C++

Inheritance
Inheritance

- Introduction
- Base Classes and Derived Classes
- Five Examples of Base Class and Derived Class Relationships
- Constructors and Destructors in Derived Classes
Inheritance is a form of software reuse where a class is created that absorbs an existing class’s data and behaviors and enhances them with new capabilities.

The new class, the derived class, inherits the members of the existing class, the base class.
A direct base class is the base class from which a derived class explicitly inherits.

An indirect base class is inherited from two or more levels up in the class hierarchy.

In single inheritance, a class is derived from one base class.

With multiple inheritance, a derived class inherits from multiple base classes.
C++ offers three types of inheritance:

- **public**: every object of a derived class is also an object of that derived class’s base class. {Note, base-class objects are NOT objects of their derived classes.}

- **private**: is essentially an alternative to composition.

- **protected**: is rarely used.
Member functions of a derived class cannot directly access private members of the base class.
C++ Abstractions

- **is-a** relationship:: inheritance
  - e.g., derived class, *car*, is a base class, *vehicle*.

- **has-a** relationship:: composition
  - e.g. object *employee* has an object *birthdate*. 
Base Classes and Derived Classes

- Base class typically represents larger set of objects than derived classes.

Example

Base class: vehicle

Includes cars, trucks, boats, bicycles, etc.

Derived class: car

a smaller, more-specific subset of vehicles

- Inheritance relationships form treelike hierarchical structures.
Fig. 23.2 Inheritance Hierarchy for University Community Member

Systems Programming: Inheritance
Base Classes and Derived Classes

- **public** inheritance specified by:
  
  ```
  Class Employee : public CommunityMember
  ```

- **Class Employee** inherits from class **CommunityMember**

- Base class **private** members are not accessible directly, but they are inherited.
  - Manipulated through inherited **public** member functions.

- Base class **public** and **protected** members
  - Are inherited with original member access.

- **friend** functions
  - Are not inherited.
protected Members

- A base class’s protected members can be accessed within the body of that base class by members and friends of that base class and by members and friends of any class derived from that base class.

- By simply using member names, derived-class member functions can refer to public and protected members of the base class.

- When a derived-class member function redefines a base-class member function, by preceding the base-class member with the base-class name and the binary scope resolution operator (::), the derived-class can access the base-class member.
Five Examples of Base Class and Derived Class Relationships

1. Create a `CommissionEmployee` class with private data members: First name, last name, SSN, commission rate, gross sale amount.

2. Create a `BasePlusCommissionEmployee` class without inheritance with private data members: First name, last name, SSN, commission rate, gross sale amount and base salary.
Five Examples of Base Class and Derived Class Relationships

3. Create a `CommissionEmployee` inheritance hierarchy with private members.

4. Create a `CommissionEmployee` inheritance hierarchy with protected members.

5. Create a `CommissionEmployee` inheritance hierarchy with private members but access through public member functions.
**Example 1: CommissionEmployee Class**

- **CommissionEmployee header file**
  - Specify public services:
    - Constructor
    - *get* and *set* functions
    - Member functions *earnings* and *print*

- **CommissionEmployee source code file**
  - Specify member-function definitions.
// Fig. 23.4: CommissionEmployee.h
// CommissionEmployee class definition represents a commission employee.
#ifndef COMMISSION_H
#define COMMISSION_H

#include <string> // C++ standard string class
using std::string;

class CommissionEmployee
{
public:
    CommissionEmployee( const string &, const string &, const string &, double = 0.0, double = 0.0 );
    void setFirstName( const string & ); // set first name
    string getFirstName() const; // return first name

    void setLastName( const string & ); // set last name
    string getLastName() const; // return last name

    void setSocialSecurityNumber( const string & ); // set SSN
    string getSocialSecurityNumber() const; // return SSN

    void setGrossSales( double ); // set gross sales amount
    double getGrossSales() const; // return gross sales amount

    void setCommissionRate( double ); // set commission rate (percentage)
    double getCommissionRate() const; // return commission rate

}; // end class CommissionEmployee
Example 1: CommissionEmployee Class

```cpp
class CommissionEmployee
{
public:
    double earnings() const; // calculate earnings
    void print() const; // print CommissionEmployee object
private:
    string firstName;
    string lastName;
    string socialSecurityNumber;
    double grossSales; // gross weekly sales
    double commissionRate; // commission percentage
}; // end class CommissionEmployee
```

Declare private data members
Example 1: CommissionEmployee Class

```cpp
// Fig. 23.5: CommissionEmployee.cpp
// Class CommissionEmployee member-function definitions.
#include <iostream>
#include "CommissionEmployee.h" // CommissionEmployee class definition

// constructor
CommissionEmployee::CommissionEmployee(
    const string &first, const string &last, const string &ssn,
    double sales, double rate )
{
    firstName = first; // should validate
    lastName = last;  // should validate
    socialSecurityNumber = ssn; // should validate
    setGrossSales( sales ); // validate and store gross sales
    setCommissionRate( rate ); // validate and store commission rate
} // end CommissionEmployee constructor

// set first name
void CommissionEmployee::setFirstName( const string &first )
{
    firstName = first; // should validate
} // end function setFirstName

// return first name
string CommissionEmployee::getFirstName() const
{
    return firstName;
} // end function getFirstName
```

