C++

Inheritance
Inheritance

- Introduction
- Base Classes and Derived Classes
- Five Examples of Base Class and Derived Class Relationships
- Constructors and Destructors in Derived Classes
Introduction

- **Inheritance** is a form of software reuse where a class is created that absorbs an existing class’s data and behaviors, then customizes or enhances them with new capabilities.
- The new class, the **derived class**, inherits the members of the existing class, the **base class**.
- The derived class represents a more specialized group of objects.
Introduction

- 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., *car* (derived class) is a *vehicle* (base class).

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

- Base class typically represents a 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 (i.e. a class hierarchy).
Fig. 20.2 Inheritance Hierarchy for University CommunityMember

Arrows represent an is-a relationship
**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** from its derived classes, 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.
1. Create and use 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.
3. Create a **CommissionEmployee-BasePlusCommissionEmployee** inheritance hierarchy with private members.

4. Create a **CommissionEmployee-BasePlusCommissionEmployee** inheritance hierarchy with protected members.

5. Create a **CommissionEmployee-BasePlusCommissionEmployee** 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.
Example 1: CommissionEmployee Class

```cpp
// 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
```

Class CommissionEmployee constructor
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

#endif
// Fig. 23.5: 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
}

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

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

Initialize data members
// 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

Function setGrossSales validates grossSales amount

© 2007 Pearson Ed -All rights reserved.
// return gross sales amount
double CommissionEmployee::getGrossSales() const
{
    return grossSales;
}

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

// return commission rate
double CommissionEmployee::getCommissionRate() const
{
    return commissionRate;
}

Function setCommissionRate validates commissionRate
// calculate earnings

double CommissionEmployee::earnings() const
{
    return commissionRate * grossSales;
}

// end function earnings

// print CommissionEmployee object

void CommissionEmployee::print() const
{
    cout << "commission employee: " << firstName << ' ' << lastName
         << "\nsocial security number: " << socialSecurityNumber
         << "\ngross sales: " << grossSales
         << "\ncommission rate: " << commissionRate;
}

// end function print
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.
**Example 1: CommissionEmployee Class**

```cpp
employee.setGrossSales(8000); // set gross sales
employee.setCommissionRate(.1); // set commission rate

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

// display the employee's earnings
cout << "\n\nEmployee'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.

© 2007 Pearson Ed - All rights reserved.
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**
// 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
void setBaseSalary(double); // set base salary
double getBaseSalary() const; // return base salary

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

private:
  string firstName;
  string lastName;
  string socialSecurityNumber;
  double grossSales; // gross weekly sales
  double commissionRate; // commission percentage
  double baseSalary; // base salary
}; // end class BasePlusCommissionEmployee

#if defined(Handwritten)
  void setBaseSalary(double);
  double getBaseSalary() const;
  double earnings() const;
  void print() const;
#endif
```

Define get and set functions for data member baseSalary

Add data member baseSalary
Example 2: BasePlusCommissionEmployee Class

```cpp
// 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
// 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

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

```c++
// 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 << ' ' << lastName << "\nsocial security number: " << socialSecurityNumber << "\ngross sales: " << grossSales << "\ncommission rate: " << commissionRate << "\nbase 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`.

© 2007 Pearson Ed - All rights reserved.
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()
{
    BasePlusCommissionEmployee employee( "Bob", "Lewis", "333-33-3333", 5000, .04, 300 );

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

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

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

employee.print(); // display the new employee information

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

return 0;  // end main
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
- 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**.
Example 3: a CommissionEmployee-BasePlusCommissionEmployee Inheritance Hierarchy

1 // Fig. 23.10: 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 &, const string &,
16       const string &, double = 0.0, double = 0.0, double = 0.0 );
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:
24     double baseSalary; // base salary
25   } // end class BasePlusCommissionEmployee
26
27 #endif

© 2007 Pearson Ed - All rights reserved.
// 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 )

// 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

Inheritance Hierarchy
// calculate earnings
double BasePlusCommissionEmployee::earnings() const
{
    // derived class cannot access the base class's private data
    return baseSalary + (commissionRate * grossSales);
} // end function earnings

