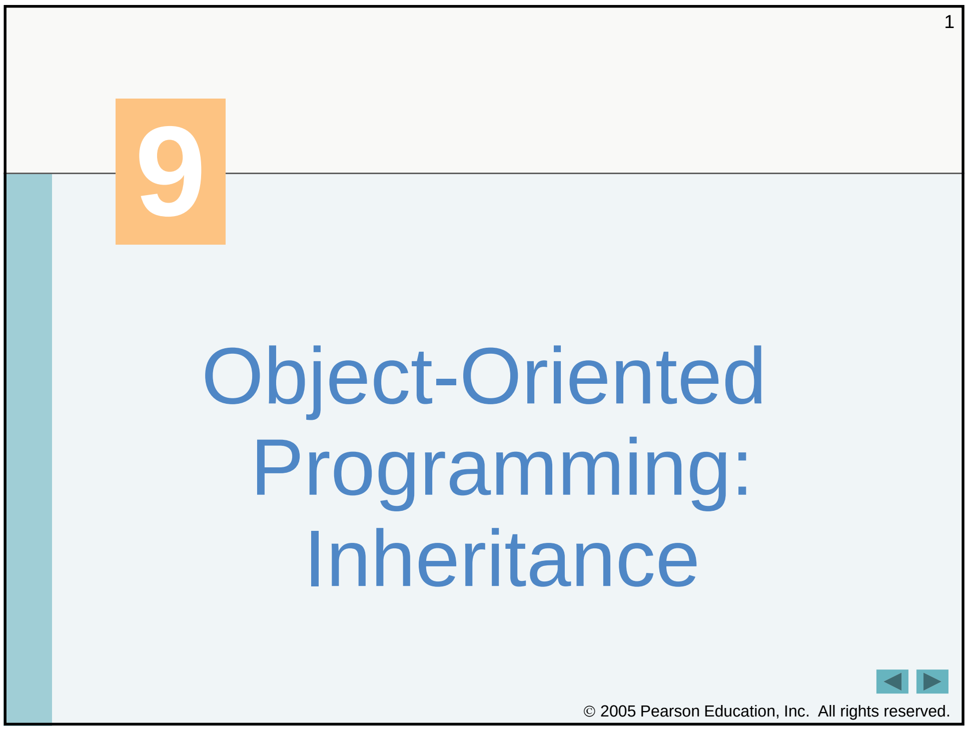
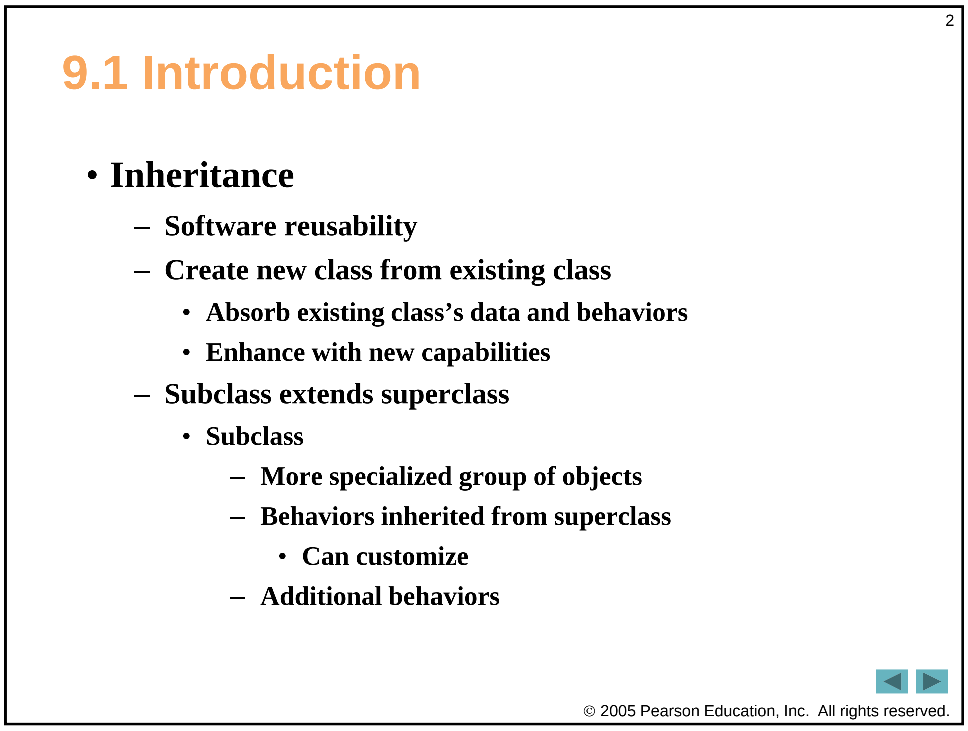


9

Object-Oriented Programming: Inheritance



9.1 Introduction

- **Inheritance**
  - **Software reusability**
  - **Create new class from existing class**
    - **Absorb existing class's data and behaviors**
    - **Enhance with new capabilities**
  - **Subclass extends superclass**
    - **Subclass**
      - **More specialized group of objects**
      - **Behaviors inherited from superclass**
        - **Can customize**
      - **Additional behaviors**

9.1 Introduction (Cont.)

- Class hierarchy
  - Direct superclass
    - Inherited explicitly (one level up hierarchy)
  - Indirect superclass
    - Inherited two or more levels up hierarchy
  - Single inheritance
    - Inherits from one superclass
  - Multiple inheritance
    - Inherits from multiple superclasses
      - Java does not support multiple inheritance