Initialize data members
Example 1: CommissionEmployee Class

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 grossSales amount
Example 1: CommissionEmployee Class

```cpp
// return gross sales amount
double CommissionEmployee::getGrossSales() const
{
    return grossSales;
} // end function getGrossSales

// set commission rate
void CommissionEmployee::setCommissionRate( double rate )
{
    commissionRate = ( rate > 0.0 && rate < 1.0 ) ? rate : 0.0;
} // end function setCommissionRate

// return commission rate
double CommissionEmployee::getCommissionRate() const
{
    return commissionRate;
} // end function getCommissionRate
```
Example 1: `CommissionEmployee` Class

```cpp
// calculate earnings
double CommissionEmployee::earnings() const
{
    return commissionRate * grossSales;
} // end function earnings

// print CommissionEmployee object
void CommissionEmployee::print() const
{
    cout << "commission employee: " << firstName << ' ' << lastName << "
social security number: " << socialSecurityNumber << "
gross sales: " << grossSales << "
commission rate: " << commissionRate;
} // end function print
```

Function `earnings` calculates earnings

Function `print` displays `CommissionEmployee` object
Example 1: CommissionEmployee Class

```cpp
// Fig. 23.6: fig23_06.cpp
// Testing class CommissionEmployee.
#include <iostream>
using std::cout;
using std::endl;
using std::fixed;

#include <iomanip>
using std::setprecision;

#include "CommissionEmployee.h" // CommissionEmployee class definition

int main()
{
    // instantiate a CommissionEmployee object
    CommissionEmployee employee(
        "Sue", "Jones", "222-22-2222", 10000, .06 );

    // set floating-point output formatting
    cout << fixed << setprecision( 2 );

    // get commission employee data
    cout << "Employee information obtained by get functions: \n"
        << "\nFirst name is " << employee.getFirstName()
        << "\nLast name is " << employee.getLastName()
        << "\nSocial security number is "
        << employee.getSocialSecurityNumber()
        << "\nGross sales is " << employee.getGrossSales()
        << "\nCommission rate is " << employee.getCommissionRate() << endl;
}
```

- Instantiate `CommissionEmployee` object
- Use `CommissionEmployee`'s `get` functions to retrieve the object’s instance variable values

© 2007 Pearson Ed - All rights reserved.
Example 1: CommissionEmployee Class

```cpp
employee.setGrossSales(8000); // set gross sales
employee.setCommissionRate(0.1); // set commission rate

cout << "Updated employee information output by print function: " << endl;
employee.print(); // display the new employee information

cout << "Employee's earnings: $" << employee.earnings() << endl;

return 0;
} // end main
```

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

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
Example 2: BasePlusCommissionEmployee Class

- **Class** BasePlusCommissionEmployee
  - Much of the code is similar to CommissionEmployee
    - **private** data members
    - **public** member functions
    - constructor
  - Additions
    - **private** data member baseSalary
    - member functions setBaseSalary and getBaseSalary
Example 2:  
BasePlusCommissionEmployee Class

// Fig. 23.7: BasePlusCommissionEmployee.h
// BasePlusCommissionEmployee class definition represents an employee
// that receives a base salary in addition to commission.
#ifndef BASEPLUS_H
#define BASEPLUS_H

#include <string> // C++ standard string class
using std::string;

class BasePlusCommissionEmployee
{
public:
    BasePlusCommissionEmployee( const string &, const string &, const string &, double = 0.0, double = 0.0, double = 0.0 );

    void setFirstName( const string & ); // set first name
    string getFirstName() const; // return first name

    void setLastName( const string & ); // set last name
    string getLastName() const; // return last name

    void setSocialSecurityNumber( const string & ); // set SSN
    string getSocialSecurityNumber() const; // return SSN

    void setGrossSales( double ); // set gross sales amount
    double getGrossSales() const; // return gross sales amount

    void setCommissionRate( double ); // set commission rate
    double getCommissionRate() const; // return commission rate
};

Constructor takes one more argument, which specifies the base salary.
Example 2:
BasePlusCommissionEmployee Class

```cpp
30  void setBaseSalary( double ); // set base salary
31  double getBaseSalary() const; // return base salary
32
33  double earnings() const; // calculate earnings
34  void print() const; // print BasePlusCommissionEmployee object
35
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`
Example 2: BasePlusCommissionEmployee Class

// Fig. 23.8: BasePlusCommissionEmployee.cpp
// Class BasePlusCommissionEmployee member-function definitions.
#include <iostream>
using std::cout;

// BasePlusCommissionEmployee class definition
#include "BasePlusCommissionEmployee.h"

// constructor
BasePlusCommissionEmployee::BasePlusCommissionEmployee(
    const string &first, const string &last, const string &ssn,
    double sales, double rate, double salary )
{
    firstName = first; // should validate
    lastName = last; // should validate
    socialSecurityNumber = ssn; // should validate
    setGrossSales( sales ); // validate and store gross sales
    setCommissionRate( rate ); // validate and store commission rate
    setBaseSalary( salary ); // validate and store base salary
} // end BasePlusCommissionEmployee constructor

// set first name
void BasePlusCommissionEmployee::setFirstName( const string &first )
{
    firstName = first; // should validate
} // end function setFirstName

Constructor takes one more argument, which specifies the base salary
Use function setBaseSalary to validate data
Example 2:
BasePlusCommissionEmployee Class

```cpp
// return first name
string BasePlusCommissionEmployee::getFirstName() const
{
    return firstName;
} // end function getFirstName

// set last name
void BasePlusCommissionEmployee::setLastName( const string &last )
{
    lastName = last; // should validate
} // end function setLastName

// return last name
string BasePlusCommissionEmployee::getLastName() const
{
    return lastName;
} // end function getLastName

// set social security number
void BasePlusCommissionEmployee::setSocialSecurityNumber( const string &ssn )
{
    socialSecurityNumber = ssn; // should validate
} // end function setSocialSecurityNumber
```
Example 2:
BasePlusCommissionEmployee Class

```cpp
// return social security number
string BasePlusCommissionEmployee::getSocialSecurityNumber() const
{
    return socialSecurityNumber;
} // end function getSocialSecurityNumber

