

10

Object-Oriented Programming: Inheritance

© 2006 Pearson Education, Inc. All rights reserved.

- ### OBJECTIVES
- In this chapter you will learn:
- How inheritance promotes software reusability.
 - The concepts of base classes and derived classes.
 - To create a derived class that inherits attributes and behaviors from a base class.
 - To use access modifier protected to give derived class methods access to base class members.
 - To access base class members with base.
 - How constructors are used in inheritance hierarchies.
 - The methods of class object, the direct or indirect base class of all classes.
- © 2006 Pearson Education, Inc. All rights reserved.

- ### 10.1 Introduction
- Inheritance**
 - Software reusability
 - Create new class from existing class
 - Absorb existing class's data and behaviors
 - Enhance with new capabilities
 - Save time in application development by reusing proven and debugged code
 - Base class = an already existing class
 - Derived class = extends base class, a new class that inherits the members of an existing class
 - More specialized group of objects
 - Behaviors inherited from base class and it can be customize with additional behaviors (methods or fields)
 - Each derived class can become the base case for future derived classes
- © 2006 Pearson Education, Inc. All rights reserved.

- ### Is-a and Has-a Relationships
- Has-a relationship**
 - Represents composition
 - Composition = a class can have references to objects of other classes as members
 - An object contains as members references to other objects
 - Chapter 10, Section 8 for more details
 - Is-a relationship**
 - Represents inheritance
 - An object of a derived class can also be treated as an object of its base class
- © 2006 Pearson Education, Inc. All rights reserved.

- ### 10.2 Base Classes and Derived Classes
- Base classes and derived classes**
 - Object of one class "is an" object of another class
 - Example: Rectangle is a quadrilateral.
 - Class **Rectangle** inherits from class **Quadrilateral**
 - Quadrilateral**: base class
 - Rectangle**: derived class
 - Base class typically represents larger set of objects than derived classes
 - Example:
 - Base class: **Vehicle**
 - Cars, trucks, boats, bicycles, ...
 - Derived class: **Car**
 - Smaller, more-specific subset of vehicles
- © 2006 Pearson Education, Inc. All rights reserved.

Base class	Derived classes
Student	GraduateStudent, UndergraduateStudent
Shape	Circle, Triangle, Rectangle
Loan	CarLoan, HomeImprovementLoan, MortgageLoan
Employee	Faculty, Staff, HourlyWorker, CommissionWorker
BankAccount	CheckingAccount, SavingsAccount

Fig. 10.1 | Inheritance examples.

© 2006 Pearson Education, Inc. All rights reserved.

10.2 Base Classes and Derived Classes (Cont.)

• Inheritance hierarchy

- Inheritance relationships: tree-like hierarchy structure
- Each class becomes
 - Base class - exists in a hierarchical relationship with its derived classes
 - Supply members to other classes
- OR
- Derived class
 - Inherit members from other classes

© 2006 Pearson Education, Inc. All rights reserved.

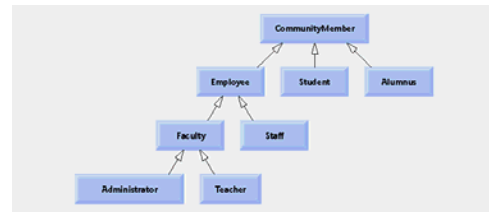


Fig. 10.2 | UML class diagram showing an inheritance hierarchy for university CommunityMembers.

© 2006 Pearson Education, Inc. All rights reserved.

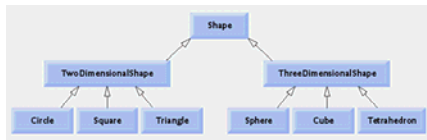


Fig. 10.3 | UML class diagram showing an inheritance hierarchy for Shapes.

© 2006 Pearson Education, Inc. All rights reserved.

Access Modifiers

- Access Modifiers = Control access to a class's variables and methods
- There are several types previously covered:
 - Public
 - Accessible wherever the application has a reference to an object of that class or one of its derived classes
 - Private
 - Accessible only within the class itself
 - A base class's private members are inherited by its derived classes, but are not directly accessible by derived class methods and properties

© 2006 Pearson Education, Inc. All rights reserved.

10.3 protected Members

• protected access

- Intermediate level of protection between public and private
- protected members accessible by
 - Base class members
 - Derived class members
- Derived class access to base class member
 - Keyword **base** and a dot (.) operator

© 2006 Pearson Education, Inc. All rights reserved.

Other Thoughts

- Declaring instance variables with the access modifier private is known as information hiding
- All non-private base class members retain their original access modifier when they become members of the derived class (private members are not inherited!)
- Derived class methods can refer to public and protected members inherited from the base class simply by using the member names
- When a derived class method overrides a base class method, the base class version of the method can be accessed from the derived class by preceding the base class method name with the keyword base and the dot (.) operator

© 2006 Pearson Education, Inc. All rights reserved.

10.4 Relationship between Base Classes and Derived Classes

- We will use an inheritance hierarchy containing types of employees in a payroll application to discuss the relationship between Base Classes and Derived Classes
- Commission Employees – base class
 - Paid a % of their sales
- Base-salaried Commission Employees = derived class
 - Paid a base salary + percentage of sales

© 2006 Pearson Education, Inc. All rights reserved.

10.4 Relationship between Base Classes and Derived Classes

Base class and derived class relationship

– Example:

CommissionEmployee/BasePlusCommissionEmployee inheritance hierarchy

- CommissionEmployee
 - First name, last name, SSN, commission rate, gross sale amount
- BasePlusCommissionEmployee
 - First name, last name, SSN, commission rate, gross sale amount
 - Base salary

© 2006 Pearson Education, Inc. All rights reserved.

