



# Inheritance

## Lecture 5

**Department of Computer Engineering**  
**INHA University**  
**Dr. Tamer ABUHMED**



# Outline



- What is Inheritance?
- Subclass and Superclass
- “is-a” vs. “has-a” relationships
- Simple Inheritance Example
- Strategy for Coding with Inheritance
- protected Members
- Constructors in Subclasses
- Full Example with Inheritance



# What is Inheritance?

- *Inheritance* is one of the main techniques of object-oriented programming (OOP)
- Using this technique, a very general form of a class is first defined and compiled, and then more specialized versions of the class are defined by adding instance variables and methods
  - The specialized classes are said to *inherit* the methods and instance variables of the general class



# What is Inheritance?

- A form of software reuse
- Create a new class from an existing class
  - Absorb existing class data (fields) and methods
  - Enhance with new or modified capabilities
- Why do we use Inheritance?
  - Used to eliminate redundant code
- Example
  - Dog class inherits from Animal class
  - Dog *extends* Animal



# Subclass and Superclass

- Subclass extends superclass
  - Subclass
    - Also called *child class* or *derived class*
    - More specialized group of objects
    - Inherits data and methods from superclass
    - Can add or modify methods
      - Modifying methods is called *overriding*
  - Superclass
    - Also called *parent class* or *base class*
    - Typically represents larger group of objects
    - Supplies data and behaviors to subclass
    - May be direct or indirect
- Java does not support multiple inheritance



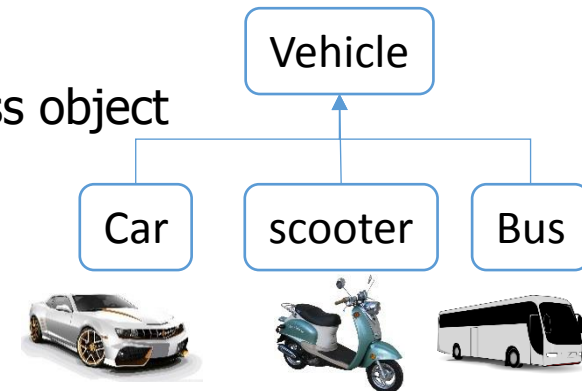
-

# “is-a” vs. “has-a” relationships



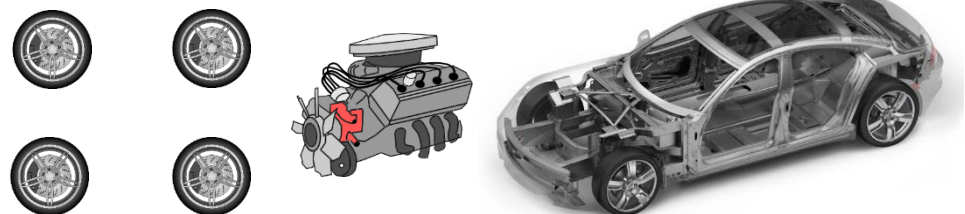
- “is-a”

- Represents ***inheritance***
- subclass object is an example of the superclass object
- Example: a Car *is a* Vehicle
- Car is subclass; Vehicle is superclass
- Keywords: extends, implements



- “has-a”

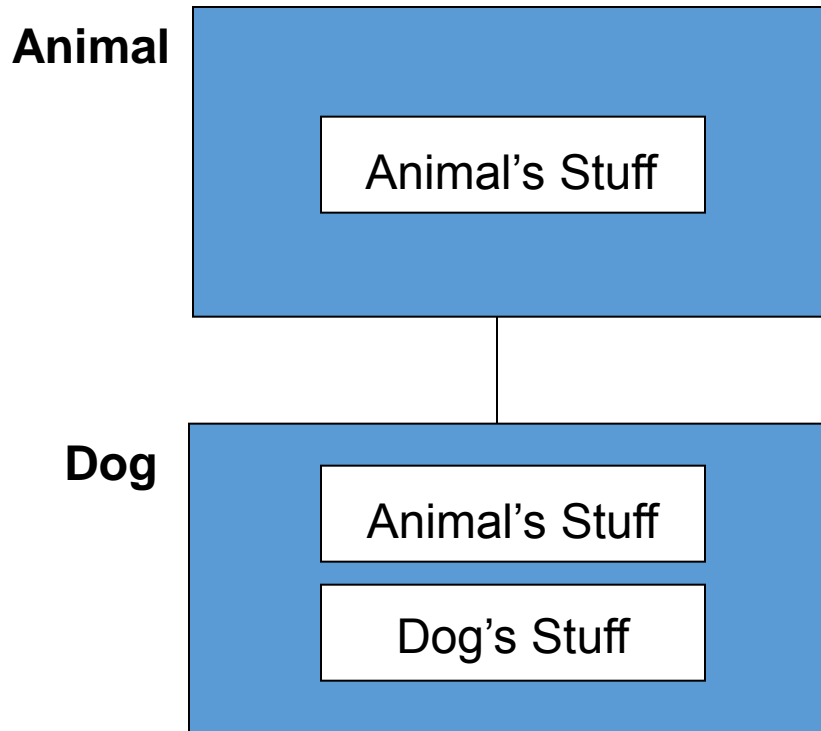
- Represents ***composition***
- Object contains one or more objects of other classes as members
- Example: Car *has a* Steering Wheel