// set gross sales amount
void BasePlusCommissionEmployee::setGrossSales(double sales)
{
    grossSales = (sales < 0.0) ? 0.0 : sales;
} // end function setGrossSales

// return gross sales amount
double BasePlusCommissionEmployee::getGrossSales() const
{
    return grossSales;
} // end function getGrossSales

// set commission rate
void BasePlusCommissionEmployee::setCommissionRate(double rate)
{
    commissionRate = (rate > 0.0 && rate < 1.0) ? rate : 0.0;
} // end function setCommissionRate

// return commission rate
double BasePlusCommissionEmployee::getCommissionRate() const
{
    return commissionRate;
} // end function getCommissionRate
```

© 2007 Pearson Ed - All rights reserved.
Example 2: BasePlusCommissionEmployee Class

```cpp
// set base salary
void BasePlusCommissionEmployee::setBaseSalary( double salary )
{
    baseSalary = ( salary < 0.0 ) ? 0.0 : salary;
} // end function setBaseSalary

// return base salary
double BasePlusCommissionEmployee::getBaseSalary() const
{
    return baseSalary;
} // end function getBaseSalary

// calculate earnings
double BasePlusCommissionEmployee::earnings() const
{
    return baseSalary + ( commissionRate * grossSales );
} // end function earnings

// print BasePlusCommissionEmployee object
void BasePlusCommissionEmployee::print() const
{
    cout << "base-salaried commission employee: " << firstName << ' ';
    cout << lastName << " social security number: " << socialSecurityNumber << " gross sales: " << grossSales << " commission rate: " << commissionRate << " base salary: " << baseSalary;
} // end function print
```

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

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 `baseSalary`
Example 2: BasePlusCommissionEmployee Class

```cpp
// Fig. 23.9: fig23_09.cpp
// Testing class BasePlusCommissionEmployee.
#include <iostream>
using std::cout;
using std::endl;
using std::fixed;

#include <iomanip>
using std::setprecision;

// BasePlusCommissionEmployee class definition
#include "BasePlusCommissionEmployee.h"

int main()
{
    // Instantiate BasePlusCommissionEmployee object
    BasePlusCommissionEmployee employee( "Bob", "Lewis", "333-33-3333", 5000, .04, 300 );
    // set floating-point output formatting
    cout << fixed << setprecision( 2 );
    // Instantiate BasePlusCommissionEmployee object
    BasePlusCommissionEmployee employee( "Bob", "Lewis", "333-33-3333", 5000, .04, 300 );
    // set floating-point output formatting
    cout << fixed << setprecision( 2 );
```
Example 2: BasePlusCommissionEmployee Class

```cpp
// get commission employee data
cout << "Employee information obtained by get functions: \n"
<< "\nFirst name is " << employee.getFirstName()
<< "\nLast name is " << employee.getLastName()
<< "\nSocial security number is "
employee.getSocialSecurityNumber()
<< "\nGross sales is " << employee.getGrossSales()
<< "\nCommission rate is " << employee.getCommissionRate()
<< "\nBase salary is " << employee.getBaseSalary() << endl;

employee.setBaseSalary( 1000 ); // set base salary

cout << "Updated employee information output by print function: 
" << endl;
employee.print(); // display the new employee information

// display the employee's earnings
cout << "Employee's earnings: $" << employee.earnings() << endl;
```

Use BasePlusCommissionEmployee’s `get` functions to retrieve the object’s instance variable values

Use BasePlusCommissionEmployee’s `setBaseSalary` function to set `baseSalary`

Call object’s `print` function to display employee information

Call object’s `earnings` function to calculate employee’s earnings

© 2007 Pearson Ed - All rights reserved.
Example 2:
BasePlusCommissionEmployee Class

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
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.
Example 3:

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`.

Inheritance Hierarchy
Example 3:

**CommissionEmployee - BasePlusCommissionEmployee**

Inheritance Hierarchy

```
#include <string> // C++ standard string class
using std::string;

#include "CommissionEmployee.h" // CommissionEmployee class declaration

class BasePlusCommissionEmployee : public CommissionEmployee
{
public:
    BasePlusCommissionEmployee( const string &name, const string &salesRep, const string &region, double baseSalary = 0.0, double commissionRate = 0.0, double grossSales = 0.0 );

    void setBaseSalary( double ); // set base salary
    double getBaseSalary() const; // return base salary

    double earnings() const; // calculate earnings
    void print() const; // print BasePlusCommissionEmployee object

private:
    double baseSalary; // base salary
}; // end class BasePlusCommissionEmployee
```

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

Class BasePlusCommissionEmployee derives publicly from class CommissionEmployee
Example 3:

```
// Fig. 23.11: BasePlusCommissionEmployee.cpp
// Class BasePlusCommissionEmployee member-function definitions.
#include <iostream>
using std::cout;

// BasePlusCommissionEmployee class definition
#include "BasePlusCommissionEmployee.h"

// constructor
BasePlusCommissionEmployee::BasePlusCommissionEmployee(
    const string &first, const string &last, const string &ssn,
    double sales, double rate, double salary )
    : CommissionEmployee( first, last, ssn, sales, rate )
{ 
    setBaseSalary( salary ); // validate and store base salary
} // end BasePlusCommissionEmployee constructor

// set base salary
void BasePlusCommissionEmployee::setBaseSalary( double salary )
{
    baseSalary = ( salary < 0.0 ) ? 0.0 : salary;
} // end function setBaseSalary

// return base salary
double BasePlusCommissionEmployee::getBaseSalary() const
{
    return baseSalary;
} // end function getBaseSalary
```

Initialize base class data member by calling the base-class constructor using base-class initializer syntax
Example 3:

