12

Object-Oriented Programming: Inheritance

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.)

- **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**

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**

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**

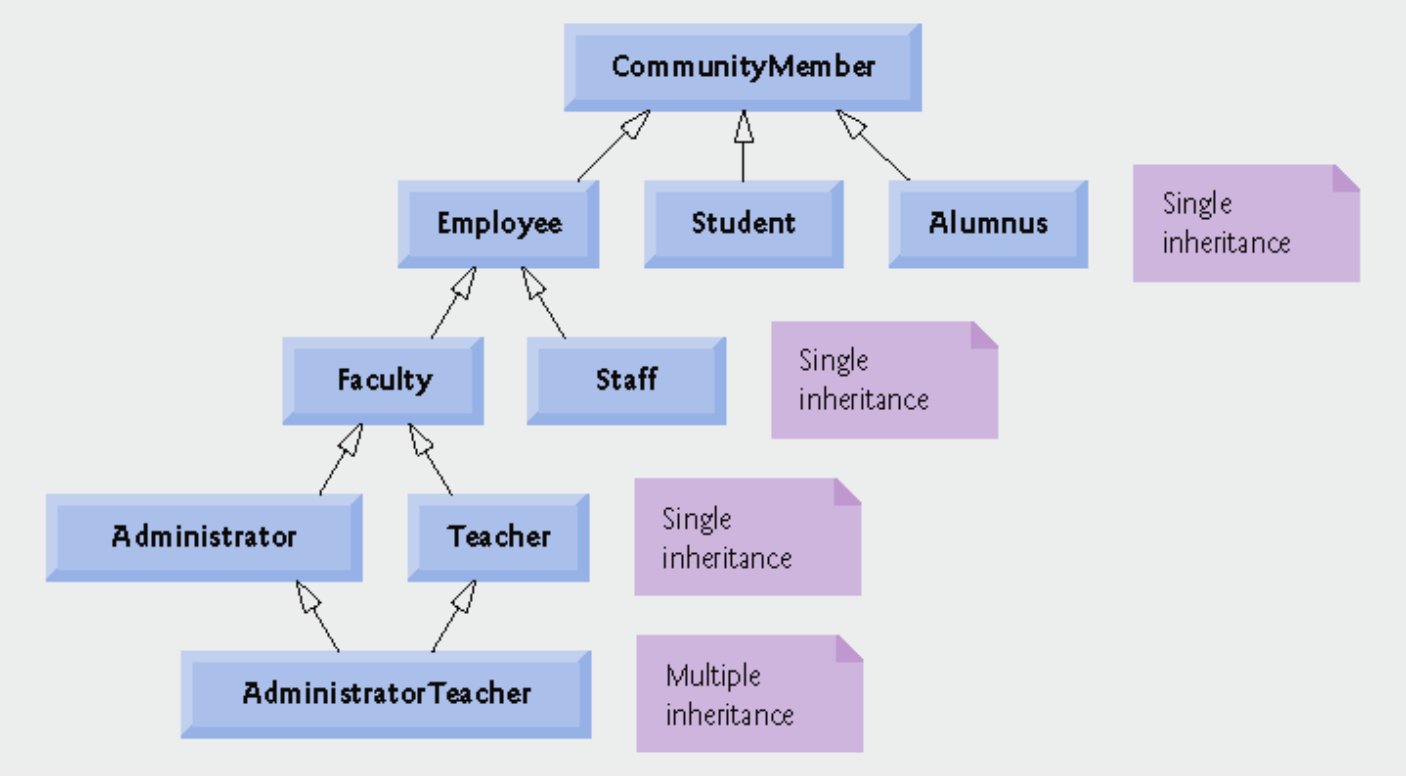

Fig. 12.2 | Inheritance hierarchy for university `CommunityMembers`.