9.2 Superclasses and subclasses

- Superclasses and subclasses
  - Object of one class “is an” object of another class
    - Example: Rectangle is quadrilateral.
      - Class `Rectangle` inherits from class `Quadrilateral`
      - `Quadrilateral`: superclass
      - `Rectangle`: subclass
  - Superclass typically represents larger set of objects than subclasses
    - Example:
      - superclass: `Vehicle`
        - Cars, trucks, boats, bicycles, …
      - subclass: `Car`
        - Smaller, more-specific subset of vehicles

Superclass	Subclasses
`Student`	`GraduateStudent`, `UndergraduateStudent`
`Shape`	`Circle`, `Triangle`, `Rectangle`
`Loan`	`CarLoan`, `HomeImprovementLoan`, `MortgageLoan`
`Employee`	`Faculty`, `Staff`
`BankAccount`	`CheckingAccount`, `SavingsAccount`

Fig. 9.1 | Inheritance examples.

9.2 Superclasses and subclasses (Cont.)

- Inheritance hierarchy
  - Inheritance relationships: tree-like hierarchy structure
  - Each class becomes
    - superclass
      - Supply members to other classes

    OR
    - subclass
      - Inherit members from other classes

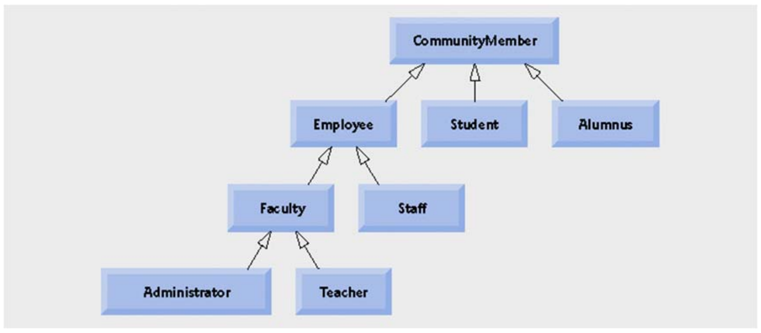


Fig. 9.2 | Inheritance hierarchy for university `CommunityMembers`

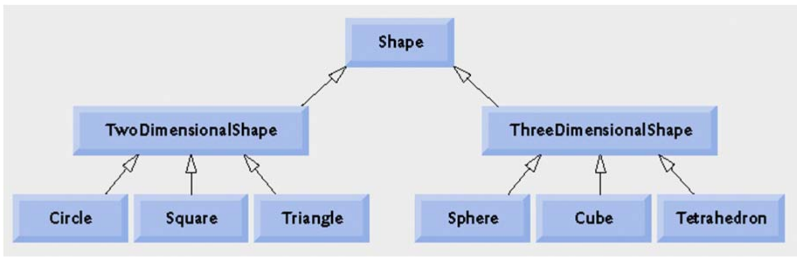
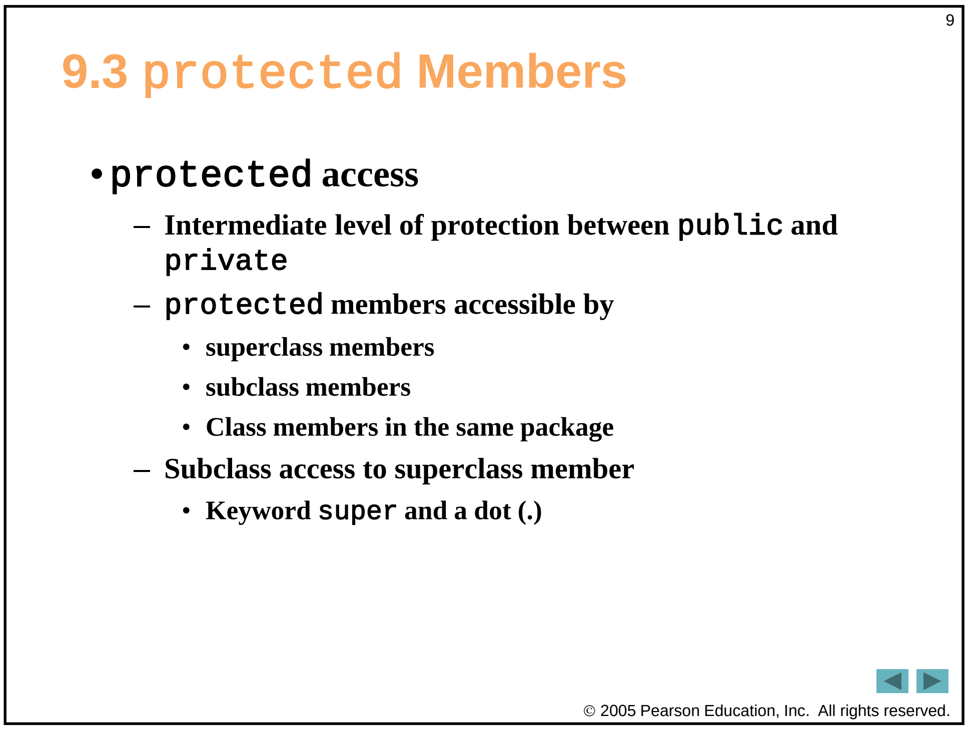
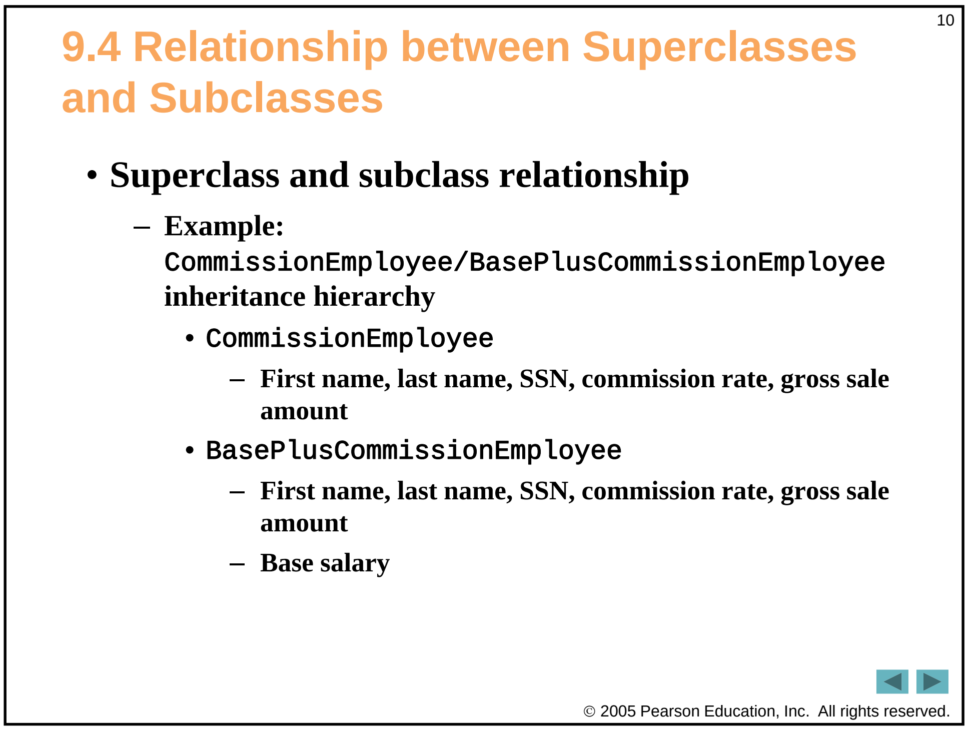