```cpp
30 // calculate earnings
31 double BasePlusCommissionEmployee::earnings() const
32 {
33   // derived class cannot access the base class's private data
34   return baseSalary + (commissionRate * grossSales);
35 } // end function earnings
36
37 // print BasePlusCommissionEmployee object
38 void BasePlusCommissionEmployee::print() const
39 {
40   // derived class cannot access the base class's private data
41   cout << "base-salaried commission employee: " << firstName << ' ';
42   << lastName << "social security number: " << socialSecurityNumber
43   << "gross sales: " << grossSales
44   << "commission rate: " << commissionRate
45   << "base salary: " << baseSalary;
46 } // end function print
```
Example 3:

\[ \text{a CommissionEmployee} \supset \text{BasePlusCommissionEmployee} \]

Inheritance Hierarchy

```
c:\examples\ch23\Fig23_10_11\BasePlusCommission-Employee.cpp(35) :
  error C2248: 'CommissionEmployee::commissionRate'
  cannot access private member declared in class 'CommissionEmployee'
  C:\examples\ch23\Fig23_10_11\CommissionEmployee.h(37) :
    see declaration of 'CommissionEmployee::commissionRate'
  C:\examples\ch23\Fig23_10_11\CommissionEmployee.h(10) :
    see declaration of 'CommissionEmployee'

c:\examples\ch23\Fig23_10_11\BasePlusCommission-Employee.cpp(35) :
  error C2248: 'CommissionEmployee::grossSales'
  cannot access private member declared in class 'CommissionEmployee'
  C:\examples\ch23\Fig23_10_11\CommissionEmployee.h(36) :
    see declaration of 'CommissionEmployee::grossSales'
  C:\examples\ch23\Fig23_10_11\CommissionEmployee.h(10) :
    see declaration of 'CommissionEmployee'

c:\examples\ch23\Fig23_10_11\BasePlusCommission-Employee.cpp(42) :
  error C2248: 'CommissionEmployee::firstName'
  cannot access private member declared in class 'CommissionEmployee'
  C:\examples\ch23\Fig23_10_11\CommissionEmployee.h(33) :
    see declaration of 'CommissionEmployee::firstName'
  C:\examples\ch23\Fig23_10_11\CommissionEmployee.h(10) :
    see declaration of 'CommissionEmployee'
```
Example 3:

- `CommissionEmployee` - `BasePlusCommissionEmployee`

Inheritance Hierarchy

```cpp
C:\examples\ch23\Fig23_10_11\BasePlusCommissionEmployee.cpp(43)
: error C2248: 'CommissionEmployee::lastName'
  cannot access private member declared in class 'CommissionEmployee'
C:\examples\ch23\Fig23_10_11\CommissionEmployee.h(34)
: see declaration of 'CommissionEmployee::lastName'
C:\examples\ch23\Fig23_10_11\CommissionEmployee.h(10)
: see declaration of 'CommissionEmployee'

C:\examples\ch23\Fig23_10_11\BasePlusCommissionEmployee.cpp(43)
: error C2248: 'CommissionEmployee::socialSecurityNumber'
  cannot access private member declared in class 'CommissionEmployee'
C:\examples\ch23\Fig23_10_11\CommissionEmployee.h(35)
: see declaration of 'CommissionEmployee::socialSecurityNumber'
C:\examples\ch23\Fig23_10_11\CommissionEmployee.h(10)
: see declaration of 'CommissionEmployee'

C:\examples\ch23\Fig23_10_11\BasePlusCommissionEmployee.cpp(44)
: error C2248: 'CommissionEmployee::grossSales'
  cannot access private member declared in class 'CommissionEmployee'
C:\examples\ch23\Fig23_10_11\CommissionEmployee.h(36)
: see declaration of 'CommissionEmployee::grossSales'
C:\examples\ch23\Fig23_10_11\CommissionEmployee.h(10)
: see declaration of 'CommissionEmployee'

C:\examples\ch23\Fig23_10_11\BasePlusCommissionEmployee.cpp(45)
: error C2248: 'CommissionEmployee::commissionRate'
  cannot access private member declared in class 'CommissionEmployee'
C:\examples\ch23\Fig23_10_11\CommissionEmployee.h(37)
: see declaration of 'CommissionEmployee::commissionRate'
C:\examples\ch23\Fig23_10_11\CommissionEmployee.h(10)
: see declaration of 'CommissionEmployee'
```
Example 3:

`CommissionEmployee - BasePlusCommissionEmployee`

Inheritance Hierarchy

- The base class header file must be included in the derived class header file for three reasons, the compiler must
  - Know that the base class exists.
  - Know the size of inherited data members.
  - Ensure that inherited class members are used properly.
Example 4:

Inheritance Hierarchy using Protected Data

- Uses **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.
Example 4:

- CommissionEmployee - BasePlusCommissionEmployee

Inheritance Hierarchy using Protected Data

```cpp
// Fig. 23.12: CommissionEmployee.h

// CommissionEmployee class definition with protected data.
#ifndef COMMISSION_H
#define COMMISSION_H

#include <string> // C++ standard string class

using std::string;

class CommissionEmployee
{
    public:
    CommissionEmployee( const string & , const string & , const string & ,
                        double = 0.0, double = 0.0 );

    void setFirstName( const string & ); // set first name
    string getFirstName() const; // return first name

    void setLastName( const string & ); // set last name
    string getLastName() const; // return last name

    void setSocialSecurityNumber( const string & ); // set SSN
    string getSocialSecurityNumber() const; // return SSN

#endif // COMMISSION_H
```

© 2007 Pearson Ed - All rights reserved.


Example 4:

```
void setGrossSales( double ); // set gross sales amount

double getGrossSales() const; // return gross sales amount

void setCommissionRate( double ); // set commission rate

double getCommissionRate() const; // return commission rate


double earnings() const; // calculate earnings

void print() const; // print CommissionEmployee object

protected:

   string firstName;
   string lastName;
   string socialSecurityNumber;

double grossSales; // gross weekly sales

double commissionRate; // commission percentage

}; // end class CommissionEmployee
```

Declare protected data
Example 4:

### CommissionEmployee - BasePlusCommissionEmployee

Inheritance Hierarchy using Protected Data

