnings
30
                                                                                      Outline
     void print() const; // print CommissionEmployee object
31
32 private:
                                                            Declare private data
     string firstName;
33
                                                                                     Commission
     string lastName;
34
                                                                                     Employee.h
35
     string socialSecurityNumber;
     double grossSales; // gross weekly sales
36
                                                                                     (2 \text{ of } 2)
     double commissionRate; // commission percentage
37
38 }; // end class CommissionEmployee
39
40 #endif
```

Commission

(1 of 4)

Employee.cpp

```
1 // Fig. 12.18: CommissionEmployee.cpp
2 // Class CommissionEmployee member-function definitions.
3 #include <iostream>
  using std::cout;
5
  #include "CommissionEmployee.h" // CommissionEmployee class definition
7
  // constructor
  CommissionEmployee::CommissionEmployee(
     const string &first, const string &last, const string &ssn,
10
11
     double sales, double rate )
     : firstName( first ), lastName( last ), socialSecurityNumber( ssn )
12
13 {
     setGrossSales( sales ); // validate and store gros
14
                                                         Use member initializers to set the values
     setCommissionRate( rate ); // validate and store
15
                                                         of members firstName, lastname
16 } // end CommissionEmployee constructor
                                                         and socialSecurityNumber
17
18 // set first name
19 void CommissionEmployee::setFirstName( const string &first )
20 {
     firstName = first; // should validate
21
22 } // end function setFirstName
23
24 // return first name
25 string CommissionEmployee::getFirstName() const
26 {
     return firstName:
27
28 } // end function getFirstName
```



#### 29 30 // set last name 31 void CommissionEmployee::setLastName( const string &last ) 32 { lastName = last; // should validate 33 34 } // end function setLastName 35 36 // return last name 37 string CommissionEmployee::getLastName() const 38 { return lastName: 39 40 } // end function getLastName 41 42 // set social security number 43 void CommissionEmployee::setSocialSecurityNumber( const string &ssn ) 44 { socialSecurityNumber = ssn; // should validate 45 46 } // end function setSocialSecurityNumber 47 48 // return social security number 49 string CommissionEmployee::getSocialSecurityNumber() const 50 { return socialSecurityNumber; 51 52 } // end function getSocialSecurityNumber 53 54 // set gross sales amount 55 void CommissionEmployee::setGrossSales( double sales ) 56 { grossSales = ( sales < 0.0 ) ? 0.0 : sales;57 58 } // end function setGrossSales

## Outline

Commission Employee.cpp

(2 of 4)



# 59 60 // return gross sales amount 61 double CommissionEmployee::getGrossSales() const 62 { 63 return grossSales; 64 } // and function getGrossSales

## **Outline**

Commission Employee.cpp

(3 of 4)

63 64 } // end function getGrossSales 65 66 // set commission rate 67 void CommissionEmployee::setCommissionRate( double rate ) 68 { commissionRate = ( rate > 0.0 && rate < 1.0 ) ? rate : 0.0;69 70 } // end function setCommissionRate 71 72 // return commission rate 73 double CommissionEmployee::getCommissionRate() const 74 { return commissionRate; 75 76 } // end function getCommissionRate 77 78 // calculate earnings

79 double CommissionEmployee::earnings() const

82 } // end function earnings

return getCommissionRate() \* getGrossSales();

80 {

83

Use *get* functions to obtain the values of data

members

```
84 // print CommissionEmployee object
85 void CommissionEmployee::print() const
                                                                                          Outline
86 {
      cout << "commission employee: "</pre>
87
         << getFirstName() << ' ' << getLastName()</pre>
88
                                                                                          Commission
         << "\nsocial security number: " << getSocialSecurityNumber()</pre>
89
                                                                                          Employee.cpp
         << "\ngross sales: " << getGrossSales()</pre>
90
         << "\ncommission rate: " << getCommissionRate();</pre>
91
                                                                        Use get functions to obtain
92 } // end function print
                                                                         the values of data
                                                                         members
```

# **Performance Tip 12.2**

Using a member function to access a data member's value can be slightly slower than accessing the data directly. However, today's optimizing compilers are carefully designed to perform many optimizations implicitly (such as inlining set and get member-function calls). As a result, programmers should write code that adheres to proper software engineering principles, and leave optimization issues to the compiler. A good rule is, "Do not secondguess the compiler."