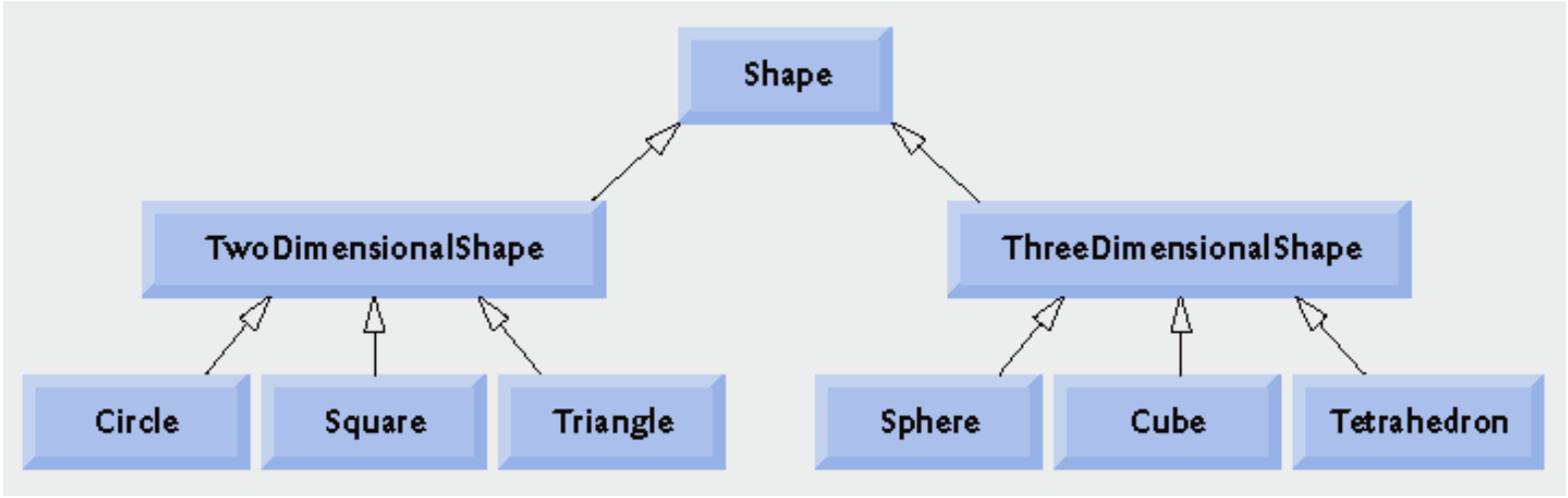

Fig. 12.3 | Inheritance hierarchy for Shapes.

12.2 Base Classes and Derived Classes (Cont.)

- `public` inheritance
  - Specify with:
    `Class TwoDimensionalShape : public Shape`
    - Class `TwoDimensionalShape` 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

12.4.1 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)

```
1  // Fig. 12.4: CommissionEmployee.h
2  // CommissionEmployee class definition represents a commission employee.
3  #ifndef COMMISSION_H
4  #define COMMISSION_H
5
6  #include <string> // C++ standard string class
7  using std::string;
8
9  class CommissionEmployee
10 {
11 public:
12    CommissionEmployee( const string &, const string &, const string &,
13       double = 0.0, double = 0.0 );
14
15    void setFirstName( const string & ); // set first name
16    string getFirstName() const; // return first name
17
18    void setLastName( const string & ); // set last name
19    string getLastName() const; // return last name
20
21    void setSocialSecurityNumber( const string & ); // set SSN
22    string getSocialSecurityNumber() const; // return SSN
23
24    void setGrossSales( double ); // set gross sales amount
25    double getGrossSales() const; // return gross sales amount
26
27    void setCommissionRate( double ); // set commission rate (percentage)
28    double getCommissionRate() const; // return commission rate
```

Class **CommissionEmployee** constructor

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

Declare **`private`** data members

Outline

`CommissionEmployee.h`

(2 of 2)

Outline

Commission Employee.cpp

(1 of 4)

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

Initialize data members

Outline

Commission Employee.cpp

(2 of 4)

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

Function **setGrossSales** validates gross sales amount

Outline

```
61
62 // return gross sales amount
63 double CommissionEmployee::getGrossSales() const
64 {
65    return grossSales;
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 {
77    return commissionRate;
78 } // end function getCommissionRate
```

Function **`setCommissionRate`** validates commission rate

(3 of 4)

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

Outline

Function **earnings** calculates earnings

Function **print** displays **CommissionEmployee** object

CommissionEmployee.cpp

(4 of 4)

Outline

fig12_06.cpp

(1 of 2)

```
1  // Fig. 12.6: fig12_06.cpp
2  // Testing class CommissionEmployee.
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 #include "CommissionEmployee.h" // CommissionEmployee class definition
12
13 int main()
14 {
15    // instantiate a CommissionEmployee object
16    CommissionEmployee employee(
17       "Sue", "Jones", "222-22-2222", 10000, .06 );
18
19    // set floating-point output formatting
20    cout << fixed << setprecision( 2 );
21
22    // get commission employee data
23    cout << "Employee information obtained by get functions: \n"
24       << "\nFirst name is " << employee.getFirstName()
25       << "\nLast name is " << employee.getLastName()
26       << "\nSocial security number is "
27       << employee.getSocialSecurityNumber()
28       << "\nGross sales is " << employee.getGrossSales()
29       << "\nCommission rate is " << employee.getCommissionRate() << endl;
```