```cpp
// Fig. 23.13: CommissionEmployee.cpp
// Class CommissionEmployee member-function definitions.
#include <iostream>
using std::cout;

#include "CommissionEmployee.h" // CommissionEmployee class definition

// constructor
CommissionEmployee::CommissionEmployee(
    const string &first, const string &last, const string &ssn,
    double sales, double rate )
{
    firstName = first; // should validate
    lastName = last; // should validate
    socialSecurityNumber = ssn; // should validate
    setGrossSales( sales ); // validate and store gross sales
    setCommissionRate( rate ); // validate and store commission rate
} // end CommissionEmployee constructor

// set first name
void CommissionEmployee::setFirstName( const string &first )
{
    firstName = first; // should validate
} // end function setFirstName

// return first name
string CommissionEmployee::getFirstName() const
{
    return firstName;
} // end function getFirstName
```
Example 4:

```
// set last name
void CommissionEmployee::setLastName( const string &last )
{
    lastName = last; // should validate
} // end function setLastName

// return last name
string CommissionEmployee::getLastName() const
{
    return lastName;
} // end function getLastName

// set social security number
void CommissionEmployee::setSocialSecurityNumber( const string &ssn )
{
    socialSecurityNumber = ssn; // should validate
} // end function setSocialSecurityNumber

// return social security number
string CommissionEmployee::getSocialSecurityNumber() const
{
    return socialSecurityNumber;
} // end function getSocialSecurityNumber

// set gross sales amount
void CommissionEmployee::setGrossSales( double sales )
{
    grossSales = ( sales < 0.0 ) ? 0.0 : sales;
} // end function setGrossSales
```
Example 4:

`CommissionEmployee` - `BasePlusCommissionEmployee`

Inheritance Hierarchy using Protected Data

61 // return gross sales amount
62 double CommissionEmployee::getGrossSales() const
63 {
   return grossSales;
} // end function getGrossSales
67 // set commission rate
69 void CommissionEmployee::setCommissionRate( double rate )
70 {
   commissionRate = ( rate > 0.0 && rate < 1.0 ) ? rate : 0.0;
} // end function setCommissionRate
73 // return commission rate
75 double CommissionEmployee::getCommissionRate() const
76 {
   return commissionRate;
} // end function getCommissionRate
79 // calculate earnings
81 double CommissionEmployee::earnings() const
82 {
   return commissionRate * grossSales;
} // end function earnings
Example 4:

Inheritance Hierarchy using Protected Data

```cpp
85 // print CommissionEmployee object
86 void CommissionEmployee::print() const
87 {
88     cout << "commission employee: " << firstName << ' ' << lastName
89     << "social security number: " << socialSecurityNumber
90     << "gross sales: " << grossSales
91     << "commission rate: " << commissionRate;
92 } // end function print
```
Example 4:

- **CommissionEmployee - BasePlusCommissionEmployee**

Inheritance Hierarchy using Protected Data

```
// Fig. 23.14: BasePlusCommissionEmployee.h
// BasePlusCommissionEmployee class derived from class CommissionEmployee.
#ifndef BASEPLUS_H
#define BASEPLUS_H

#include <string> // C++ standard string class
using std::string;

#include "CommissionEmployee.h" // CommissionEmployee class declaration

class BasePlusCommissionEmployee : public CommissionEmployee
{
public:
    BasePlusCommissionEmployee( const string &, const string &, const string &, double = 0.0, double = 0.0, double = 0.0 ); // set base salary
    void setBaseSalary( double ); // set base salary
    double getBaseSalary() const; // return base salary

    double earnings() const; // calculate earnings
    void print() const; // print BasePlusCommissionEmployee object

private:
    double baseSalary; // base salary
}; // end class BasePlusCommissionEmployee

#endif
```

BasePlusCommissionEmployee still inherits publicly from CommissionEmployee
Call base-class constructor using base-class initializer syntax
Example 4:

- **CommissionEmployee** - **BasePlusCommissionEmployee**

Inheritance Hierarchy using Protected Data

```cpp
// calculate earnings
double BasePlusCommissionEmployee::earnings() const
{
    // can access protected data of base class
    return baseSalary + (commissionRate * grossSales);
} // end function earnings

// print BasePlusCommissionEmployee object
void BasePlusCommissionEmployee::print() const
{
    // can access protected data of base class
    cout << "base-salaried commission employee: " << firstName << ' ';
    << lastName << "\nsocial security number: " << socialSecurityNumber
    << "\ngross sales: " << grossSales
    << "\ncommission rate: " << commissionRate
    << "\nbase salary: " << baseSalary;
} // end function print
```

Directly access base class’s protected data
```cpp
// Fig. 23.16: fig23_16.cpp
// Testing class BasePlusCommissionEmployee.
#include <iostream>
#include <iomanip>
using namespace std;

// BasePlusCommissionEmployee class definition
#include "BasePlusCommissionEmployee.h"

int main()
{
    // Instantiate BasePlusCommissionEmployee object
    BasePlusCommissionEmployee employee( "Bob", "Lewis", "333-33-3333", 5000, .04, 300 );
    // Set floating-point output formatting
    cout << fixed << setprecision( 2 );
}```
Example 4:

```cpp
// get commission employee data
cout << "Employee information obtained by get functions: \n"
  << "\nFirst name is " << employee.firstName() 
  << "\nLast name is " << employee.lastName() 
  << "\nSocial security number is " 
  << employee.socialSecurityNumber() 
  << "\nGross sales is " << employee.grossSales() 
  << "\nCommission rate is " << employee.commissionRate() 
  << "\nBase salary is " << employee.baseSalary() << endl;
employee.setBaseSalary(1000); // set base salary

// display the updated employee information output by print function: \n
  << endl;
employee.print(); // display the new employee information

// display the employee's earnings
cout << "\nEmployee's earnings: \" << employee.earnings() << endl;
return 0;
```