```
1 // Fig. 12.19: BasePlusCommissionEmployee.h
2 // BasePlusCommissionEmployee class derived from class
3 // CommissionEmployee.
4 #ifndef BASEPLUS H
  #define BASEPLUS H
7 #include <string> // C++ standard string class
  using std::string;
9
10 #include "CommissionEmployee.h" // CommissionEmployee class declaration
11
12 class BasePlusCommissionEmployee : public CommissionEmployee
13 {
14 public:
      BasePlusCommissionEmployee( const string &, const string &,
15
16
         const string &, double = 0.0, double = 0.0, double = 0.0);
17
     void setBaseSalary( double ); // set base salary
18
      double getBaseSalary() const; // return base salary
19
20
     double earnings() const; // calculate earnings
21
22
     void print() const; // print BasePlusCommissionEmployee object
23 private:
      double baseSalary; // base salary
24
25 }; // end class BasePlusCommissionEmployee
26
27 #endif
```

BasePlus Commission Employee.h

(1 of 1)



```
1 // Fig. 12.20: BasePlusCommissionEmployee.cpp
2 // Class BasePlusCommissionEmployee member-function definitions.
3 #include <iostream>
4 using std::cout;
5
  // BasePlusCommissionEmployee class definition
7 #include "BasePlusCommissionEmployee.h"
8
9 // constructor
10 BasePlusCommissionEmployee::BasePlusCommissionEmployee(
     const string &first, const string &last, const string &ssn,
11
     double sales, double rate, double salary )
12
     // explicitly call base-class constructor
13
14
     : CommissionEmployee(first, last, ssn, sales, rate)
15 {
     setBaseSalary( salary ); // validate and store base salary
16
17 } // end BasePlusCommissionEmployee constructor
18
19 // set base salary
20 void BasePlusCommissionEmployee::setBaseSalary( double salary )
21 {
     baseSalary = (salary < 0.0)? 0.0 : salary;
22
23 } // end function setBaseSalary
24
25 // return base salary
26 double BasePlusCommissionEmployee::getBaseSalary() const
27 {
     return baseSalary;
28
29 } // end function getBaseSalary
```

BasePlus Commission Employee.cpp

(1 of 2)



```
30
31 // calculate earnings
                                                                                       Outline
32 double BasePlusCommissionEmployee::earnings() const
33 {
      return getBaseSalary() + CommissionEmployee::earnings();
34
                                                                                       BasePlus
35 } // end function earnings
36
                                                              Invoke base class' searnings
37 // print BasePlusCommissionEmployee object
                                                              function
38 void BasePlusCommissionEmployee::print() const
                                                                                       (2 \text{ of } 2)
39 {
40
     cout << "base-salaried ";</pre>
41
42
     // invoke CommissionEmployee's print function
      CommissionEmployee::print();
43
44
45
     cout << "\nbase salary: " << getBaseSalary()</pre>
                                                    Invoke base class' sprint
46 } // end function print
                                                     function
```

# **Common Programming Error 12.2**

When a base-class member function is redefined in a derived class, the derived-class version often calls the base-class version to do additional work. Failure to use the :: operator prefixed with the name of the base class when referencing the base class's member function causes infinite recursion, because the derived-class member function would then call itself.



# **Common Programming Error 12.3**

Including a base-class member function with a different signature in the derived class hides the base-class version of the function. Attempts to call the base-class version through the public interface of a derived-class object result in compilation errors.