Fig. 9.3 | Inheritance hierarchy for Shapes.



9.3 `protected` Members

- `protected` access
  - Intermediate level of protection between `public` and `private`
  - `protected` members accessible by
    - superclass members
    - subclass members
    - Class members in the same package
  - Subclass access to superclass member
    - Keyword `super` and a dot (.)



9.4 Relationship between Superclasses and Subclasses

- Superclass and subclass 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

9.4.1 Creating and Using a `CommissionEmployee` Class

- Class `CommissionEmployee`
  - Extends class `Object`
    - Keyword `extends`
    - Every class in Java extends an existing class
      - Except `Object`
    - Every class inherits `Object`'s methods
    - New class implicitly extends Object
      - If it does not extend another class

```
1  // Fig. 9.4: CommissionEmployee.java
2  // CommissionEmployee class represents a commission empl
3
4  public class CommissionEmployee extends Object
5  {
6     private String firstName;
7     private String lastName;
8     private String socialSecurityNumber;
9     private double grossSales; // gross weekly sales
10    private double commissionRate; // commission percent
11
12    // five-argument constructor
13    public CommissionEmployee( String first, String last, String ssn,
14       double sales, double rate )
15    {
16       // implicit call to Object constructor occurs h
17       firstName = first;
18       lastName = last;
19       socialSecurityNumber = ssn;
20       setGrossSales( sales ); // validate and store gross sales
21       setCommissionRate( rate ); // validate and store commission rate
22    } // end five-argument CommissionEmployee constructor
23
24    // set first name
25    public void setFirstName( String first )
26    {
27       firstName = first;
28    } // end method setFirstName
29
```

Declare `private` instance variables

Class `CommissionEmployee` `extends` class `Object`

Implicit call to `Object` constructor

Initialize instance variables

Invoke methods `setGrossSales` and `setCommissionRate` to validate data

Outline

`issionEmployee`
`.java`

(1 of 4)

Line 4

Lines 6-10

Line 1

Lines 20-21

Outline

```java
30    // return first name
31    public String getFirstName()
32    {
33       return firstName;
34    } // end method getFirstName
35
36    // set last name
37    public void setLastName( String last )
38    {
39       lastName = last;
40    } // end method setLastName
41
42    // return last name
43    public String getLastName()
44    {
45       return lastName;
46    } // end method getLastName
47
48    // set social security number
49    public void setSocialSecurityNumber( String ssn )
50    {
51       socialSecurityNumber = ssn; // should validate
52    } // end method setSocialSecurityNumber
53
54    // return social security number
55    public String getSocialSecurityNumber()
56    {
57       return socialSecurityNumber;
58    } // end method getSocialSecurityNumber
59
```

`CommissionEmployee.java`

(2 of 4)

Outline

```java
60    // set gross sales amount
61    public void setGrossSales( double sales )
62    {
63       grossSales = ( sales < 0.0 ) ? 0.0 : sales;
64    } // end method setGrossSales
65
66    // return gross sales amount
67    public double getGrossSales()
68    {
69       return grossSales;
70    } // end method getGrossSales
71
72    // set commission rate
73    public void setCommissionRate( double rate )
74    {
75       commissionRate = ( rate > 0.0 && rate < 1.0 ) ? rate : 0.0;
76    } // end method setCommissionRate
77
78    // return commission rate
79    public double getCommissionRate()
80    {
81       return commissionRate;
82    } // end method getCommissionRate
83
84    // calculate earnings
85    public double earnings()
86    {
87       return commissionRate * grossSales;
88    } // end method earnings
89
```