Inheritance Hierarchy using Protected Data
Example 4:

- CommissionEmployee
- BasePlusCommissionEmployee

Inheritance Hierarchy using Protected Data

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
Using protected data members

- **Advantages**
  - Derived class can modify values directly.
  - Avoid *set/get* function 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.
    - This is fragile (brittle) software.
Example 5:

- Reexamine hierarchy
  - Use the best software engineering practice
    - Declare data members as `private`.
    - Provide `public get` and `set` functions.
    - Use `get` function to obtain values of data members.
Example 5:

Inheritance Hierarchy using Private Data

```cpp
#include <string>
using std::string;

class CommissionEmployee {
public:
    CommissionEmployee(const string &first, const string &last, const string &ssn, double grossSales = 0.0, double commissionRate = 0.0);

    void setFirstName(const string &); // set first name
    string getFirstName() const; // return first name

    void setLastName(const string &); // set last name
    string getLastName() const; // return last name

    void setSocialSecurityNumber(const string &); // set SSN
    string getSocialSecurityNumber() const; // return SSN

    void setGrossSales(double); // set gross sales amount
    double getGrossSales() const; // return gross sales amount

    void setCommissionRate(double); // set commission rate
    double getCommissionRate() const; // return commission rate
};
```

© 2007 Pearson Ed - All rights reserved.
Example 5:

```
double earnings() const; // calculate earnings

void print() const; // print CommissionEmployee object

private:

    string firstName;
    string lastName;
    string socialSecurityNumber;
    double grossSales; // gross weekly sales
    double commissionRate; // commission percentage

}; // end class CommissionEmployee
```

Declare private data
Example 5:

Inheritance Hierarchy using Private Data

Use member initializers to set the values of members firstName, lastName and socialSecurityNumber.
Example 5:

Inheritance Hierarchy using Private Data

// set last name
void CommissionEmployee::setLastName( const string &last )
{
    lastName = last; // should validate
} // end function setLastName

// return last name
string CommissionEmployee::getLastName() const
{
    return lastName;
} // end function getLastName

// set social security number
void CommissionEmployee::setSocialSecurityNumber( const string &ssn )
{
    socialSecurityNumber = ssn; // should validate
} // end function setSocialSecurityNumber

// return social security number
string CommissionEmployee::getSocialSecurityNumber() const
{
    return socialSecurityNumber;
} // end function getSocialSecurityNumber

// set gross sales amount
void CommissionEmployee::setGrossSales( double sales )
{
    grossSales = ( sales < 0.0 ) ? 0.0 : sales;
} // end function setGrossSales
Example 5:  

\texttt{CommissionEmployee}\rightarrow \texttt{BasePlusCommissionEmployee} 

\textbf{Inheritance Hierarchy using Private Data}

59
60 // return gross sales amount
61 \texttt{double CommissionEmployee::getGrossSales()} \texttt{const}
62 {
63 \quad \texttt{return grossSales;}
64 } // end function getGrossSales
65
66 // set commission rate
67 \texttt{void CommissionEmployee::setCommissionRate( double rate )}
68 {
69 \quad \texttt{commissionRate = ( rate > 0.0 \&\& rate < 1.0 ) ? rate : 0.0;}
70 } // end function setCommissionRate
71
72 // return commission rate
73 \texttt{double CommissionEmployee::getCommissionRate()} \texttt{const}
74 {
75 \quad \texttt{return commissionRate;}
76 } // end function getCommissionRate
77
78 // calculate earnings
79 \texttt{double CommissionEmployee::earnings()} \texttt{const}
80 {
81 \quad \texttt{return getCommissionRate()} * \texttt{getGrossSales();}
82 } // end function earnings
83

\textbf{Use} \textit{get} \textbf{functions to obtain the values of data members}
Example 5:

Inheritance Hierarchy using Private Data

```cpp
// print CommissionEmployee object
void CommissionEmployee::print() const
{
    cout << "commission employee: " <<
    getFirstName() << ' ' << getLastName()
    << "social security number: " << getSocialSecurityNumber()
    << "gross sales: " << getGrossSales()
    << "commission rate: " << getCommissionRate();
} // end function print
```

Use *get* functions to obtain the values of data members.
Example 5:

a CommissionEmployee - BasePlusCommissionEmployee

Inheritance Hierarchy using Private Data

```cpp
// Fig. 23.19: BasePlusCommissionEmployee.h
// BasePlusCommissionEmployee class derived from class CommissionEmployee.
#ifndef BASEPLUS_H
#define BASEPLUS_H

#include <string> // C++ standard string class
using std::string;

#include "CommissionEmployee.h" // CommissionEmployee class declaration

class BasePlusCommissionEmployee : public CommissionEmployee
{
public:
    BasePlusCommissionEmployee( const string &, const string &, const string &, double = 0.0, double = 0.0, double = 0.0 );

    void setBaseSalary( double ); // set base salary
    double getBaseSalary() const; // return base salary

    double earnings() const; // calculate earnings
    void print() const; // print BasePlusCommissionEmployee object

private:
    double baseSalary; // base salary
}; // end class BasePlusCommissionEmployee

#endif
```

© 2007 Pearson Ed - All rights reserved.
// Fig. 23.20: BasePlusCommissionEmployee.cpp
// Class BasePlusCommissionEmployee member-function definitions.
#include <iostream>
#include "BasePlusCommissionEmployee.h"
using std::cout;