```
1 // Fig. 12.21: fig12_21.cpp
2 // Testing class BasePlusCommissionEmployee.
3 #include <iostream>
4 using std::cout;
5 using std::endl;
6 using std::fixed;
7
8 #include <iomanip>
9 using std::setprecision;
10
11 // BasePlusCommissionEmployee class definition
12 #include "BasePlusCommissionEmployee.h"
13
```

fig12\_21.cpp

(1 of 3)

```
14 int main()
                                                                                                           81
15 {
                                                                                       Outline
16
     // instantiate BasePlusCommissionEmployee object
17
      BasePlusCommissionEmployee
         employee( "Bob", "Lewis", "333-33-3333", 5000, .04, 300 );
18
19
                                                                                       fig12_21.cpp
     // set floating-point output formatting
20
                                                    Create BasePlusCommissionEmployee object
21
      cout << fixed << setprecision( 2 );</pre>
22
     // get commission employee data
23
      cout << "Employee information obtained by get functions: \n"</pre>
24
         << "\nFirst name is " << employee.getFirstName()</pre>
25
26
         << "\nLast name is " << employee.getLastName()</pre>
                                                                        Use inherited get methods to access
         << "\nSocial security number is "</pre>
27
                                                                        base class private members
         << employee.getSocialSecurityNumber()</pre>
28
29
         << "\nGross sales is " << employee.getGrossSales()</pre>
         << "\nCommission rate is " << employee.getCommissionRate()</pre>
30
         << "\nBase salary is " << employee.getBaseSalary() << endl;</pre>
31
32
                                                             Use BasePlusCommissionEmployee
33
      employee.setBaseSalary( 1000 ); // set base salary
                                                             get method to access private member
34
      cout << "\nUpdated employee information</pre>
35
                                               Use BasePlusCommissionEmployee set method
36
         << end1;
                                               to modify private data member baseSalary
37
      employee.print(); // display the new em
38
     // display the employee's earnings
39
      cout << "\n\nEmployee's earnings: $" << employee.earnings() << endl;</pre>
40
41
```

return 0:

43 } // end main

42

#### Employee information obtained by get functions:

First name is Bob Last name is Lewis Social security number is 333-33-3333 Gross sales is 5000.00 Commission rate is 0.04 Base salary is 300.00

Updated employee information output by print function:

base-salaried commission employee: Bob Lewis

social security number: 333-33-3333

gross sales: 5000.00 commission rate: 0.04 base salary: 1000.00

Employee's earnings: \$1200.00

## **Outline**

fig12\_21.cpp

(3 of 3)

# 12.5 Constructors and Destructors in Derived Classes

- Instantiating derived-class object
  - Chain of constructor calls
    - Derived-class constructor invokes base class constructor
      - Implicitly or explicitly
    - Base of inheritance hierarchy
      - Last constructor called in chain
      - First constructor body to finish executing
      - Example: CommissionEmployee/
         BasePlusCommissionEmployee hierarchy
        - CommissionEmployee constructor called last
        - CommissionEmployee constructor body finishes execution first
    - Initializing data members
      - Each base-class constructor initializes its data members that are inherited by derived class



When a program creates a derived-class object, the derived-class constructor immediately calls the base-class constructor, the base-class constructor's body executes, then the derived class's member initializers execute and finally the derived-class constructor's body executes. This process cascades up the hierarchy if the hierarchy contains more than two levels.



# 12.5 Constructors and Destructors in Derived Classes (Cont.)

- Destroying derived-class object
  - Chain of destructor calls
    - Reverse order of constructor chain
    - Destructor of derived-class called first
    - Destructor of next base class up hierarchy next
      - Continue up hierarchy until final base reached
        - After final base-class destructor, object removed from memory
- Base-class constructors, destructors, assignment operators
  - Not inherited by derived classes



Suppose that we create an object of a derived class where both the base class and the derived class contain objects of other classes. When an object of that derived class is created, first the constructors for the base class's member objects execute, then the base-class constructor executes, then the constructors for the derived class's member objects execute, then the derived class' s constructor executes. Destructors for derived-class objects are called in the reverse of the order in which their corresponding constructors are called.