10.4.1 Creating and Using a CommissionEmployee Class

- Class CommissionEmployee
 - Extends class object
 - Colon ":"
 - Every class in C# extends an existing class
 - Except object
 - Every class inherits object's methods
 - New class implicitly extends object
 - To override base class method use keyword **override** with the same signature (Overridden method must be declared **virtual**)
 - Constructors are not inherited
 - The first task of any derived class constructor is to call its direct base class's constructor

© 2006 Pearson Education, Inc. All rights reserved.

```

1 // Fig. 10.4: CommissionEmployee.cs
2 // CommissionEmployee class represents a commission employee.
3 public class CommissionEmployee : object
4 {
5     private string firstName;
6     private string lastName;
7     private string socialSecurityNumber;
8     private decimal grossSales; // gross weekly sales
9     private decimal commissionRate; // commission percentage
10
11 // five-parameter constructor
12 public CommissionEmployee( string first, string last, string ssn,
13     decimal sales, decimal rate )
14 {
15     // implicit call to object constructor occurs here
16     firstName = first;
17     lastName = last;
18     socialSecurityNumber = ssn;
19     GrossSales = sales; // validate gross sales via property
20     CommissionRate = rate; // validate commission rate via property
21 } // end five-parameter CommissionEmployee constructor
22
23 // read-only property that gets commission employee's first name
24 public string FirstName
25 {
26     get
27     {
28         return firstName;
29     } // end get
30 } // end property FirstName
    
```

Class CommissionEmployee extends class object

Declare private instance variables

Implicit call to object constructor

Initialize instance variables

Invoke properties GrossSales and CommissionRate to validate data

© 2006 Pearson Education, Inc. All rights reserved.

```

31
32 // read-only property that gets commission employee's last name
33 public string LastName
34 {
35     get
36     {
37         return lastName;
38     } // end get
39 } // end property LastName
40
41 // read-only property that gets
42 // commission employee's social security number
43 public string SocialSecurityNumber
44 {
45     get
46     {
47         return socialSecurityNumber;
48     } // end get
49 } // end property SocialSecurityNumber
    
```

Outline

CommissionEmployee .CS

(2 of 4)

© 2006 Pearson Education, Inc. All rights reserved.

```

50
51 // property that gets and sets commission employee's gross sales
52 public decimal GrossSales
53 {
54     get
55     {
56         return grossSales;
57     } // end get
58     set
59     {
60         grossSales = ( value < 0 ) ? 0 : value;
61     } // end set
62 } // end property GrossSales
63
64 // property that gets and sets commission employee's commission rate
65 public decimal CommissionRate
66 {
67     get
68     {
69         return commissionRate;
70     } // end get
71     set
72     {
73         commissionRate = ( value > 0 && value < 1 ) ? value : 0;
74     } // end set
75 } // end property CommissionRate
    
```

Outline

CommissionEmployee .CS

(3 of 4)

© 2006 Pearson Education, Inc. All rights reserved.

```

76 // calculate commission on employee's pay
77 public decimal Earnings()
78 {
79     return commissionRate * grossSales;
80 } // end method Earnings
81
82 // return string representation of CommissionEmployee object
83 public override string ToString()
84 {
85     return string.Format(
86         "{0:1} {2}{3:12} {4:C}{5:12} {6:F2}",
87         "commission on employee", FirstName, LastName,
88         "social security number", SocialSecurityNumber,
89         "gross sales", GrossSales, "commission rate", CommissionRate );
90 } // end method ToString
91 } // end class CommissionEmployee
92

```

Calculate earnings

CommissionEmployee.cs (4 of 4)

Override method ToString of class object

Outline

© 2006 Pearson Education, Inc. All rights reserved.

```

1 // Fig. 10.8: CommissionEmployeeTest.cs
2 // Testing class CommissionEmployee.
3 using System;
4
5 public class CommissionEmployeeTest
6 {
7     // instantiate CommissionEmployee object
8     public static void Main( string[] args )
9     {
10        // instantiate CommissionEmployee object
11        CommissionEmployee employee = new CommissionEmployee( "Sue",
12            "Jones", "222-22-2222", 10000.00M, 0.06M );
13
14        // display commission on employee data
15        Console.WriteLine(
16            "Employee information obtained by properties and methods: \n" );
17        Console.WriteLine( "{0:1}", "First name is",
18            employee.FirstName );
19        Console.WriteLine( "{0:1}", "Last name is",
20            employee.LastName );
21        Console.WriteLine( "{0:13}", "Social security number is",
22            employee.SocialSecurityNumber );
23        Console.WriteLine( "{0:12} ", "Gross sales are",
24            employee.GrossSales );
25        Console.WriteLine( "{0:1:F2}", "Commission rate is",
26            employee.CommissionRate );
27        Console.WriteLine( "{0:12} ", "Earnings are",
28            employee.Earnings );
29    }
30 }

```

Instantiate CommissionEmployee object

CommissionEmployeeTest.cs (1 of 2)

Use CommissionEmployee's properties to retrieve and change the object's instance variable values

Outline

© 2006 Pearson Education, Inc. All rights reserved.

```

28 employee.GrossSales = 10000.00M; // set gross sales
29 employee.CommissionRate = 0.06M; // set commission rate
30
31 Console.WriteLine( "\n\n(1)" );
32 "Updated employee information obtained by ToString(): employee: ";
33 Console.WriteLine( "earnings: {0:C}", employee.Earnings );
34 } // end Main
35 } // end class CommissionEmployeeTest
36

```

Employee information obtained by properties and methods:

```

First name is Sue
Last name is Jones
Social security number is 222-22-2222
Gross sales are $10,000.00
Commission rate is 0.06
Earnings are $600.00

```

Updated employee information obtained by ToString():

```

commission on employee: Sue Jones
social security number: 222-22-2222
gross sales: $5,000.00
commission rate: 0.10
earnings: $500.00

```

Implicitly call the object's ToString method

CommissionEmployeeTest.cs (2 of 2)

Outline

© 2006 Pearson Education, Inc. All rights reserved.