Calculate earnings

`CommissionEmployee.java`

(3 of 4)

Lines 85-88

Outline

```
90    // return String representation of CommissionEmployee object
91    public String toString()
92    {
93       return String.format( "%s: %s %s\n%s: %s\n%s: %
94          "commission employee", firstName, lastName,
95          "social security number", socialSecurityNumber,
96          "gross sales", grossSales,
97          "commission rate", commissionRate );
98    } // end method toString
99 } // end class CommissionEmployee
```

Override method `toString` of class `Object`

CommissionEmployee.java

(4 of 4)

Lines 91-98

Outline

```
1  // Fig. 9.5: CommissionEmployeeTest.java
2  // Testing class CommissionEmployee.
3
4  public class CommissionEmployeeTest
5  {
6     public static void main( String args[] )
7     {
8        // instantiate CommissionEmployee object
9        CommissionEmployee employee = new CommissionEmployee(
10          "Sue", "Jones", "222-22-2222", 10000, .06 );
11
12       // get commission employee data
13       System.out.println(
14          "Employee information obtained by get methods: \n" );
15       System.out.printf( "%s %s\n", "
16          employee.getFirstName() );
17       System.out.printf( "%s %s\n", "
18          employee.getLastName() );
19       System.out.printf( "%s %s\n", "Social security number is",
20          employee.getSocialSecurityNumber() );
21       System.out.printf( "%s %.2f\n", "Gross sales is",
22          employee.getGrossSales() );
23       System.out.printf( "%s %.2f\n", "Commiss
24          employee.getCommissionRate() );
25
26       employee.setGrossSales( 500 ); // set gross sales
27       employee.setCommissionRate( .1 ); // set commission rate
28
```

Instantiate `CommissionEmployee` object

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

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

CommissionEmployeeTest.java

(1 of 2)

Lines 9-10

Lines 15-25

26-27

```
29       System.out.printf( "\n%s:\n\n%s\n",
30          "Updated employee information obtained by toString", employee );
31    } // end main
32 } // end class CommissionEmployeeTest
```

```
Employee information obtained by get methods:

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 obtained by toString:

commission employee: Sue Jones
social security number: 222-22-2222
gross sales: 500.00
commission rate: 0.10
```

Implicitly call object's `toString` method

Outline

`onEmployee`
`Test.java`

(2 of 2)

Line 30

Program output

9.4.2 Creating a `BasePlusCommissionEmployee` Class without Using Inheritance

- Class `BasePlusCommissionEmployee`
  - Implicitly extends `Object`
  - Much of the code is similar to `CommissionEmployee`
    - `private` instance variables
    - `public` methods
    - constructor
  - Additions
    - `private` instance variable `baseSalary`
    - Methods `setBaseSalary` and `getBaseSalary`

Outline

```
1  // Fig. 9.6: BasePlusCommissionEmployee.java
2  // BasePlusCommissionEmployee class represents an employee that receives
3  // a base salary in addition to commission.
4
5  public class BasePlusCommissionEmployee
6  {
7     private String firstName;
8     private String lastName;
9     private String socialSecurityNumber;
10    private double grossSales; // gross weekly sales
11    private double commissionRate; // commission percentage
12    private double baseSalary; // base salary per week
13
14    // six-argument constructor
15    public BasePlusCommissionEmployee( String first, String last,
16       String ssn, double sales, double rate, double salary )
17    {
18       // implicit call to Object constructor occurs here
19       firstName = first;
20       lastName = last;
21       socialSecurityNumber = ssn;
22       setGrossSales( sales ); // validate and store
23       setCommissionRate( rate ); // validate and store commission rate
24       setBaseSalary( salary ); // validate and store base salary
25    } // end six-argument BasePlusCommissionEmployee constructor
26
```

Add instance variable `baseSalary`

Use method `setBaseSalary` to validate data

`BasePlusCommissionEmployee.java`

Line 12

Line 24

Outline

```
27    // set first name
28    public void setFirstName( String first )
29    {
30       firstName = first;
31    } // end method setFirstName
32
33    // return first name
34    public String getFirstName()
35    {
36       return firstName;
37    } // end method getFirstName
38
39    // set last name
40    public void setLastName( String last )
41    {
42       lastName = last;
43    } // end method setLastName
44
45    // return last name
46    public String getLastName()
47    {
48       return lastName;
49    } // end method getLastName
50
51    // set social security number
52    public void setSocialSecurityNumber( String ssn )
53    {
54       socialSecurityNumber = ssn; // should validate
55    } // end method setSocialSecurityNumber
56
```

`BasePlusCommissionEmployee.java`

(2 of 4)

Outline

```java
57    // return social security number
58    public String getSocialSecurityNumber()
59    {
60       return socialSecurityNumber;
61    } // end method getSocialSecurityNumber
62
63    // set gross sales amount
64    public void setGrossSales( double sales )
65    {
66       grossSales = ( sales < 0.0 ) ? 0.0 : sales;
67    } // end method setGrossSales
68
69    // return gross sales amount
70    public double getGrossSales()
71    {
72       return grossSales;
73    } // end method getGrossSales
74
75    // set commission rate
76    public void setCommissionRate( double rate )
77    {
78       commissionRate = ( rate > 0.0 && rate < 1.0 ) ? rate : 0.0;
79    } // end method setCommissionRate
80
81    // return commission rate
82    public double getCommissionRate()
83    {
84       return commissionRate;
85    } // end method getCommissionRate
86
```