```
1 // Fig. 12.22: CommissionEmployee.h
2 // CommissionEmployee class definition represents a commission employee.
                                                                                     Outline
3 #ifndef COMMISSION_H
4 #define COMMISSION H
5
6 #include <string> // C++ standard string class
                                                                                     Commission
7 using std::string;
                                                                                     Employee.h
8
  class CommissionEmployee
                                                                                     (1 \text{ of } 2)
10 [
11 public:
     CommissionEmployee( const string &, const string &, const string &,
12
13
        double = 0.0, double = 0.0);
                                                               CommissionEmployee destructor
     ~CommissionEmployee(); // destructor ←
14
15
16
     void setFirstName( const string & ); // set first name
     string getFirstName() const; // return first name
17
18
     void setLastName( const string & ); // set last name
19
     string getLastName() const; // return last name
20
21
22
     void setSocialSecurityNumber( const string & ); // set SSN
23
     string getSocialSecurityNumber() const; // return SSN
24
     void setGrossSales( double ); // set gross sales amount
25
     double getGrossSales() const; // return gross sales amount
26
27
     void setCommissionRate( double ); // set commission rate
28
     double getCommissionRate() const; // return commission rate
29
```



```
30
31
     double earnings() const; // calculate earnings
32
     void print() const; // print CommissionEmployee object
33 private:
34
     string firstName;
35
     string lastName;
36
     string socialSecurityNumber;
     double grossSales; // gross weekly sales
37
     double commissionRate; // commission percentage
38
39 }; // end class CommissionEmployee
40
41 #endif
```

Commission Employee.h

(2 of 2)

```
1 // Fig. 12.23: CommissionEmployee.cpp
2 // Class CommissionEmployee member-function definitions.
                                                                                       Outline
3 #include <iostream>
4 using std::cout;
  using std::endl;
6
                                                                                       Commission
  #include "CommissionEmployee.h" // CommissionEmployee class definition
                                                                                       Employee.cpp
8
  // constructor
                                                                                       (1 \text{ of } 4)
10 CommissionEmployee::CommissionEmployee(
      const string &first, const string &last, const string &ssn,
11
12
      double sales, double rate )
13
      : firstName( first ), lastName( last ), socialSecurityNumber( ssn )
14 {
      setGrossSales( sales ); // validate and store gross sales
15
16
      setCommissionRate( rate ); // validate and store commission rate
17
18
     cout << "CommissionEmployee constructor: " << endl;</pre>
     print();
19
     cout << "\n\n";
20
                                                              Constructor and destructor output messages
21 } // end CommissionEmployee constructor
                                                              to demonstrate function call order
22
23 // destructor
24 CommissionEmployee::~CommissionEmployee()
25 {
26
     cout << "CommissionEmployee destructor: " << endl;</pre>
27
     print():
     cout << "\n\n";
28
29 } // end CommissionEmployee destructor
```

#### 30 31 // set first name 32 void CommissionEmployee::setFirstName( const string &first ) 33 { firstName = first; // should validate 34 35 } // end function setFirstName 36 37 // return first name 38 string CommissionEmployee::getFirstName() const 39 { return firstName: 40 41 } // end function getFirstName 42 43 // set last name 44 void CommissionEmployee::setLastName( const string &last ) 45 { lastName = last; // should validate 46 47 } // end function setLastName 48 49 // return last name 50 string CommissionEmployee::getLastName() const 51 { return lastName; 52 53 } // end function getLastName 54 55 // set social security number 56 void CommissionEmployee::setSocialSecurityNumber( const string &ssn ) **57** { socialSecurityNumber = ssn; // should validate 58 59 } // end function setSocialSecurityNumber

## Outline

Commission Employee.cpp

(2 of 4)



#### 60 61 // return social security number 62 string CommissionEmployee::getSocialSecurityNumber() const 63 { return socialSecurityNumber; 64 65 } // end function getSocialSecurityNumber 66 67 // set gross sales amount 68 void CommissionEmployee::setGrossSales( double sales ) 69 { grossSales = (sales < 0.0)? 0.0 : sales; 70 71 } // end function setGrossSales 72 73 // return gross sales amount 74 double CommissionEmployee::getGrossSales() const **75** { return grossSales; 76 77 } // end function getGrossSales 78 79 // set commission rate 80 void CommissionEmployee::setCommissionRate( double rate ) 81 commissionRate = ( rate > 0.0 && rate < 1.0 ) ? rate : 0.0;82 83 } // end function setCommissionRate 84 85 // return commission rate 86 double CommissionEmployee::getCommissionRate() const 87 [ return commissionRate; 88 89 } // end function getCommissionRate

### Outline

Commission Employee.cpp

(3 of 4)



#### 90 91 // calculate earnings 92 double CommissionEmployee::earnings() const 93 { return getCommissionRate() \* getGrossSales(); 94 95 } // end function earnings 96 97 // print CommissionEmployee object 98 void CommissionEmployee::print() const 99 { cout << "commission employee: "</pre> 100 << getFirstName() << ' ' << getLastName()</pre> 101 << "\nsocial security number: " << getSocialSecurityNumber()</pre> 102 << "\ngross sales: " << getGrossSales()</pre> 103 << "\ncommission rate: " << getCommissionRate();</pre> 104 105} // end function print

## Outline

Commission Employee.cpp

(4 of 4)