10.4.2 Creating a BasePlusCommissionEmployee Class without Using Inheritance

- Class BasePlusCommissionEmployee
 - Much of the code is similar to CommissionEmployee
 - private instance variables
 - public methods
 - Constructor
 - Properties
 - Additions
 - private instance variable baseSalary
 - BaseSalary property

Outline

© 2006 Pearson Education, Inc. All rights reserved.

```

1 // Fig. 10.9: BasePlusCommissionEmployee.cs
2 // BasePlusCommissionEmployee class represents an employee that receives
3 // a base salary in addition to a commission.
4 public class BasePlusCommissionEmployee
5 {
6     private string firstName;
7     private string lastName;
8     private string socialSecurityNumber;
9     private decimal grossSales; // gross weekly sales
10    private decimal commissionRate; // commission on percentage
11    private decimal baseSalary; // base salary per week
12
13    // six-parameter constructor
14    public BasePlusCommissionEmployee( string first, string last,
15        string ssn, decimal sales, decimal rate, decimal salary )
16    {
17        // implicit call to object constructor occurs here
18        firstName = first;
19        lastName = last;
20        socialSecurityNumber = ssn;
21        GrossSales = sales; // validate gross sales via property
22        CommissionRate = rate; // validate commission rate via property
23        BaseSalary = salary; // validate base salary via property
24    } // end six-parameter BasePlusCommissionEmployee constructor
25

```

Add instance variable baseSalary

BasePlusCommissionEmployee.cs (1 of 5)

Use property BaseSalary to validate data

Outline

© 2006 Pearson Education, Inc. All rights reserved.

```

25 // read-only property that gets
26 // base-salaried commission on employee's first name
27 public string FirstName
28 {
29     get
30     {
31         return firstName;
32     } // end get
33 } // end property FirstName
34
35 // read-only property that gets
36 // base-salaried commission on employee's last name
37 public string LastName
38 {
39     get
40     {
41         return lastName;
42     } // end get
43 } // end property LastName
44
45 // read-only property that gets
46 // base-salaried commission on employee's social security number
47 public string SocialSecurityNumber
48 {
49     get
50     {
51         return socialSecurityNumber;
52     } // end get
53 } // end property SocialSecurityNumber
54

```

BasePlusCommissionEmployee.cs (2 of 5)

Outline

© 2006 Pearson Education, Inc. All rights reserved.

```

56 // property that gets and sets
57 // base-salaried commission employee's gross sales
58 public decimal GrossSales
59 {
60     get
61     {
62         return grossSales;
63     } // end get
64     set
65     {
66         grossSales = ( value < 0 ) ? 0 : value;
67     } // end set
68 } // end property GrossSales
69
70 // property that gets and sets
71 // base-salaried commission employee's commission rate
72 public decimal CommissionRate
73 {
74     get
75     {
76         return commissionRate;
77     } // end get
78     set
79     {
80         commissionRate = ( value > 0 && value < 1 ) ? value : 0;
81     } // end set
82 } // end property CommissionRate

```

Outline

BasePI usCommi ssi onEmpl oye e. cs
(3 of 5)

© 2006 Pearson Education, Inc. All rights reserved.

```

84 // property that gets and sets
85 // base-salaried commission employee's base salary
86 public decimal BaseSalary
87 {
88     get
89     {
90         return baseSalary;
91     } // end get
92     set
93     {
94         baseSalary = ( value < 0 ) ? 0 : value;
95     } // end set
96 } // end property BaseSalary
97
98 // calculate earnings
99 public decimal Earnings()
100 {
101     return BaseSalary * ( CommissionRate * GrossSales );
102 } // end method earnings

```

Outline

BasePI usCommi ssi onEmpl oye e. cs
(4 of 5)

Validates data and sets instance variable baseSalary

Update method Earnings to calculate the earnings of a base-salaried commission employee

© 2006 Pearson Education, Inc. All rights reserved.

```

104 // return string representation of BasePI usCommi ssi onEmpl oye e
105 public override string ToString()
106 {
107     return string.Format(
108         "{0: (1) {2}\n(3): {4}\n(5): {6:C}\n(7): {8:F2}\n(9): {10:C}",
109         "base-salaried commission employee", FirstName, LastName,
110         "social security number", SocialSecurityNumber,
111         "gross sales", GrossSales, "commission rate", CommissionRate,
112         "base salary", BaseSalary );
113 } // end method ToString
114 } // end class BasePI usCommi ssi onEmpl oye e

```

Outline

BasePI usCommi ssi onEmpl oye e. cs
(5 of 5)

Update method ToString to display base salary

© 2006 Pearson Education, Inc. All rights reserved.

```

1 // Fig. 10.7: BasePI usCommi ssi onEmpl oye eTest.cs
2 // Testing class BasePI usCommi ssi onEmpl oye e.
3 using System;
4
5 public class BasePI usCommi ssi onEmpl oye eTest
6 {
7     public static void Main( string[] args )
8     {
9         // instantiate BasePI usCommi ssi onEmpl oye e object
10        BasePI usCommi ssi onEmpl oye e empl oye =
11            new BasePI usCommi ssi onEmpl oye e( "Bob", "Lewis",
12            "333-33-3333", 5000.00, 0.04, 300.00 );
13
14        // display base-salaried commission employee data
15        Console.WriteLine(
16            "Employee information obtained by properties and methods: \n" );
17        Console.WriteLine( "{0: (1)", "First name is",
18            empl oye. FirstName );
19        Console.WriteLine( "{0: (1)", "Last name is",
20            empl oye. LastName );
21        Console.WriteLine( "{0: (1)", "Social security number is",
22            empl oye. SocialSecurityNumber );
23        Console.WriteLine( "{0: (1:C)", "Gross sales are",
24            empl oye. GrossSales );
25        Console.WriteLine( "{0: (1:F2)", "Commission rate is",
26            empl oye. CommissionRate );
27        Console.WriteLine( "{0: (1:C)", "Earnings are",
28            empl oye. Earnings() );
29        Console.WriteLine( "{0: (1:C)", "Base salary is",
30            empl oye. BaseSalary );

```

Outline

BasePI usCommi ssi onEmpl oye eTest. cs
(1 of 2)

Instantiate BasePI usCommi ssi onEmpl oye e object

Use BasePI usCommi ssi onEmpl oye e's properties to retrieve and change the object's instance variable values

© 2006 Pearson Education, Inc. All rights reserved.

```

31
32     empl oye. BaseSalary = 1000.00; // set base salary
33
34     Console.WriteLine( "\n(0): \n\n(1)",
35         "Updated employee information obtained by ToString", empl oye );
36     Console.WriteLine( "earnings: {0:C}", empl oye. Earnings() );
37 } // end Main
38 } // end class BasePI usCommi ssi onEmpl oye eTest

```

Outline

BasePI usCommi ssi onEmpl oye eTest. cs
(2 of 2)

Implicitly call the object's ToString method

Employee information obtained by properties and methods:
 First name is Bob
 Last name is Lewis
 Social security number is 333-33-3333
 Gross sales are \$5,000.00
 Commission rate is 0.04
 Earnings are \$500.00
 Base salary is \$300.00

Updated employee information obtained by ToString:
 base-salaried commission employee: Bob Lewis
 social security number: 333-33-3333
 gross sales: \$5,000.00
 commission rate: 0.04
 base salary: \$1,000.00
 earnings: \$1,200.00

© 2006 Pearson Education, Inc. All rights reserved.