`BasePlusCommission`
`Employee.java`

(3 of 4)

Outline

```java
87    // set base salary
88    public void setBaseSalary( double salary )
89    {
90       baseSalary = ( salary < 0.0 ) ? 0.0 : salary;
91    } // end method setBaseSalary
92
93    // return base salary
94    public double getBaseSalary()
95    {
96       return baseSalary;
97    } // end method getBaseSalary
98
99    // calculate ea
100      public double
101      {
102         return baseSalary + ( commissionRate * grossSales );
103      } // end method earnings
104
105      // return String representation of BasePlusCommissionEmployee
106      public String toString()
107      {
108         return String.format(
109            "%s: %s %s\n%s: %s\n%s: %.2f\n%s: %.2f\n%s: %.2f",
110            "base-salaried commission employee", firstName, lastName,
111            "social security number", socialSecurityNumber,
112            "gross sales", grossSales, "commission rate'
113            "base salary", baseSalary );
114      } // end method toString
115   } // end class BasePlusCommissionEmployee
```

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

Method `getBaseSalary` returns the value of instance variable `baseSalary`

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

Update method `toString` to display base salary

`BasePlusCommission`
`Employee.java`

(4 of 4)

Lines 88-91

Lines 94-97

Line 102

Lines 108-113

Outline

```java
1  // Fig. 9.7: BasePlusCommissionEmployeeTest.java
2  // Testing class BasePlusCommissionEmployee.
3
4  public class BasePlusCommissionEmployeeTest
5  {
6     public static void main( String args[] )
7     {
8        // instantiate BasePlusCommissionEmployee object
9        BasePlusCommissionEmployee employee =
10          new BasePlusCommissionEmployee(
11          "Bob", "Lewis", "333-33-3333", 5000, .04, 300 );
12
13       // get base-salaried commission employee data
14       System.out.println(
15          "Employee information obtained by get methods: \n" );
16       System.out.printf( "%s %s\n",
17          employee.getFirstName() );
18       System.out.printf( "%s %s\n",
19          employee.getLastName() );
20       System.out.printf( "%s %s\n", "Social security number is",
21          employee.getSocialSecurityNumber() );
22       System.out.printf( "%s %.2f\n", "Gross sales is",
23          employee.getGrossSales() );
24       System.out.printf( "%s %.2f\n", "Commission rate is",
25          employee.getCommissionRate() );
26       System.out.printf( "%s %.2f\n", "Base salary is",
27          employee.getBaseSalary() );
28
```

Instantiate `BasePlusCommissionEmployee` object

Use `BasePluCommissionEmployee`’s *get* methods to retrieve the object’s instance variable values

sePlusCommission ployeeTest.java

(1 of 2)

Line 9-11

Lines 16-27

Outline

```java
29       employee.setBaseSalary( 1000 ); // set base salary
30
31       System.out.printf( "\n%s:\n\n%s\n",
32          "Updated employee information obta
33          employee.toString() );
34    } // end main
35 } // end class BasePlusCommissionEmplo
```

```
Employee information obtained by get met

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 obtained by toString:

base-salaried commission employee: Bob Lewis
social security number: 333-33-3333
gross sales: 5000.00
commission rate: 0.04
base salary: 1000.00
```

Use `BasePlusCommissionEmployee`’s `setBaseSalary` methods to set base salary

Explicitly call object’s `toString` method

BasePlusCommission EmployeeTest.java

(2 of 2)

Line 29

Line 33

Program output

Software Engineering Observation 9.4

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

Software Engineering Observation 9.5

With inheritance, the common instance variables and methods of all the classes in the hierarchy are declared in a superclass. When changes are required for these common features, software developers need only to make the changes in the superclass—subclasses 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.

9.4.3 Creating a CommissionEmployee-BasePlusCommiionEmployee Inheritance Hierarchy

- Class `BasePlusCommissionEmployee2`
  - Extends class `CommissionEmployee`
  - Is a `CommissionEmployee`
  - Has instance variable `baseSalary`
  - Inherits `public` and `protected` members
  - Constructor not inherited

Outline

```
1  // Fig. 9.8: BasePlusCommissionEmployee2.java
2  // BasePlusCommissionEmployee2 inherits from class CommissionEmployee.
3
4  public class BasePlusCommissionEmployee2 extends CommissionEmployee
5  {
6     private double baseSalary; // base salary per week
7
8     // six-argument constructor
9     public BasePlusCommissionEmployee2( Stri
10       String ssn, double sales, double rate, double salary )
11    {
12       // explicit call to superclass CommissionEmployee constructor
13       super( first, last, ssn, sales, rate );
14
15       setBaseSalary( amount ); // validate and store base salary
16    } // end six-argument BasePlusCommissi
17
18    // set base salary
19    public void setBaseSalary( double salary )
20    {
21       baseSalary = ( salary < 0.0 ) ? 0.0 : salary;
22    } // end method setBaseSalary
23
```

Class `BasePluCommissionEmployee2` is a subclass of `CommissionEmployee`

Invoke the superclass constructor using the superclass constructor call syntax

`BasePlusCommission`
`ee2.java`

Line 4

Line 13

Outline

`BasePlusCommission`

(2 of 3)