```
1 // Fig. 12.24: BasePlusCommissionEmployee.h
2 // BasePlusCommissionEmployee class derived from class
                                                                                     Outline
3 // CommissionEmployee.
4 #ifndef BASEPLUS H
  #define BASEPLUS H
6
                                                                                    BasePlus
7 #include <string> // C++ standard string class
                                                                                    Commission
  using std::string;
                                                                                    Employee.h
9
10 #include "CommissionEmployee.h" // CommissionEmployee class declaration
                                                                                    (1 \text{ of } 1)
11
12 class BasePlusCommissionEmployee : public CommissionEmployee
13 {
14 public:
     BasePlusCommissionEmployee( const string &, const string &,
15
        const string &, double = 0.0, double = 0.0, double = 0.0);
16
     ~BasePlusCommissionEmployee(); // destructor
17
18
                                                               BasePlusCommissionEmployee
     void setBaseSalary( double ); // set base salary
19
                                                               destructor
20
     double getBaseSalary() const; // return base salary
21
22
     double earnings() const; // calculate earnings
     void print() const; // print BasePlusCommissionEmployee object
23
24 private:
     double baseSalary; // base salary
25
26 }; // end class BasePlusCommissionEmployee
27
28 #endif
```



```
1 // Fig. 12.25: BasePlusCommissionEmployee.cpp
2 // Class BasePlusCommissionEmployee member-function definitions.
                                                                                       Outline
3 #include <iostream>
4 using std::cout;
  using std::endl;
6
                                                                                       BasePlus
7 // BasePlusCommissionEmployee class definition
                                                                                       Commission
  #include "BasePlusCommissionEmployee.h"
                                                                                       Employee.cpp
9
10 // constructor
                                                                                       (1 \text{ of } 2)
11 BasePlusCommissionEmployee::BasePlusCommissionEmployee(
      const string &first, const string &last, const string &ssn,
12
     double sales, double rate, double salary )
13
     // explicitly call base-class constructor
14
15
      : CommissionEmployee(first, last, ssn, sales, rate)
16 {
17
      setBaseSalary( salary ); // validate and store base salary
18
     cout << "BasePlusCommissionEmployee constructor: " << endl:</pre>
19
     print();
20
     cout << "\n\n";</pre>
                                                                        Constructor and destructor
21
22 } // end BasePlusCommissionEmployee constructor
                                                                         output messages to demonstrate
23
                                                                         function call order
24 // destructor
25 BasePlusCommissionEmployee::~BasePlusCommissionEmployee()
26 {
     cout << "BasePlusCommissionEmployee destructor: " << endl;</pre>
27
     print();
28
     cout << "\n\n";</pre>
29
30 } // end BasePlusCommissionEmployee destructor
```



#### 31 32 // set base salary 33 void BasePlusCommissionEmployee::setBaseSalary( double salary ) 34 { baseSalary = (salary < 0.0)? 0.0 : salary; 35 36 } // end function setBaseSalary 37 38 // return base salary 39 double BasePlusCommissionEmployee::getBaseSalary() const 40 { return baseSalary; 41 42 } // end function getBaseSalary 43 44 // calculate earnings 45 double BasePlusCommissionEmployee::earnings() const 46 { return getBaseSalary() + CommissionEmployee::earnings(); 47 48 } // end function earnings 49 50 // print BasePlusCommissionEmployee object 51 void BasePlusCommissionEmployee::print() const 52 { cout << "base-salaried ":</pre> 53 54 55 // invoke CommissionEmployee's print function CommissionEmployee::print(); 56 57 cout << "\nbase salary: " << getBaseSalary();</pre> 58 59 } // end function print

## Outline

BasePlus Commission Employee.cpp

(2 of 2)



```
1 // Fig. 12.26: fig12_26.cpp
2 // Display order in which base-class and derived-class constructors
3 // and destructors are called.
4 #include <iostream>
5 using std::cout;
6 using std::endl;
7 using std::fixed;
8
9 #include <iomanip>
10 using std::setprecision;
11
12 // BasePlusCommissionEmployee class definition
13 #include "BasePlusCommissionEmployee.h"
```

fig12\_26.cpp

(1 of 4)