10.4.3 Creating a CommissionEmployee-BasePI usCommi ssi onEmpl oye e Inheritance Hierarchy

- Class BasePI usCommi ssi onEmpl oye e2
 - Extends class CommissionEmployee
 - Is a CommissionEmployee
 - Has instance variable baseSalary
 - Inherits public and protected members
 - Constructor not inherited

© 2006 Pearson Education, Inc. All rights reserved.

```

1 // Fig. 10.8: BasePlusCommi ssi onEmpl oye2.cs
2 // BasePlusCommi ssi onEmpl oye2 inherits from class Commi ssi onEmpl oye.
3 public class BasePlusCommi ssi onEmpl oye2 : Commi ssi onEmpl oye
4 {
5     private decimal baseSalary; // base salary per week
6
7     // six-parameter derived class constructor
8     // with call to base class Commi ssi onEmpl oye constructor
9     public BasePlusCommi ssi onEmpl oye2( string first, string last,
10        string ssn, decimal sales, decimal rate, decimal salary )
11     {
12         base( first, last, ssn, sales, rate );
13     }
14
15     // BaseSalary = salary; // validate base salary via property
16     // end six-parameter BasePlusCommi ssi onEmpl oye2 constructor
17
18     // property that gets and sets
19     // base-salaried commi ssi on empl oye's base salary
20     public decimal BaseSalary
21     {
22         get
23         {
24             return baseSalary;
25         } // end get
26         set
27         {
28             baseSalary = ( value < 0 ) ? 0 : value;
29         } // end set
30     } // end property BaseSalary
31 }

```

Outline

Class BasePlusCommi ssi onEmpl oye2 is a derived class of Commi ssi onEmpl oye. BasePlusCommi ssi onEmpl oye2.cs (1 of 2)

Invoke the base class constructor using the base class constructor call syntax

© 2006 Pearson Education, Inc. All rights reserved.

```

32 // calculate earnings
33 public override decimal Earnings()
34 {
35     // not allowed: commi ssi onRate and grossSales private in base class
36     return baseSalary + ( commi ssi onRate * grossSales );
37 } // end method Earnings
38
39 // return string representation of BasePlusCommi ssi onEmpl oye2
40 public override string ToString()
41 {
42     // not allowed: attempts to access private base class members
43     return string.Format(
44         "(0): {1} (2){n(3): (4){n(5): (6:C){n(7): (8:F2){n(9): (10:C)",
45         "base-salaried commi ssi on empl oye", first, last, ssn,
46         "social security number", socialSecurityNumber,
47         "gross sales", grossSales, "commi ssi on rate", commi ssi onRate,
48         "base salary", baseSalary );
49 } // end method ToString
50 } // end class BasePlusCommi ssi onEmpl oye2

```

Outline

BasePlusCommi ssi onEmpl oye2.cs

Compiler generates errors because base class's instance variable are private

© 2006 Pearson Education, Inc. All rights reserved.

Fig. 10.9 | Compilation errors generated by BasePlusCommi ssi onEmpl oye2 (Fig. 10.8) after declaring the Earnings method in Fig. 10.4 with keyword virtual.

© 2006 Pearson Education, Inc. All rights reserved.

10.4.4 Commi ssi onEmpl oye-BasePlusCommi ssi onEmpl oye Inheritance Hierarchy Using protected Instance Variables

- Use **protected** instance variables
 - Enable class BasePlusCommi ssi onEmpl oye to directly access base class instance variables
 - Base class's **protected** members are inherited by all derived classes of that base class

© 2006 Pearson Education, Inc. All rights reserved.

```

1 // Fig. 10.10: Commi ssi onEmpl oye2.cs
2 // Commi ssi onEmpl oye2 with protected instance variables.
3 public class Commi ssi onEmpl oye2
4 {
5     protected string firstName;
6     protected string lastName;
7     protected string socialSecurityNumber;
8     protected decimal grossSales; // gross weekly sales
9     protected decimal commi ssi onRate; // commi ssi on percentage
10
11     // five-parameter constructor
12     public Commi ssi onEmpl oye2( string first, string last, string ssn,
13        decimal sales, decimal rate )
14     {
15         // implicit call to object constructor occurs here
16         firstName = first;
17         lastName = last;
18         socialSecurityNumber = ssn;
19         grossSales = sales; // validate gross sales via property
20         commi ssi onRate = rate; // validate commi ssi on rate via property
21     } // end five-parameter Commi ssi onEmpl oye2 constructor
22
23     // read-only property that gets commi ssi on empl oye's first name
24     public string FirstName
25     {
26         get
27         {
28             return firstName;
29         } // end get
30     } // end property FirstName
31 }

```

Outline

Commi ssi onEmpl oye2.cs (1 of 4)

Declare protected instance variables

© 2006 Pearson Education, Inc. All rights reserved.

```

32 // read-only property that gets commi ssi on empl oye's last name
33 public string LastName
34 {
35     get
36     {
37         return lastName;
38     } // end get
39 } // end property LastName
40
41 // read-only property that gets
42 // commi ssi on empl oye's social security number
43 public string SocialSecurityNumber
44 {
45     get
46     {
47         return socialSecurityNumber;
48     } // end get
49 } // end property SocialSecurityNumber

```

Outline

Commi ssi onEmpl oye2.cs (2 of 4)

© 2006 Pearson Education, Inc. All rights reserved.

```

51 // property that gets and sets comml ssi on employee's gross sales
52 public decimal GrossSales
53 {
54     get
55     {
56         return grossSales;
57     } // end get
58     set
59     {
60         grossSales = ( value < 0 ) ? 0 : value;
61     } // end set
62 } // end property GrossSales
63
64 // property that gets and sets comml ssi on employee's comml ssi on rate
65 public decimal Comml ssi onRate
66 {
67     get
68     {
69         return comml ssi onRate;
70     } // end get
71     set
72     {
73         comml ssi onRate = ( value > 0 && value < 1 ) ? value : 0;
74     } // end set
75 } // end property Comml ssi onRate