// BasePlusCommissionEmployee class definition
const string &first, const string &last, const string &ssn,
double sales, double rate, double salary )
// explicitly call base-class constructor
: CommissionEmployee( first, last, ssn, sales, rate )
{
    setBaseSalary( salary ); // validate and store base salary
    } // end BasePlusCommissionEmployee constructor

// set base salary
void BasePlusCommissionEmployee::setBaseSalary( double salary )
{
    baseSalary = ( salary < 0.0 ) ? 0.0 : salary;
} // end function setBaseSalary

// return base salary
double BasePlusCommissionEmployee::getBaseSalary() const
{
    return baseSalary;
} // end function getBaseSalary

Example 5:
CommissionEmployee - BasePlusCommissionEmployee
Inheritance Hierarchy using Private Data
30
31 // calculate earnings
32 double BasePlusCommissionEmployee::earnings() const
33 {
34     return getBaseSalary() + CommissionEmployee::earnings();
35 } // end function earnings
36
37 // print BasePlusCommissionEmployee object
38 void BasePlusCommissionEmployee::print() const
39 {
40     cout << "base-salaried \nbase salary: " << getBaseSalary();
41
42     // invoke CommissionEmployee's print function
43     CommissionEmployee::print();
44
45 } // end function print

Invoke base class’s earnings function

Invoke base class’s print function
Example 5:

CommissionEmployee = BasePlusCommissionEmployee

Inheritance Hierarchy using Private Data

```
1 // Fig. 23.21: fig23_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```

© 2007 Pearson Ed - All rights reserved.
Example 5:

```
int main()
{
    // Instantiate BasePlusCommissionEmployee object
    BasePlusCommissionEmployee employee( "Bob", "Lewis", "333-33-3333", 5000, .04, 300 );

    // Set floating-point output formatting
    cout << fixed << setprecision( 2 );

    // Get commission employee data
    cout << "Employee information obtained by get functions: \n" << endl;
    cout << "First name is " << employee.getFirstName() << endl;
    cout << "Last name is " << employee.getLastName() << endl;
    cout << "Social security number is " << employee.getSocialSecurityNumber() << endl;
    cout << "Gross sales is " << employee.getGrossSales() << endl;
    cout << "Commission rate is " << employee.getCommissionRate() << endl;
    cout << "Base salary is " << employee.getBaseSalary() << endl;

    employee.setBaseSalary( 1000 ); // Set base salary
    cout << "Updated employee information output by print function: \n" << endl;
    employee.print(); // Display the new employee information

    // Display the employee's earnings
    cout << "Employee's earnings: $" << employee.earnings() << endl;

    return 0;
} // End main
```

Create BasePlusCommissionEmployee object

Use inherited get methods to access base class private members

Use BasePlusCommissionEmployee get method to access private member

Use BasePlusCommissionEmployee set method to modify private data member baseSalary

© 2007 Pearson Ed - All rights reserved.

Systems Programming: Inheritance
Example 5:

```
Inheritance Hierarchy using Private Data

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
```
23.5 Constructors and Destructors in Derived Classes

- **Instantiating derived-class object**
  - Chain of constructor calls
    - Derived-class constructor invokes base class constructor either implicitly (via a base-class member initializer) or explicitly (by calling the base classes default constructor).

- **Base of inheritance hierarchy**
  - The last constructor called in an inheritance chain is at the base of the hierarchy and this constructor is the first constructor body to finish executing.

Example: `CommissionEmployee/BasePlusCommissionEmployee` hierarchy
- `CommissionEmployee` constructor called last.
- `CommissionEmployee` constructor body executes first and initializes private 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.
Constructors and Destructors in Derived Classes

- Destroying derived-class objects
  - Chain of destructor calls
    - Reverse order of constructor chain
    - Destructor of derived-class called first.
    - Destructor of next base class up hierarchy is called next.
    - This continues up hierarchy until the final base class is reached.
      - After final base-class destructor, the object is removed from memory.
  - Base-class constructors, destructors, and overloaded assignment operators are **not** inherited by derived classes.
Software Engineering Observation 23.8

- 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.
Constructors and Destructors in Derived Classes

```cpp
// Fig. 23.22: CommissionEmployee.h
// CommissionEmployee class definition represents a commission employee.
#ifndef COMMISSION_H
#define COMMISSION_H

#include <string> // C++ standard string class
using std::string;

class CommissionEmployee
{
    public:
        CommissionEmployee( const string & first, const string & last, const string & social, double gross = 0.0, double commission = 0.0 );
        ~CommissionEmployee(); // destructor

        void setFirstName( const string & ); // set first name
        string getFirstName() const; // return first name

        void setLastName( const string & ); // set last name
        string getLastName() const; // return last name

        void setSocialSecurityNumber( const string & ); // set SSN
        string getSocialSecurityNumber() const; // return SSN

        void setGrossSales( double ); // set gross sales amount
        double getGrossSales() const; // return gross sales amount

        void setCommissionRate( double ); // set commission rate
        double getCommissionRate() const; // return commission rate

};
#endif
```

© 2007 Pearson Ed - All rights reserved.
Constructors and Destructors in Derived Classes