```
14
15 int main()
                                                                                     Outline
16 {
     // set floating-point output formatting
17
     cout << fixed << setprecision( 2 );</pre>
18
                                                               CommissionEmployee object
19
                                                               goes in and out of scope immediately
     { // begin new scope
20
        CommissionEmployee employee1( 4
21
                                                                                    (2 \text{ of } 4)
22
      "Bob", "Lewis", "333-33-3333", 5000, .04 );
     } // end scope
23
24
25
     cout << endl;</pre>
     BasePlusCommissionEmployee
26
27
        employee2( "Lisa", "Jones", "555-55-5555", 2000, .06, 800 );
28
29
     cout << endl;</pre>
     BasePlusCommissionEmployee
30
        employee3( "Mark", "Sands", ")
                                     31
     cout << endl;</pre>
32
33
     return 0:
                                               Instantiate two BasePlusCommissionEmployee
34 } // end main
                                                objects to demonstrate order of derived-class and base
                                               -class constructor/destructor function calls
```

CommissionEmployee constructor: commission employee: Bob Lewis social security number: 333-33-3333 gross sales: 5000.00

gross sales: 5000.00 commission rate: 0.04

CommissionEmployee destructor: commission employee: Bob Lewis social security number: 333-33-3333

gross sales: 5000.00 commission rate: 0.04

**CommissionEmployee** constructor called for object in block; destructor called immediately as execution leaves scope

CommissionEmployee constructor: ←

base-salaried commission employee: Lisa Jones

social security number: 555-55-5555

gross sales: 2000.00 commission rate: 0.06

BasePlusCommissionEmployee constructor: 

base-salaried commission employee: Lisa Jones

social security number: 555-55-555

gross sales: 2000.00 commission rate: 0.06 base salary: 800.00

CommissionEmployee constructor: ← commission employee: Mark Sands social security number: 888-88-8888

gross sales: 8000.00 commission rate: 0.15

Base-class **CommissionEmployee** constructor executes first when instantiating derived-class **BasePlusCommissionEmployee** object

Derived-class BasePlusCommissionEmployee constructor body executes after base-class

CommissionEmployee' s constructor finishes execution

Base-class **CommissionEmployee** constructor executes first when instantiating derived-class

BasePlusCommissionEmployee object

(commuea ar top of next strae...)



gross sales: 8000.00 commission rate: 0.15 base salary: 2000.00

BasePlusCommissionEmployee destructor:

base-salaried commission employee: Mark Sands

social security number: 888-88-8888

gross sales: 8000.00 commission rate: 0.15 base salary: 2000.00

CommissionEmployee destructor: Commission employee: Mark Sands social security number: 888-88-8888

gross sales: 8000.00 commission rate: 0.15

BasePlusCommissionEmployee destructor:

base-salaried commission employee: Lisa Jones

social security number: 555-55-5555

gross sales: 2000.00 commission rate: 0.06 base salary: 800.00

CommissionEmployee destructor: Commission employee: Lisa Jones social security number: 555-55-555

gross sales: 2000.00 commission rate: 0.06

<u>Outlina</u>

 $(A \circ f A)$ 

Derived-class BasePlusCommissionEmployee constructor body executes after base-class CommissionEmployee's constructor finishes execution

Destructors for

BasePlusCommissionEmployee object called in reverse order of constructors

Destructors for

BasePlusCommissionEmployee object called in reverse order of constructors