```

Outline

Comml ssi on
Empl oyee2. cs
(3 of 4)

© 2006 Pearson Education, Inc. All rights reserved.

```

77 // calculate comml ssi on employee's pay
78 public virtual decimal Earnings()
79 {
80     return comml ssi onRate * grossSales;
81 } // end method Earnings
82
83 // return string representation of Comml ssi onEmpl oyee object
84 public override string ToString()
85 {
86     return string.Format(
87         "(0): (1) (2)\n(3): (4)\n(5): (6)\n(7): (8)\n(9): (9)",
88         "comml ssi on empl oyee", firstName, lastName,
89         "social security number", socialSecurityNumber,
90         "gross sales", grossSales, "comml ssi on rate", comml ssi onRate );
91 } // end method ToString
92 // end class Comml ssi onEmpl oyee2

```

Outline

Mark Earnings as **virtual** so the derived class can override the method

Empl oyee2. cs
(4 of 4)

© 2006 Pearson Education, Inc. All rights reserved.

```

1 // Fig. 10.11: BasePI usComml ssi onEmpl oyee3. cs
2 // BasePI usComml ssi onEmpl oyee3 inherits from Comml ssi onEmpl oyee2 and has
3 // access to Comml ssi onEmpl oyee2's protected members.
4 public class BasePI usComml ssi onEmpl oyee3 : Comml ssi onEmpl oyee2
5 {
6     private decimal baseSalary; // base salary per week
7
8     // six-parameter derived class constructor
9     // with call to base class Comml ssi onEmpl oyee constructor
10    public BasePI usComml ssi onEmpl oyee3( string first, string last,
11        string ssn, decimal sales, decimal rate, decimal salary )
12        : base( first, last, ssn, sales, rate )
13    {
14        baseSalary = salary; // validate base salary via property
15    } // end six-parameter BasePI usComml ssi onEmpl oyee3 constructor
16
17    // property that gets and sets
18    // base-salaried comml ssi on employee's base salary
19    public decimal BaseSalary
20    {
21        get
22        {
23            return baseSalary;
24        } // end get
25        set
26        {
27            baseSalary = ( value < 0 ) ? 0 : value;
28        } // end set
29    } // end property BaseSalary

```

Outline

BasePI usComml ssi on
Empl oyee3. cs
(1 of 2)

Must call base class's constructor

© 2006 Pearson Education, Inc. All rights reserved.

```

30 // calculate earnings
31 public override decimal Earnings()
32 {
33     return baseSalary * ( comml ssi onRate * grossSales );
34 } // end method Earnings
35
36 // return string representation of BasePI usComml ssi onEmpl oyee3
37 public override string ToString()
38 {
39     return string.Format(
40         "(0): (1) (2)\n(3): (4)\n(5): (6)\n(7): (8)\n(9): (9)",
41         "base-salaried comml ssi on empl oyee", firstName, lastName,
42         "social security number", socialSecurityNumber,
43         "gross sales", grossSales, "comml ssi on rate", comml ssi onRate,
44         "base salary", baseSalary );
45 } // end method ToString
46 // end class BasePI usComml ssi onEmpl oyee3

```

Outline

Overrides base class's Earnings method

BasePI usComml ssi on
Empl oyee3. cs
Directly access base class's protected instance variables

© 2006 Pearson Education, Inc. All rights reserved.

```

1 // Fig. 10.12: BasePI usComml ssi onEmpl oyeeTest3. cs
2 // Testing class BasePI usComml ssi onEmpl oyee3.
3 using System;
4
5 public class BasePI usComml ssi onEmpl oyeeTest3
6 {
7     // instantiate BasePI usComml ssi onEmpl oyee3 object
8     // instantiate BasePI usComml ssi onEmpl oyee3 object
9     // instantiate BasePI usComml ssi onEmpl oyee3 object
10    new BasePI usComml ssi onEmpl oyee3( "Bob", "Lew",
11        "333-33-3333", 5000.00, 0.04, 300.00 );
12
13    // display base-salaried comml ssi on employee data
14    Console.WriteLine(
15        "Employee information obtained by properties and methods:\n" );
16    Console.WriteLine( " (0) (1)", "First name is",
17        basePI usComml ssi onEmpl oyee3.firstName );
18    Console.WriteLine( " (0) (1)", "Last name is",
19        basePI usComml ssi onEmpl oyee3.lastName );
20    Console.WriteLine( " (0) (1)", "Social security number is",
21        basePI usComml ssi onEmpl oyee3.socialSecurityNumber );
22    Console.WriteLine( " (0) (1-C)", "Gross sales are",
23        basePI usComml ssi onEmpl oyee3.grossSales );
24    Console.WriteLine( " (0) (1-F2)", "Comml ssi on rate is",
25        basePI usComml ssi onEmpl oyee3.comml ssi onRate );
26    Console.WriteLine( " (0) (1-C)", "Earnings are",
27        basePI usComml ssi onEmpl oyee3.Earnings() );
28    Console.WriteLine( " (0) (1-C)", "Base salary is",
29        basePI usComml ssi onEmpl oyee3.BaseSalary );
30

```

Outline

Instantiate BasePI usComml ssi onEmpl oyee3 object

BasePI usComml ssi on
Empl oyeeTest3. cs
(1 of 2)

Use BasePI usComml ssi onEmpl oyee3's properties to retrieve and change the object's instance variable values

© 2006 Pearson Education, Inc. All rights reserved.

```

31 BasePI usComml ssi onEmpl oyee bob = new BasePI usComml ssi onEmpl oyee(
32     "Bob", "Lew", "333-33-3333", 5000.00, 0.04, 300.00 );
33
34 Console.WriteLine( "\n(0):\n\n(1)",
35     "Updated employee information obtained by ToString",
36     bob.ToString() );
37 Console.WriteLine( "\n(0):\n\n(1)",
38     "Updated employee information obtained by ToString",
39     bob.Earnings() );
40 } // end Main
41 // end class BasePI usComml ssi onEmpl oyeeTest3

```

Outline

Employee information obtained by properties and methods:

First name is Bob
Last name is Lew
Social security number is 333-33-3333
Gross sales are 50,000.00
Comml ssi on rate is 0.04
Earnings are 5000.00
Base salary is 300.00

Updated employee information obtained by ToString:

base-salaried comml ssi on empl oyee: Bob Lew
social security number: 333-33-3333
gross sales: 50,000.00
comml ssi on rate: 0.04
base salary: 300.00
earnings: 5,000.00

Implicitly call the object's ToString method

BasePI usComml ssi on
Empl oyeeTest3. cs
(2 of 2)

© 2006 Pearson Education, Inc. All rights reserved.