Line 34

Lines 41-46

```java
24    // return base salary
25    public double getBaseSalary()
26    {
27       return baseSalary;
28    } // end method getBaseSalary
29
30    // calculate earnings
31    public double earnings()
32    {
33       // not allowed: commissionRate and grossSales private in superclass
34       return baseSalary + ( commissionRate * grossSales );
35    } // end method earnings
36
37    // return String representation
38    public String toString()
39    {
40       // not allowed: attempts to a
41       return String.format(
42          "%s: %s %s\n%s: %s\n%s: %.2f\n%s: %.2f\n%s: %.2f",
43          "base-salaried commission employee", firstName, lastName,
44          "social security number", socialSecurityNumber,
45          "gross sales", grossSales, "commission rate", commissionRate,
46          "base salary", baseSalary );
47    } // end method toString
48 } // end class BasePlusCommissionEmployee2
```

Compiler generates errors because superclass’s instance variable `commissionRate` and `grossSales` are `private`

Compiler generates errors because superclass’s instance variable `firstName`, `lastName`, `socialSecurityNumber`, `grossSales` and `commissionRate` are `private`

Outline

`BasePlusCommissionEmployee2.java`

(3 of 3)

Compiler generated errorss

```
BasePlusCommissionEmployee2.java:34: commissionRate has private access in
CommissionEmployee
      return baseSalary + ( commissionRate * grossSales );
                            ^
BasePlusCommissionEmployee2.java:34: grossSales has private access in
CommissionEmployee
      return baseSalary + ( commissionRate * grossSales );
                                             ^
BasePlusCommissionEmployee2.java:43: firstName has private access in
CommissionEmployee
         "base-salaried commission employee", firstName, lastName,
                                              ^
BasePlusCommissionEmployee2.java:43: lastName has private access in
CommissionEmployee
         "base-salaried commission employee", firstName, lastName,
                                                         ^
BasePlusCommissionEmployee2.java:44: socialSecurityNumber has private access in
CommissionEmployee
         "social security number", socialSecurityNumber,
                                   ^
BasePlusCommissionEmployee2.java:45: grossSales has private access in
CommissionEmployee
         "gross sales", grossSales, "commission rate", commissionRate,
                        ^
BasePlusCommissionEmployee2.java:45: commissionRate has private access in
CommissionEmployee
         "gross sales", grossSales, "commission rate", commissionRate,
                                                       ^
7 errors
```

Common Programming Error 9.2

A compilation error occurs if a subclass constructor calls one of its superclass constructors with arguments that do not match exactly the number and types of parameters specified in one of the superclass constructor declarations.

9.4.4 `CommissionEmployee`-`BasePlusCommissionEmployee` Inheritance Hierarchy Using `protected` Instance Variables

- Use `protected` instance variables
  - Enable class `BasePlusCommissionEmployee` to directly access superclass instance variables
  - Superclass's `protected` members are inherited by all subclases of that superclass

Outline

`CommissionEmployee2.java`

(1 of 4)

Line 6-10

```
1  // Fig. 9.9: CommissionEmployee2.java
2  // CommissionEmployee2 class represents a commission employee.
3
4  public class CommissionEmployee2
5  {
6     protected String firstName;
7     protected String lastName;
8     protected String socialSecurityNumber;
9     protected double grossSales; // gross weekly sales
10    protected double commissionRate; // commission percentage
11
12    // five-argument constructor
13    public CommissionEmployee2( String first, String last, String ssn,
14       double sales, double rate )
15    {
16       // implicit call to Object constructor occurs here
17       firstName = first;
18       lastName = last;
19       socialSecurityNumber = ssn;
20       setGrossSales( sales ); // validate and store gross sales
21       setCommissionRate( rate ); // validate and store commission rate
22    } // end five-argument CommissionEmployee2 constructor
23
24    // set first name
25    public void setFirstName( String first )
26    {
27       firstName = first;
28    } // end method setFirstName
29
```

Declare `protected` instance variables

Outline

`CommissionEmployee2.java`

(2 of 4)

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

Outline

`Commission`

`Employee2.java`

(3 of 4)

```java
   // set gross sales amount
   public void setGrossSales( double sales )
   {
      grossSales = ( sales < 0.0 ) ? 0.0 : sales;
   } // end method setGrossSales

   // return gross sales amount
   public double getGrossSales()
   {
      return grossSales;
   } // end method getGrossSales

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

   // return commission rate
   public double getCommissionRate()
   {
      return commissionRate;
   } // end method getCommissionRate

   // calculate earnings
   public double earnings()
   {
      return commissionRate * grossSales;
   } // end method earnings

```

Outline

`Commission`

`Employee2.java`

(4 of 4)

```java
   // return String representation of CommissionEmployee2 object
   public String toString()
   {
      return String.format( "%s: %s %s\n%s: %s\n%s: %.2f\n%s: %.2f",
         "commission employee", firstName, lastName,
         "social security number", socialSecurityNumber,
         "gross sales", grossSales,
         "commission rate", commissionRate );
   } // end method toString
} // end class CommissionEmployee2
```

Outline

BasePlusCommissionEmployee3.java

of 2)

Line 13

```
1  // Fig. 9.10: BasePlusCommissionEmployee3.java
2  // BasePlusCommissionEmployee3 inherits from CommissionEmployee2 and has
3  // access to CommissionEmployee2's protected members.
4
5  public class BasePlusCommissionEmployee3 extends CommissionEmployee2
6  {
7     private double baseSalary; // base salary per week
8
9     // six-argument constructor
10    public BasePlusCommissionEmployee3( String first, String last,
11       String ssn, double sales, double rate, double salary )
12    {
13       super( first, last, ssn, sales, rate );
14       setBaseSalary( salary ); // validate and store base salary
15    } // end six-argument BasePlusCommissionEmployee3 constructor
16
17    // set base salary
18    public void setBaseSalary( double salary )
19    {
20       baseSalary = ( salary < 0.0 ) ? 0.0 : salary;
21    } // end method setBaseSalary
22
23    // return base salary
24    public double getBaseSalary()
25    {
26       return baseSalary;
27    } // end method getBaseSalary
28
```