Instantiate **CommissionEmployee** object

Use **CommissionEmployee**'s *get* functions to retrieve the object's instance variable values

Outline

```
30
31    employee.setGrossSales( 8000 ); // set gross sales
32    employee.setCommissionRate( .1 ); // set commission rate
33
34    cout << "\nUpdated employee information output
35       << endl;
36    employee.print(); // display the new employee information
37
38    // display the employee's earnings
39    cout << "\n\nEmployee's earnings: $" << employee.earnings() << endl;
40
41    return 0;
42 } // end main
```

Use **CommissionEmployee**'s *set* functions to change the object's instance variable values

Call object's **print** function to display employee information

Call object's **earnings** function to calculate earnings

(2 of 2)

```
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 Commission Employee.h

(1 of 2)

```
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
5  #define BASEPLUS_H
6
7  #include <string> // C++ standard string class
8  using std::string;
9
10 class BasePlusCommissionEmployee
11 {
12 public:
13    BasePlusCommissionEmployee( const string &, const string &,
14       const string &, double = 0.0, double = 0.0, double = 0.0 );
15
16    void setFirstName( const string & ); // set first name
17    string getFirstName() const; // return first name
18
19    void setLastName( const string & ); // set last name
20    string getLastName() const; // return last name
21
22    void setSocialSecurityNumber( const string & ); // set SSN
23    string getSocialSecurityNumber() const; // return SSN
24
25    void setGrossSales( double ); // set gross sales amount
26    double getGrossSales() const; // return gross sales amount
27
28    void setCommissionRate( double ); // set commission rate
29    double getCommissionRate() const; // return commission rate
```

Constructor takes one more argument, which specifies the base salary

```
30
31    void setBaseSalary( double ); // set base salary
32    double getBaseSalary() const; // return base salary
33
34    double earnings() const; // calculate earnings
35    void print() const; // print BasePlusCommissionEmployee object
36 private:
37    string firstName;
38    string lastName;
39    string socialSecurityNumber;
40    double grossSales; // gross weekly sales
41    double commissionRate; // commission percentage
42    double baseSalary; // base salary
43 }; // end class BasePlusCommissionEmployee
44
45 #endif
```

Define *get* and *set* functions for data member **baseSalary**

Add data member **baseSalary**

BasePlus Commission Employee.h

(2 of 2)

Outline

BasePlus Commission Employee.cpp

(1 of 4)

```
1  // Fig. 12.8: BasePlusCommissionEmployee.cpp
2  // Class BasePlusCommissionEmployee member-function definitions.
3  #include <iostream>
4  using std::cout;
5
6  // BasePlusCommissionEmployee class definition
7  #include "BasePlusCommissionEmployee.h"
8
9  // constructor
10 BasePlusCommissionEmployee::BasePlusCommissionEmployee(
11    const string &first, const string &last, const string &ssn,
12    double sales, double rate, double salary )
13 {
14    firstName = first; // should validate
15    lastName = last; // should validate
16    socialSecurityNumber = ssn; // should validate
17    setGrossSales( sales ); // validate and store gross sales
18    setCommissionRate( rate ); // validate and store commission rate
19    setBaseSalary( salary ); // validate and store base salary
20 } // end BasePlusCommissionEmployee constructor
21
22 // set first name
23 void BasePlusCommissionEmployee::setFirstName( const string &first )
24 {
25    firstName = first; // should validate
26 } // end function setFirstName
```

Constructor takes one more argument, which specifies the base salary

Use function **setBaseSalary** to validate data

Outline

BasePlus Commission Employee.cpp

(2 of 4)

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

```
53 // return social security number
54 string BasePlusCommissionEmployee::getSocialSecurityNumber() const
55 {
56    return socialSecurityNumber;
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 {
68    return grossSales;
69 } // end function getGrossSales
70
71 // set commission rate
72 void BasePlusCommissionEmployee::setCommissionRate( double rate )
73 {
74    commissionRate = ( rate > 0.0 && rate < 1.0 ) ? rate : 0.0;
75 } // end function setCommissionRate
76
77 // return commission rate
78 double BasePlusCommissionEmployee::getCommissionRate() const
79 {
80    return commissionRate;
81 } // end function getCommissionRate
82
```

Outline

BasePlus Commission Employee.cpp

(3 of 4)

Outline