10.4.4 Commissions on Employee - BasePlusCommissionEmployee Inheritance Hierarchy Using protected Instance Variables (Cont.)

43

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



© 2006 Pearson Education, Inc. All rights reserved.

10.4.5 Commissions on Employee - BasePlusCommissionEmployee Inheritance Hierarchy Using private Instance Variables

44

- Reexamine hierarchy
 - Use the best software engineering practice
 - Declare instance variables as **private**
 - Provide **public** *get* and *set* accessors
 - Use *get* accessor to obtain values of instance variables



© 2006 Pearson Education, Inc. All rights reserved.

```

1 // Fig. 10.13: CommissionEmployee3.cs
2 // CommissionEmployee3 class represents a commission employee.
3 public class CommissionEmployee3
4 {
5     private string firstName;
6     private string lastName;
7     private string socialSecurityNumber;
8     private decimal grossSales; // gross weekly sales
9     private decimal commissionRate; // commission percentage
10
11 // five-parameter constructor
12 public CommissionEmployee3( string first, string last, string ssn,
13     decimal sales, decimal rate )
14 {
15     // implicit call to object constructor occurs here
16     firstName = first;
17     lastName = last;
18     socialSecurityNumber = ssn;
19     GrossSales = sales; // validate gross sales via property
20     CommissionRate = rate; // validate commission rate via property
21 } // end five-parameter CommissionEmployee3 constructor
22
23 // read-only property that gets commission employee's first name
24 public string FirstName
25 {
26     get
27     {
28         return firstName;
29     } // end get
30 } // end property FirstName
    
```

Outline

CommissionEmployee3.cs
(1 of 2)

Declare private instance variables

© 2006 Pearson Education, Inc. All rights reserved.

```

31 // read-only property that gets commission employee's last name
32 public string LastName
33 {
34     get
35     {
36         return lastName;
37     } // end get
38 } // end property LastName
39
40 // read-only property that gets
41 // commission employee's social security number
42 public string SocialSecurityNumber
43 {
44     get
45     {
46         return socialSecurityNumber;
47     } // end get
48 } // end property SocialSecurityNumber
    
```

Outline

CommissionEmployee3.cs
(2 of 4)

© 2006 Pearson Education, Inc. All rights reserved.

```

50 // property that gets and sets commission employee's gross sales
51 public decimal GrossSales
52 {
53     get
54     {
55         return grossSales;
56     } // end get
57     set
58     {
59         grossSales = ( value < 0 ) ? 0 : value;
60     } // end set
61 } // end property GrossSales
62
63 // property that gets and sets commission employee's commission rate
64 public decimal CommissionRate
65 {
66     get
67     {
68         return commissionRate;
69     } // end get
70     set
71     {
72         commissionRate = ( value > 0 && value < 1 ) ? value : 0;
73     } // end set
74 } // end property CommissionRate
    
```

Outline

CommissionEmployee3.cs
(3 of 4)

© 2006 Pearson Education, Inc. All rights reserved.

```

75 // calculate commission on employee's pay
76 public virtual decimal Earnings()
77 {
78     return CommissionRate * GrossSales;
79 } // end method Earnings
80
81 // return string representation of CommissionEmployee object
82 public override string ToString()
83 {
84     return string.Format(
85         "(0): {1} {2}\n(3): {4}\n(5): {6:C}\n(7): {8:F2}",
86         "commission employee", FirstName, LastName,
87         "social security number", SocialSecurityNumber,
88         "gross sales", GrossSales, "commission rate", CommissionRate );
89 } // end method ToString
90
91 // end class CommissionEmployee3
    
```

Outline

CommissionEmployee3.cs
(4 of 4)

Use properties to obtain the values of instance variables

© 2006 Pearson Education, Inc. All rights reserved.

```

1 // Fig. 10.14: BasePI usCommi ssi onEmpI oye4. cs
2 // BasePI usCommi ssi onEmpI oye4 inherits from Commi ssi onEmpI oye3 and has
3 // access to Commi ssi onEmpI oye3's private data via
4 // its public properties.
5 public class BasePI usCommi ssi onEmpI oye4 : Commi ssi onEmpI oye3
6 {
7     private decimal baseSalary; // base salary per week
8
9     // si x-parameter derived class constructor
10    // with call to base class Commi ssi onEmpI oye3 constructor
11    public BasePI usCommi ssi onEmpI oye4( string first, string last,
12        string ssn, decimal sales, decimal rate, decimal salary )
13        : base( first, last, ssn, sales, rate )
14    {
15        BaseSalary = salary; // validate base salary via property
16    } // end si x-parameter BasePI usCommi ssi onEmpI oye4 constructor
17
18    // property that gets and sets
19    // base-salaried commi ssi on employee's base salary
20    public decimal BaseSalary
21    {
22        get
23        {
24            return baseSalary;
25        } // end get
26        set
27        {
28            baseSalary = ( value < 0 ) ? 0 : value;
29        } // end set
30    } // end property BaseSalary

```

Outline

Inherits from Commi ssi onEmpI oye3

BasePI usCommi ssi onEmpI oye4. cs (1 of 2)

© 2006 Pearson Education, Inc. All rights reserved.

```

31 // calculate earnings
32 public override decimal Earnings()
33 {
34     return BaseSalary + base.Earnings();
35 } // end method Earnings
36
37 // return string representation of BasePI usCommi ssi onEmpI oye4
38 public override string ToString()
39 {
40     return string.Format( "{0} {1}\n{2}: {3:0}",
41         "base-salaried", base.ToString(), "base salary", BaseSalary );
42 } // end method ToString
43 } // end class BasePI usCommi ssi onEmpI oye4