Must call superclass's constructor

Outline

BasePlusCommissionEmployee3.java

Line 32

Lines 38-43

```
29    // calculate earnings
30    public double earnings()
31    {
32       return baseSalary + ( commissionRate * grossSales );
33    } // end method earnings
34
35    // return String representation of BasePlusCommissionEmployee3
36    public String toString()
37    {
38       return String.format(
39          "%s: %s %s\n%s: %s\n%s: %.2f\n%s: %.2f\n%s: %.2f",
40          "base-salaried commission employee", firstName, lastName,
41          "social security number", socialSecurityNumber,
42          "gross sales", grossSales, "commission rate", commissionRate,
43          "base salary", baseSalary );
44    } // end method toString
45 } // end class BasePlusCommissionEmployee3
```

Directly access superclass's `protected` instance variables

Outline

BasePlusCommissionEmployeeTest3.java

(1 of 2)

```java
// Fig. 9.11: BasePlusCommissionEmployeeTest3.java
// Testing class BasePlusCommissionEmployee3.

public class BasePlusCommissionEmployeeTest3
{
   public static void main( String args[] )
   {
      // instantiate BasePlusCommissionEmployee3 object
      BasePlusCommissionEmployee3 employee =
         new BasePlusCommissionEmployee3(
         "Bob", "Lewis", "333-33-3333", 5000, .04, 300 );

      // get base-salaried commission employee data
      System.out.println(
         "Employee information obtained by get methods: \n" );
      System.out.printf( "%s %s\n", "First name is",
         employee.getFirstName() );
      System.out.printf( "%s %s\n", "Last name is",
         employee.getLastName() );
      System.out.printf( "%s %s\n", "Social security number is",
         employee.getSocialSecurityNumber() );
      System.out.printf( "%s %.2f\n", "Gross sales is",
         employee.getGrossSales() );
      System.out.printf( "%s %.2f\n", "Commission rate is",
         employee.getCommissionRate() );
      System.out.printf( "%s %.2f\n", "Base salary is",
         employee.getBaseSalary() );

```

Outline

BasePlusCommissionEmployeeTest3.java

(2 of 2)

Program output

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

      System.out.printf( "\n%s:\n\n%s\n",
         "Updated employee information obtained by toString",
         employee.toString() );
   } // end main
} // end class BasePlusCommissionEmployeeTest3
```

```
Employee information obtained by get methods:

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 obtained by toString:

base-salaried commission employee: Bob Lewis
social security number: 333-33-3333
gross sales: 5000.00
commission rate: 0.04
base salary: 1000.00
```

9.4.4 `CommissionEmployee-BasePlusCommissionEmployee` Inheritance Hierarchy Using protected Instance Variables (Cont.)

- Using `protected` instance variables
  - Advantages
    - subclasses can modify values directly
    - Slight increase in performance
      - Avoid set/get method call overhead
  - Disadvantages
    - No validity checking
      - subclass can assign illegal value
    - Implementation dependent
      - subclass methods more likely dependent on superclass implementation
      - superclass implementation changes may result in subclass modifications
        - Fragile (brittle) software

Error-Prevention Tip 9.1

When possible, do not include `protected` instance variables in a superclass. Instead, include non-`private` methods that access `private` instance variables. This will ensure that objects of the class maintain consistent states.

9.4.5 `CommissionEmployee`-`BasePlusCommissionEmployee` Inheritance Hierarchy Uing `private` Instance Variables

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

Outline

```
1  // Fig. 9.12: CommissionEmployee3.java
2  // CommissionEmployee3 class represents a commission employee.
3
4  public class CommissionEmployee3
5  {
6     private String firstName;
7     private String lastName;
8     private String socialSecurityNumber;
9     private double grossSales; // gross weekly sales
10    private double commissionRate; // commission percentage
11
12    // five-argument constructor
13    public CommissionEmployee3( String first, String last, String ssn,
14       double sales, double rate )
15    {
16       // implicit call to Object constructor occurs here
17       firstName = first;
18       lastName = last;
19       socialSecurityNumber = ssn;
20       setGrossSales( sales ); // validate and store gross sales
21       setCommissionRate( rate ); // validate and store commission rate
22    } // end five-argument CommissionEmployee3 constructor
23
24    // set first name
25    public void setFirstName( String first )
26    {
27       firstName = first;
28    } // end method setFirstName
29
```

Declare `private` instance variables

`CommissionEmployee3.java`

(1 of 4)

Lines 6-10

```
   // return first name
   public String getFirstName()
   {
      return firstName;
   } // end method getFirstName

   // set last name
   public void setLastName( String last )
   {
      lastName = last;
   } // end method setLastName

   // return last name
   public String getLastName()
   {
      return lastName;
   } // end method getLastName

   // set social security number
   public void setSocialSecurityNumber( String ssn )
   {
      socialSecurityNumber = ssn; // should validate
   } // end method setSocialSecurityNumber