// print BasePlusCommissionEmployee object
void BasePlusCommissionEmployee::print() const
{
    // derived class cannot access the base class's private data
    cout << "base-salaried commission employee: " << firstName << ' ' << lastName << "\nsocial security number: " << socialSecurityNumber << "\ngross sales: " << grossSales << "\ncommission rate: " << commissionRate << "\nbase salary: " << baseSalary;
} // end function print
Example 3:

**a CommissionEmployee - BasePlusCommissionEmployee**

**Inheritance Hierarchy**

```plaintext
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'
```

© 2007 Pearson Ed - All rights reserved.
Example 3:
a CommissionEmployee - BasePlusCommissionEmployee
Inheritance Hierarchy

C:\examples\ch23\Fig23_10_11\BasePlusCommission-Employee.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\BasePlusCommission-Employee.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\ch3\Fig23_10_11\BasePlusCommission-Employee.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\BasePlusCommission-Employee.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'
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.
- 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.
// 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

};

© 2007 Pearson Ed - All rights reserved.
Example 4:  
a CommissionEmployee-BasePlusCommissionEmployee  
Inheritance Hierarchy using Protected Data

```cpp
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

#if defined __cplusplus
#endif
```
// 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
// 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
// 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

// calculate earnings
double CommissionEmployee::earnings() const
{
    return commissionRate * grossSales;
} // end function earnings
// print CommissionEmployee object
void CommissionEmployee::print() const
{
    cout << "commission employee: " << firstName << ' ' << lastName
    << "\nsocial security number: " << socialSecurityNumber
    << "\ngross sales: " << grossSales
    << "\ncommission rate: " << commissionRate;
} // end function print
Example 4: 

a CommissionEmployee-BasePlusCommissionEmployee 

Inheritance Hierarchy using Protected Data

```cpp
1 // Fig. 23.14: 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 &, const string &, const string &, double = 0.0, double = 0.0, double = 0.0 ); // set base salary
16    void setBaseSalary( double ); // set base salary
17    double getBaseSalary() const; // return base salary
18 
19    double earnings() const; // calculate earnings
20    void print() const; // print BasePlusCommissionEmployee object
21 
22 private:
23    double baseSalary; // base salary
24 }; // end class BasePlusCommissionEmployee
25 #endif
```

BasePlusCommissionEmployee still inherits publicly from CommissionEmployee

© 2007 Pearson Ed - All rights reserved.
// Fig. 23.15: 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 )
    // 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 4: 
a CommissionEmployee-BasedPlusCommissionEmployee
Inheritance Hierarchy using Protected Data

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

// 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;
}
```

Directly access base class’s protected data
Example 4: a CommissionEmployee - BasePlusCommissionEmployee

Inheritance Hierarchy using Protected Data

```cpp
// Fig. 23.16: fig23_16.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);
}
```

© 2007 Pearson Ed - All rights reserved.
// get commission employee data
cout << "Employee information obtained by get functions: \n" << endl;
    << "\nFirst name is " << employee.getFirstName() << endl;
    << "\nLast name is " << employee.getLastName() << endl;
    << "\nSocial security number is " << employee.getSocialSecurityNumber() << endl;
    << "\nGross sales is " << employee.getGrossSales() << endl;
    << "\nCommission rate is " << employee.getCommissionRate() << endl;
    << "\nBase salary is " << employee.getBaseSalary() << endl;

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

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

// display the employee's earnings
cout << "\n\nEmployee's earnings: $" << employee.earnings() << endl;
return 0;
} // end main
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.
- 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.
// Fig. 23.17: CommissionEmployee.h
// CommissionEmployee class definition with good software engineering.
#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
    double getCommissionRate() const; // return commission rate
};
Example 5:  
a CommissionEmployee-BasePlusCommissionEmployee  
Inheritance Hierarchy using Private Data

```
29 double earnings() const; // calculate earnings
30 void print() const; // print CommissionEmployee object
31
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
Example 5:  
a CommissionEmployee-BasePlusCommissionEmployee  
Inheritance Hierarchy using Private Data

```
// Fig. 23.18: 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), lastName(last), socialSecurityNumber(ssn)  
{  
    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
```

Use member initializers to set the values of members firstName, lastName and socialSecurityNumber

© 2007 Pearson Ed - All rights reserved.
// 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
// 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