```

Outline

Invoke an overridden base class method from a derived class

BasePI usCommi ssi onEmpI oye4. cs (1 of 2)

Use properties to obtain the values of instance variables

Invoke an overridden base class method from a derived class

© 2006 Pearson Education, Inc. All rights reserved.

```

1 // Fig. 10.15: BasePI usCommi ssi onEmpI oyeTest4. cs
2 // Testing class BasePI usCommi ssi onEmpI oye4.
3 using System;
4
5 public class BasePI usCommi ssi onEmpI oyeTest4
6 {
7     public static void Main( string[] args )
8     {
9         // instantiate BasePI usCommi ssi onEmpI oye3 object
10        BasePI usCommi ssi onEmpI oye4 empI oye =
11            new BasePI usCommi ssi onEmpI oye4( "Bob", "Lewis",
12                "333-33-3333", 8000.00, 0.04, 300.00 );
13
14        // display base-salaried commi ssi on employee data
15        Console.WriteLine(
16            "Employee information obtained by properties and methods: \n" );
17        Console.WriteLine( "{0} {1}", "First name is",
18            empI oye.FirstName );
19        Console.WriteLine( "{0} {1}", "Last name is",
20            empI oye.LastName );
21        Console.WriteLine( "{0} {1}", "Social security number is",
22            empI oye.Social SecurityNumber );
23        Console.WriteLine( "{0} {1:0}", "Gross sales are",
24            empI oye.GrossSales );
25        Console.WriteLine( "{0} {1:F2}", "Commi ssi on rate is",
26            empI oye.Commi ssi onRate );
27        Console.WriteLine( "{0} {1:0}", "Earnings are",
28            empI oye.Earnings() );
29        Console.WriteLine( "{0} {1:0}", "Base salary is",
30            empI oye.BaseSalary );

```

Outline

BasePI usCommi ssi onEmpI oyeTest4. cs (1 of 2)

Create BasePI usCommi ssi onEmpI oye4 object.

Use inherited properties to access inherited private instance variables

Use BasePI usCommi ssi onEmpI oye4 properties to access private instance variable.

© 2006 Pearson Education, Inc. All rights reserved.

```

31 employee.BaseSalary = 3000.00; // set base salary
32
33 Console.WriteLine( "\n{0}\n\n{1}",
34     "Updated employee information obtained by ToString", employee );
35 Console.WriteLine( "earnings: {0:C}", employee.Earnings() );
36 } // end Main
37 } // end class BasePI usCommi ssi onEmpI oyeTest4

```

Outline

BasePI usCommi ssi onEmpI oyeTest4. cs (2 of 2)

Employee information obtained by properties and methods:

```

First name is Bob
Last name is Lewis
Social security number is 333-33-3333
Gross sales are $5,000.00
Commi ssi on rate is 0.04
Earnings are $500.00
Base salary is $300.00

```

Updated employee information obtained by ToString:

```

base-salaried commi ssi on employee: Bob Lewis
social security number: 333-33-3333
gross sales: $5,000.00
commi ssi on rate: 0.04
base salary: $1,000.00
earnings: $1,200.00

```

© 2006 Pearson Education, Inc. All rights reserved.

10.5 Constructors in Derived Classes

- Instantiating derived class object
 - Chain of constructor calls
 - Derived class constructor invokes base class constructor
 - Implicitly or explicitly
 - Base of inheritance hierarchy
 - Last constructor called in chain is object's constructor
 - Original derived class constructor's body finishes executing last
 - Example: Commi ssi onEmpI oye3-BasePI usCommi ssi onEmpI oye4 hierarchy
 - Commi ssi onEmpI oye3 constructor called second last (last is Object constructor)
 - Commi ssi onEmpI oye3 constructor's body finishes execution second (first is Object constructor's body)

© 2006 Pearson Education, Inc. All rights reserved.

```

1 // Fig. 10.16: Commi ssi onEmpI oye4. cs
2 // Commi ssi onEmpI oye4 class represents a commi ssi on employee.
3 using System;
4
5 public class Commi ssi onEmpI oye4
6 {
7     private string firstName;
8     private string lastName;
9     private string socialSecurityNumber;
10    private decimal grossSales; // gross weekly sales
11    private decimal commi ssi onRate; // commi ssi on percentage
12
13    // five-parameter constructor
14    public Commi ssi onEmpI oye4( string first, string last, string ssn,
15        decimal sales, decimal rate )
16    {
17        // implicit call to object constructor occurs here
18        firstName = first;
19        lastName = last;
20        socialSecurityNumber = ssn;
21        GrossSales = sales; // validate gross sales via property
22        Commi ssi onRate = rate; // validate commi ssi on rate via property
23
24        Console.WriteLine( "Commi ssi onEmpI oye4 constructor:\n" + this );
25    } // end five-parameter Commi ssi onEmpI oye4 constructor

```

Outline

Commi ssi onEmpI oye4. cs (1 of 4)

Constructor outputs message to demonstrate method call order.

© 2006 Pearson Education, Inc. All rights reserved.

```

1 // Fig. 10.17: BasePlusCommi ssi onEmp l oye e5. cs
2 // BasePlusCommi ssi onEmp l oye e5 cl ass decl arati on.
3 using System;
4
5 public class BasePlusCommi ssi onEmp l oye e5 : Commi ssi onEmp l oye e4
6 {
7     private decimal baseSalary; // base salary per week
8
9     // si x-parameter deri ved cl ass constructor
10    // w/ th call to base class Commi ssi onEmp l oye e4 constructor
11    public BasePlusCommi ssi onEmp l oye e5( string fI rst, string l ast,
12        string ssn, decimal sal es, decimal rate, decimal salary )
13        : base( fI rst, l ast, ssn, sal es, rate )
14    {
15        BaseSalary = salary; // val i date base salary vi a property
16    }
17
18    Console. Wri t eLi ne(
19        "\nBasePlusCommi ssi onEmp l oye e5 constructor: " + thi s );
20    } // end si x-parameter BasePlusCommi ssi onEmp l oye e5 constructor
21
22    // property that gets and sets
23    // base-sal ari ed commi ssi on emp l oye e's base salary
24    public decimal BaseSalary
25    {
26        get
27        {
28            return baseSalary;
29        }
30    } // end get

```

Outline
BasePlusCommi ssi onEmp l oye e5. cs
(1 of 2)
Constructor outputs message to demonstrate method call order.