   // return social security number
   public String getSocialSecurityNumber()
   {
      return socialSecurityNumber;
   } // end method getSocialSecurityNumber

```

Outline

`Commission`

`Employee3.java`

(2 of 4)

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

   // return gross sales amount
   public double getGrossSales()
   {
      return grossSales;
   } // end method getGrossSales

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

   // return commission rate
   public double getCommissionRate()
   {
      return commissionRate;
   } // end method getCommissionRate

```

Outline

`Commission`

`Employee3.java`

(3 of 4)

Outline

```
84    // calculate earnings
85    public double earnings()
86    {
87       return getCommissionRate() * getGrossSales();
88    } // end method earnings
89
90    // return String representation of CommissionEmployee3 object
91    public String toString()
92    {
93       return String.format( "%s: %s %s\n%s: %s\n%s: %.2f\n%s: %.2f",
94          "commission employee", getFirstName(), getLastName(),
95          "social security number", getSocialSecurityNumber(),
96          "gross sales", getGrossSales(),
97          "commission rate", getCommissionRate() );
98    } // end method toString
99 } // end class CommissionEmployee3
```

Use *get* methods to obtain the values of instance variables

`ission`

`Employee3.java`

(4 of 4)

Line 87

Lines 94-97

Outline

```
1  // Fig. 9.13: BasePlusCommissionEmployee4.java
2  // BasePlusCommissionEmployee4 class inherits from CommissionEmployee3 and
3  // accesses CommissionEmployee3's private data via CommissionEmployee3's
4  // public methods.
5
6  public class BasePlusCommissionEmployee4 extends CommissionEmployee3
7  {
8     private double baseSalary; // base salary per week
9
10    // six-argument constructor
11    public BasePlusCommissionEmployee4( String first, String last,
12       String ssn, double sales, double rate, double salary )
13    {
14       super( first, last, ssn, sales, rate );
15       setBaseSalary( salary ); // validate and store base salary
16    } // end six-argument BasePlusCommissionEmployee4 constructor
17
18    // set base salary
19    public void setBaseSalary( double salary )
20    {
21       baseSalary = ( salary < 0.0 ) ? 0.0 : salary;
22    } // end method setBaseSalary
23
```

Inherits from `CommissionEmployee3`

`BasePlusCommission`
`Employee4.java`

)

```
24    // return base salary
25    public double getBaseSalary()
26    {
27       return baseSalary;
28    } // end method getBaseSalary
29
30    // calculate earnings
31    public double earnings()
32    {
33       return getBaseSalary() + super.earnings();
34    } // end method earnings
35
36    // return String representation of BasePlusCommissionEmploye
37    public String toString()
38    {
39       return String.format( "%s %s\n%s: %.2f", "base-salaried",
40          super.toString(), "base salary", getBaseSalary() );
41    } // end method toString
42 } // end class BasePlusCommissionEmployee4
```

Outline

ion

(2 of 2)

Line 33

Lines 40

Invoke an overridden superclass method from a subclass

Use *get* methods to obtain the values of instance variables

Invoke an overridden superclass method from a subclass

Common Programming Error 9.3

When a superclass method is overridden in a subclass, the subclass version often calls the superclass version to do a portion of the work. Failure to prefix the superclass method name with the keyword `super` and a dot (`.`) separator when referencing the superclass's method causes the subclass method to call itself, creating an error called infinite recursion. Recursion, used correctly, is a powerful capability discussed in Chapter 15, Recursion.

Outline

ssion
.java

Lines 9-11

Lines 16-25

```
1  // Fig. 9.14: BasePlusCommissionEmployeeTest4.java
2  // Testing class BasePlusCommissionEmployee4.
3
4  public class BasePlusCommissionEmployeeTest4
5  {
6     public static void main( String args[] )
7     {
8        // instantiate BasePlusCommissionEmployee4 object
9        BasePlusCommissionEmployee4 employee =
10          new BasePlusCommissionEmployee4(
11          "Bob", "Lewis", "333-33-3333", 5000, .04, 300 );
12
13       // get base-salaried commission employee data
14       System.out.println(
15          "Employee information obtained by get methods: \n" );
16       System.out.printf( "%s %s\n", "First name is",
17          employee.getFirstName() );
18       System.out.printf( "%s %s\n", "Last name is",
19          employee.getLastName() );
20       System.out.printf( "%s %s\n", "Social security number is",
21          employee.getSocialSecurityNumber() );
22       System.out.printf( "%s %.2f\n", "Gross sales is",
23          employee.getGrossSales() );
24       System.out.printf( "%s %.2f\n", "Commission rate is",
25          employee.getCommissionRate() );
26       System.out.printf( "%s %.2f\n", "Base salary
27          employee.getBaseSalary() );
28
```

Create `BasePlusCommissionEmployee4` object.

Use inherited *get* methods to access inherited `private` instance variables

Use `BasePlusCommissionEmployee4` *get* method to access `private` instance variable.

Outline

on
EmployeeTest4.java

(2 of 2)

```
29       employee.setBaseSalary( 1000 ); // set base salary
30
31       System.out.printf( "\n%s:\n\n%s\n",
32          "Updated employee information obtaine
33          employee.toString() );
34    } // end main
35 } // end class BasePlusCommissionEmployeeTest4
```

Use `BasePlusCommissionEmployee4` *set* method to modify `private` instance variable `baseSalary`.

```
Employee information obtained by get methods:

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 obtained by toString:

base-salaried commission employee: Bob Lewis
social security number: 333-33-3333
gross sales: 5000.00
commission rate: 0.04
base salary: 1000.00
```