```
30    double earnings() const; // calculate earnings
31    void print() const; // print CommissionEmployee object
32
33 private:
34    string firstName;
35    string lastName;
36    string socialSecurityNumber;
37    double grossSales; // gross weekly sales
38    double commissionRate; // commission percentage
39
40}; // end class CommissionEmployee
41
41 #endif
```
// Fig. 23.23: CommissionEmployee.cpp
// Class CommissionEmployee member-function definitions.
#include <iostream>
using std::cout;
using std::endl;
#include "CommissionEmployee.h" // CommissionEmployee class definition

// constructor
CommissionEmployee::CommissionEmployee(
    const string &first, const string &last, const string &ssn,
    double sales, double rate)
    : firstName( first ), lastName( last ), socialSecurityNumber( ssn )
{
    setGrossSales( sales ); // validate and store gross sales
    setCommissionRate( rate ); // validate and store commission rate
    cout << "CommissionEmployee constructor: " << endl;
    print();
    cout << "\n\n";
} // end CommissionEmployee constructor

// destructor
CommissionEmployee::~CommissionEmployee()
{
    cout << "CommissionEmployee destructor: " << endl;
    print();
    cout << "\n\n";
} // end CommissionEmployee destructor

Constructor and destructor output messages to demonstrate function call order
Constructors and Destructors
in Derived Classes

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

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

// calculate earnings
double CommissionEmployee::earnings() const
{
    return getCommissionRate() * getGrossSales();
} // end function earnings

// print CommissionEmployee object
void CommissionEmployee::print() const
{
    cout << "commission employee: "
    << getFirstName() << ' ' << getLastName()
    << "social security number: " << getSocialSecurityNumber()
    << "gross sales: " << getGrossSales()
    << "commission rate: " << getCommissionRate();
} // end function print
Constructors and Destructors in Derived Classes

```cpp
// Fig. 23.24: BasePlusCommissionEmployee.h
// BasePlusCommissionEmployee class derived from class CommissionEmployee.
#ifndef BASEPLUS_H
#define BASEPLUS_H

#include <string> // C++ standard string class
using std::string;

#include "CommissionEmployee.h" // CommissionEmployee class declaration

class BasePlusCommissionEmployee : public CommissionEmployee
{
public:
    BasePlusCommissionEmployee( const string &, const string &, const string &, double = 0.0, double = 0.0, double = 0.0 );
    ~BasePlusCommissionEmployee(); // destructor

    void setBaseSalary( double ); // set base salary
    double getBaseSalary() const; // return base salary

    double earnings() const; // calculate earnings
    void print() const; // print BasePlusCommissionEmployee object

private:
    double baseSalary; // base salary
}; // end class BasePlusCommissionEmployee
#endif
```

© 2007 Pearson Ed - All rights reserved.
// Fig. 23.25: BasePlusCommissionEmployee.cpp
// Class BasePlusCommissionEmployee member-function definitions.
#include <iostream>
using std::cout;
using std::endl;

// BasePlusCommissionEmployee class definition
#include "BasePlusCommissionEmployee.h"

destructor
BasePlusCommissionEmployee::~BasePlusCommissionEmployee()
{
    cout << "BasePlusCommissionEmployee destructor: " << endl;
    print();
    cout << "\n\n";
} // end BasePlusCommissionEmployee destructor

Constructor and destructor output messages to demonstrate function call order
Constructors and Destructors in Derived Classes

```cpp
// set base salary
void BasePlusCommissionEmployee::setBaseSalary(double salary)
{
    baseSalary = (salary < 0.0) ? 0.0 : salary;
} // end function setBaseSalary

// return base salary
double BasePlusCommissionEmployee::getBaseSalary() const
{
    return baseSalary;
} // end function getBaseSalary

// calculate earnings
double BasePlusCommissionEmployee::earnings() const
{
    return getBaseSalary() + CommissionEmployee::earnings();
} // end function earnings

// print BasePlusCommissionEmployee object
void BasePlusCommissionEmployee::print() const
{
    cout << "base-salaried ";
    // invoke CommissionEmployee's print function
    CommissionEmployee::print();
    cout << "\nbase salary: " << getBaseSalary();
} // end function print
```

Systems Programming: Inheritance
Constructors and Destructors in Derived Classes

1 // Fig. 23.26: fig23_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"
```cpp
int main()
{
    // set floating-point output formatting
    cout << fixed << setprecision( 2 );

    { // begin new scope
        CommissionEmployee employee1( "Bob", "Lewis", "333-33-3333", 5000, .04 );
    } // end scope

    cout << endl;

    BasePlusCommissionEmployee employee2( "Lisa", "Jones", "555-55-5555", 2000, .06, 800 );

    cout << endl;

    BasePlusCommissionEmployee employee3( "Mark", "Sands", "888-88-8888", 8000, .15, 2000 );

    cout << endl;
    return 0;
} // end main
```

**CommissionEmployee object goes in and out of scope immediately**

**Instantiate two BasePlusCommissionEmployee objects to demonstrate order of derived-class and base-class constructor/destructor function calls**
Constructors and Destructors in Derived Classes

CommissionEmployee constructor:
- commission employee: Bob Lewis
- social security number: 333-33-3333
- 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:
- 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: 555-55-5555
- 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

© 2007 Pearson Ed - All rights reserved.
Constructors and Destructors in Derived Classes

(... continued from bottom of previous slide)

BasePlusCommissionEmployee constructor:
base-salaried commission employee: Mark Sands
social security number: 888-88-8888
gross sales: 8000.00
commission rate: 0.15
base salary: 2000.00

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

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

Destructors for BasePlusCommissionEmployee object called in reverse order of constructors

CommissionEmployee destructor:
commission employee: Mark Sands
social security number: 888-88-8888
gross sales: 8000.00
commission rate: 0.15

CommissionEmployee destructor:
commission employee: Lisa Jones
social security number: 555-55-5555
gross sales: 2000.00
commission rate: 0.06
base salary: 800.00

Destructors for BasePlusCommissionEmployee object called in reverse order of constructors

© 2007 Pearson Ed - All rights reserved.
Summary

- Base Classes and Derived Classes
- Five Examples of Base Class and Derived Class Relationships
  - Focused on the distinctions in using public, private and protected data members and public get/set member functions
- Order of execution of constructors and destructors in inheritance hierarchy chains.