© 2006 Pearson Education, Inc. All rights reserved.

```

1 // Fig. 10.18: ConstructorTest. cs
2 // Di spl ay order i n whi ch base cl ass and deri ved cl ass constructors
3 // are cal l ed.
4 using System;
5
6 public class ConstructorTest
7 {
8     public static void Main( string[] args )
9     {
10        Commi ssi onEmp l oye e4 emp l oye e1 = new Commi ssi onEmp l oye e4( "Bob",
11            "Lewis", "333-33-3333", 5000.00W, 0.04 );
12
13        Console. Wri t eLi ne( );
14        BasePlusCommi ssi onEmp l oye e5 emp l oye e2 =
15            new BasePlusCommi ssi onEmp l oye e5( "Li sa", "Jones",
16                "555-55-5555", 2000.00W, 0.06, 500.00M );
17
18        Console. Wri t eLi ne( );
19        BasePlusCommi ssi onEmp l oye e5 emp l oye e3 =
20            new BasePlusCommi ssi onEmp l oye e5( "Mark", "Sends",
21                "888-88-8888", 8000.00W, 0.15, 2000.00M );
22    } // end Main
23 } // end class ConstructorTest

```

Outline
ConstructorTest. cs
(1 of 2)
Instantiate Commi ssi onEmp l oye e4 object
Instantiate two BasePlusCommi ssi onEmp l oye e5 objects to demonstrate order of derived class and base class constructor method calls.

© 2006 Pearson Education, Inc. All rights reserved.

```

Commi ssi onEmp l oye e4 constructor:
commi ssi on emp l oye e: Bob Lewis
social security number: 333-33-3333
gross sal es: $5,000.00
commi ssi on rate: 0.04

Commi ssi onEmp l oye e4 constructor:
base-sal ari ed commi ssi on emp l oye e: Li sa Jones
social security number: 555-55-5555
gross sal es: $2,000.00
commi ssi on rate: 0.06
base salary: $500.00

BasePlusCommi ssi onEmp l oye e5 constructor:
base-sal ari ed commi ssi on emp l oye e: Li sa Jones
social security number: 555-55-5555
gross sal es: $2,000.00
commi ssi on rate: 0.06
base salary: $500.00

Commi ssi onEmp l oye e4 constructor:
base-sal ari ed commi ssi on emp l oye e: Mark Sends
social security number: 888-88-8888
gross sal es: $8,000.00
commi ssi on rate: 0.15
base salary: $0.00

BasePlusCommi ssi onEmp l oye e5 constructor:
base-sal ari ed commi ssi on emp l oye e: Mark Sends
social security number: 888-88-8888
gross sal es: $8,000.00
commi ssi on rate: 0.15
base salary: $2,000.00

```

Outline
ConstructorTest. cs
(2 of 2)

© 2006 Pearson Education, Inc. All rights reserved.

10.7 Class object

- Class object methods
 - Equal s
 - FInal I ze
 - GetHashCode
 - GetType
 - Memberwi seCl one
 - ReferenceEqual s
 - ToStri ng

© 2006 Pearson Education, Inc. All rights reserved.

Method	Description
Equal s	This method compares two objects for equality and returns true if they are equal and false otherwise. The method takes any object as an argument. When objects of a particular class must be compared for equality, the class should override method Equal s to compare the contents of the two objects. The method's implementation should meet the following requirements: <ul style="list-style-type: none"> • It should return false if the argument is null. • It should return true if an object is compared to itself, as in <code>obj ect1. Equal s(obj ect1)</code>. • It should return true only if both <code>obj ect1. Equal s(obj ect2)</code> and <code>obj ect2. Equal s(obj ect1)</code> would return true. • For three objects, if <code>obj ect1. Equal s(obj ect2)</code> returns true and <code>obj ect2. Equal s(obj ect3)</code> returns true, then <code>obj ect1. Equal s(obj ect3)</code> should also return true. • A class that overrides the method Equal s should also override the method GetHashCode to ensure that equal objects have identical hashcodes. The default Equal s implementation determines only whether two references refer to the same object in memory.
FInal I ze	This method cannot be explicitly declared or called. When a class contains a destructor, the compiler implicitly renames it to override the protected method FInal I ze , which is called only by the garbage collector before it reclaims an object's memory. The garbage collector is not guaranteed to reclaim an object, thus it is not guaranteed that an object's FInal I ze method will execute. When a derived class's FInal I ze method executes, it performs its task, then invokes the base class's FInal I ze method. FInal I ze 's default implementation is a placeholder that simply invokes the base class's FInal I ze method.

Fig. 10.19 | Object methods that are inherited directly or indirectly by all classes.
(Part 1 of 2)

© 2006 Pearson Education, Inc. All rights reserved.

Method	Description
GetHashCode	A hashtable is a data structure that relates one object, called the key, to another object, called the value. We discuss Hashtabl e in Chapter 27, Collections. When initially inserting a value into a hashtable, the key's GetHashCode method is called. The hashcode value returned is used by the hashtable to determine the location at which to insert the corresponding value. The key's hashcode is also used by the hashtable to locate the key's corresponding value.
GetType	Every object knows its own type at execution time. Method GetType (used in Section 11.5) returns an object of class Type (namespace System) that contains information about the object's type, such as its class name (obtained from Type property FullName).
Memberwi seCl one	This protected method, which takes no arguments and returns an object reference, makes a copy of the object on which it is called. The implementation of this method performs a shallow copy—instance variable values in one object are copied into another object of the same type. For reference types, only the references are copied.
Reference-Equal s	This static method takes two object arguments and returns true if two objects are the same instance or if they are null references. Otherwise, it returns false.
ToStri ng	This method (introduced in Section 7.4) returns a string representation of an object. The default implementation of this method returns the namespace followed by a dot and the class name of the object's class.

Fig. 10.19 | Object methods that are inherited directly or indirectly by all classes.
(Part 2 of 2)

© 2006 Pearson Education, Inc. All rights reserved.