```
83 // set base salary
84 void BasePlusCommissionEmployee::setBaseSalary( double salary )
85 {
86    baseSalary = ( salary < 0.0 ) ? 0.0 : salary;
87 } // end function setBaseSalary
88
89 // return base salary
90 double BasePlusCommissionEmployee::getBaseSalary() const
91 {
92    return baseSalary;
93 } // end function getBaseSalary
94
95 // calculate earnings
96 double BasePlusCommissionEmployee::earnings() const
97 {
98    return baseSalary + ( commissionRate * grossSales );
99 } // end function earnings
100
101// print BasePlusCommissionEmployee object
102void BasePlusCommissionEmployee::print() const
103{
104   cout << "base-salaried commission employee: " << firstName << ' '
105      << lastName << "\nsocial security number: " << socialSecurityNumber
106      << "\ngross sales: " << grossSales
107      << "\ncommission rate: " << commissionRate
108      << "\nbase salary: " << baseSalary;
109} // end function print
```

Function **setBaseSalary** validates data and sets instance variable **baseSalary**

Commission Employee.cpp

Function **getBaseSalary** returns the value of instance variable **baseSalary**

Update function **earnings** to calculate the earnings of a base-salaried commission employee

Update function **print** to display base salary

Outline

fig12_09.cpp

(1 of 3)

```
1  // Fig. 12.9: fig12_09.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
14 int main()
15 {
16    // instantiate BasePlusCommissionEmployee object
17    BasePlusCommissionEmployee
18       employee( "Bob", "Lewis", "333-33-3333", 5000, .04, 300 );
19
20    // set floating-point output formatting
21    cout << fixed << setprecision( 2 );
22
```

Instantiate **BasePlusCommissionEmployee** object

Outline

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

Use **BasePlusCommissionEmployee**'s *get* functions to retrieve the object's instance variable values

(2 of 3)

Use **BasePlusCommissionEmployee**'s **setBaseSalary** function to set base salary

Call object's **print** function to display employee information

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)

12.4.3 Creating a CommissionEmployee-BasePlusCommissionEmployee Inheritance Hierarchy

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

Outline

BasePlus
ssion
yee.h

(1 of 1)

```
1  // Fig. 12.10: BasePlusCommissionEmployee.h
2  // BasePlusCommissionEmployee class derived from class
3  // CommissionEmployee.
4  #ifndef BASEPLUS_H
5  #define BASEPLUS_H
6
7  #include <string> // C++ standard string class
8  using std::string;
9
10 #include "CommissionEmployee.h" // CommissionEmployee class declaration
11
12 class BasePlusCommissionEmployee : public CommissionEmployee
13 {
14 public:
15    BasePlusCommissionEmployee( const string &,
16       const string &, double = 0.0, double = 0.
17
18    void setBaseSalary( double ); // set base salary
19    double getBaseSalary() const; // return base salary
20
21    double earnings() const; // calculate earnings
22    void print() const; // print BasePlusCommissionEmployee object
23 private:
24    double baseSalary; // base salary
25 }; // end class BasePlusCommissionEmployee
26
27 #endif
```

Include the base-class header file in the derived-class header file

Class `BasePlusCommissionEmployee` derives **`public`**ly from class `CommissionEmployee`

Outline

BasePlus Commission Employee.cpp

(1 of 4)

```
1  // Fig. 12.11: BasePlusCommissionEmployee.cpp
2  // Class BasePlusCommissionEmployee member-function definitions.
3  #include <iostream>
4  using std::cout;
5
6  // BasePlusCommissionEmployee class definition
7  #include "BasePlusCommissionEmployee.h"
8
9  // constructor
10 BasePlusCommissionEmployee::BasePlusCommissionEmployee(
11    const string &first, const string &last, const string &ssn,
12    double sales, double rate, double salary )
13    // explicitly call base-class constructor
14    : CommissionEmployee( first, last, ssn, sales, rate )
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 {
22    baseSalary = ( salary < 0.0 ) ? 0.0 : salary;
23 } // end function setBaseSalary
24
25 // return base salary
26 double BasePlusCommissionEmployee::getBaseSalary() const
27 {
28    return baseSalary;
29 } // end function getBaseSalary
```

Initialize base class data member by calling the base-class constructor using base-class initializer syntax

Outline

```
30
31 // calculate earnings
32 double BasePlusCommissionEmployee::earnings() const
33 {
34    // derived class cannot access the base class's private data
35    return baseSalary + ( commissionRate * grossSales );
36 } // end function earnings
37
38 // print BasePlusCommissionEmployee object
39 void BasePlusCommissionEmployee::print() const
40 {
41    // derived class cannot access the base class's private data
42    cout << "base-salaried commission employee: " << firstName << ' '
43       << lastName << "\nsocial security number: " << socialSecurityNumber
44       << "\ngross sales: " << grossSales
45       << "\ncommission rate: " << commissionRate
46       << "\nbase salary: " << baseSalary;
47 } // end function print
```

BasePlus Commission

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

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