# 12.6 public, protected and private Inheritance

- public inheritance
  - Base class public members → derived class public members
  - Base class protected members → derived class protected members
  - Base class private members are not accessible
- protected inheritance (not is-a relationship)
  - Base class public and protected members → derived class protected members
- private inheritance (not is-a relationship)
  - Base class public and protected members → derived class private members

| Base-class<br>member-<br>access<br>specifier | Type of inheritance  |  |   |
|--|--|--|---|
|  | public<br>inheritance  | protected<br>inheritance   | private<br>inheritance  |
| public                                       | public in derived class.  Can be accessed directly by member functions, friend functions and nonmember functions.                                  | protected in derived class.  Can be accessed directly by member functions and friend functions.  | private in derived class.  Can be accessed directly by member functions and fri end functions.  |
| protected                                    | protected in derived class.  Can be accessed directly by member functions and friend functions.  | protected in derived class.  Can be accessed directly by member functions and friend functions.  | private in derived class.  Can be accessed directly by member functions and fri end functions.  |
| private                                      | Hidden in derived class.  Can be accessed by member functions and friend functions through public or protected member functions of the base class. | Hidden in derived class.  Can be accessed by member functions and friend functions through public or protected member functions of the base class. | Hidden in derived class.  Can be accessed by membe functions and friend functions through public or protected member functions of the base class. |

Fig. 12.27 | Summary of base-class member accessibility in a derived class.



# 12.7 Software Engineering with Inheritance

- Customizing existing software
  - Inheriting from existing classes
    - Can include additional members
    - Can redefine base-class members
    - No direct access to base class's source code
      - Only links to object code
  - Independent software vendors (ISVs)
    - Develop proprietary code for sale/license
      - Available in object-code format
    - Users derive new classes
      - Without accessing ISV proprietary source code

At the design stage in an object-oriented system, the designer often determines that certain classes are closely related. The designer should "factor out" common attributes and behaviors and place these in a base class, then use inheritance to form derived classes, endowing them with capabilities beyond those inherited from the base class.

The creation of a derived class does not affect its base class's source code. Inheritance preserves the integrity of a base class.



Just as designers of non-object-oriented systems should avoid proliferation of functions, designers of object-oriented systems should avoid proliferation of classes. Proliferation of classes creates management problems and can hinder software reusability, because it becomes difficult for a client to locate the most appropriate class of a huge class library. The alternative is to create fewer classes that provide more substantial functionality, but such classes might provide too much functionality.



# **Performance Tip 12.3**

If classes produced through inheritance are larger than they need to be (i.e., contain too much functionality), memory and processing resources might be wasted. Inherit from the class whose functionality is "closest" to what is needed.