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

Use get functions to obtain the values of data members

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

a CommissionEmployee - BasePlusCommissionEmployee

Inheritance Hierarchy using Private Data

```c++
84 // print CommissionEmployee object
85 void CommissionEmployee::print() const
86 {
87    cout << "commission employee: 
88       " << getFirstName() << ' ' << getLastName()
89       << "\nsocial security number: " << getSocialSecurityNumber()
90       << "\ngross sales: " << getGrossSales()
91       << "\ncommission rate: " << getCommissionRate();
92 } // end function print
```

Use get functions to obtain the values of data members
// 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
// Fig. 23.20: 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 )
  // 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:

a CommissionEmployee - BasePlusCommissionEmployee

Inheritance Hierarchy using Private Data

```cpp
// 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
```

Invoke base class’s `earnings` function

Invoke base class’s `print` function
// Fig. 23.21: fig23_21.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"

Example 5:
a CommissionEmployee-BasePlusCommissionEmployee
Inheritance Hierarchy using Private Data
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"
        << "First name is " << employee.getFirstName()
        << "Last name is " << employee.getLastName()
        << "Social security number is "
        << employee.getSocialSecurityNumber()
        << "Gross sales is " << employee.getGrossSales()
        << "Commission rate is " << employee.getCommissionRate()
        << "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
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
20.5 Constructors and Destructors in Derived Classes

- Instantiating derived-class object
  - Chain of constructor calls
    - Derived-class constructor invokes its base class constructor either explicitly (via a base-class member initializer) or implicitly (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.
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.
CommissionEmployee.h

CommissionEmployee class definition represents a commission employee.

#ifndef COMMISSION_H
#define COMMISSION_H

#include <string>

using std::string;

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

};
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

#endif
Constructors and Destructors in Derived Classes

```
// Fig. 23.23: CommissionEmployee.cpp
#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.
// 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

// 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

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

1 // Fig. 23.24: BasePlusCommissionEmployee.h
2 // BasePlusCommissionEmployee class derived from class
3 // CommissionEmployee.
4 #ifndef BASEPLUS_H
5 #define BASEPLUS_H

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

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

11 class BasePlusCommissionEmployee : public CommissionEmployee
12 {
14 public:
15     BasePlusCommissionEmployee( const string &, const string &,
16                                    const string &, double = 0.0, double = 0.0, double = 0.0 );
17         ~BasePlusCommissionEmployee(); // destructor
18
19     void setBaseSalary( double ); // set base salary
20     double getBaseSalary() const; // return base salary
21
22     double earnings() const; // calculate earnings
23     void print() const; // print BasePlusCommissionEmployee object
24 private:
25         double baseSalary; // base salary
26 }; // end class BasePlusCommissionEmployee

27 #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"

// constructor
BasePlusCommissionEmployee::BasePlusCommissionEmployee(
    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

    cout << "BasePlusCommissionEmployee constructor: " << endl;
    print();
    cout << "\n";
} // end BasePlusCommissionEmployee constructor

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

Constructor and destructor output messages to demonstrate function call order.
// set base salary
void BasePlusCommissionEmployee::setBaseSalary(double salary)
{
    baseSalary = (salary < 0.0) ? 0.0 : salary;
}

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

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

// print BasePlusCommissionEmployee object
void BasePlusCommissionEmployee::print() const
{
    cout << "base-salaried ";
    CommissionEmployee::print();
    cout << "\nbase salary: " << getBaseSalary();
}
// Fig. 23.26: fig23_26.cpp
// Display order in which base-class and derived-class constructors
// and destructors are called.
#include <iostream>
using std::cout;
using std::endl;
using std::fixed;

#include <iomanip>
using std::setprecision;

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

© 2007 Pearson Ed -All rights reserved.
int main()
{
    // set floating-point output formatting
    cout << fixed << setprecision(2);

    CommissionEmployee employee1("Bob", "Lewis", "333-33-3333", 5000, .04);

    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
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

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

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.
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

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: 555-55-5555
gross sales: 2000.00
commission rate: 0.06

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.
Review of Inheritance

- **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
  - Elements that are not inherited by derived classes.

- **Order of execution of constructors and destructors in inheritance hierarchy chains.**