# Inheritance



- Dog extends (“is-a”) Animal





# Simple Inheritance Example (1/2)

```
public class Animal
{
    private String name;

    public String getName()
    {
        return name;
    }

    public void setName(String name)
    {
        this.name = name;
    }

    public String voice()
    {
        return "?";
    }
}
```

```
public class Dog extends Animal
{
    public String voice()
    {
        return "WOOF!";
    }

    public void fetch(String toy)
    {
        System.out.println("Fetching a " + toy);
    }
}
```

This is an overridden method,  
Inherited from animal.

This is a new method.

# Simple Inheritance Example (2/2)



```
public class PetStore
{
    public static void main (String[] args)
    {
        Dog d = new Dog();
        d.setName("Henry");
        System.out.println (d.getName() + " says "
        + d.voice());
    }
}
```

**Output**

Henry says WOOF!

# Exercise



Draw an Inheritance Hierarchy for these classes:

Triangle  
Sphere  
2DShape  
Shape  
Pyramid  
Square  
3DShape  
Cube  
Circle

# Strategy for Coding with Inheritance



- Design classes for objects
- Identify **characteristics** classes have in **common**
  - Abstraction: focus on commonalities among objects in a system
- Design **superclasses** to store common characteristics
- Design the **subclasses** to store specialized characteristics



# Inherited Members

What members of the **superclass** are going to be inherited by the **subclass**?

Inherited	<i>Not</i> Inherited
<b>Public</b> members	<b>Constructors</b>
<b>Protected</b> members	<b>Private</b> methods, and Variables
Protected Variables	<b>Static</b> members

But public은 (접근은) 가능  
word : redefining  
just rule^^;

**static**  
상속의 개념 X  
접근의 개념 O

# protected Members



- Intermediate level of protection between `public` and `private`
- Accessible to
  - `superclasses`
  - `subclasses`
  - classes in the same `package`
- Use `super.` to access a superclass method that has been overridden by a subclass method
- **Recommendation:** Don't use protected instance variables!
  - "Fragile" software can "break" if superclass changes



# Constructors in Subclasses

- Constructors are *not* inherited!
- Chain of constructor calls
  - subclass constructor invokes superclass constructor
    - Implicitly or explicitly
    - To call explicitly, use `super()`
    - Superclass constructor call must be first statement in subclass constructor
  - Object constructor is always fired last
- All instance variables are inherited
  - Private variables not directly accessible

# Full Example V1.0 (1/2)

```
public class CommissionEmployee extends Object
{
    private String firstName;
    private String lastName;
    private String socialSecurityNumber;
    private double grossSales; // gross weekly sales
    private double commissionRate; // commission percentage

    // five-argument constructor
    public CommissionEmployee( String first, String last, String ssn,
        double sales, double rate )
    {
        // implicit call to Object constructor occurs here
        firstName = first;
        lastName = last;
        socialSecurityNumber = ssn;
        setGrossSales( sales ); // validate and store gross sales
        setCommissionRate( rate ); // validate and store commission rate
    } // end five-argument CommissionEmployee constructor

    // set first name
    public void setFirstName( String first )
    {
        firstName = first; // should validate
    } // end method setFirstName

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

    // set last name
    public void setLastName( String last )
    {
        lastName = last; // should validate
    } // 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
    }
}
```

```
public String getSocialSecurityNumber()
{
    return socialSecurityNumber;
} // end method getSocialSecurityNumber

// set gross sales amount
public void setGrossSales( double sales )
{
    if ( sales >= 0.0 )
        grossSales = sales;
    else
        throw new IllegalArgumentException(
            "Gross sales must be >= 0.0" );
} // end method setGrossSales

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

// set commission rate
public void setCommissionRate( double rate )
{
    if ( rate > 0.0 && rate < 1.0 )
        commissionRate = rate;
    else
        throw new IllegalArgumentException(
            "Commission rate must be > 0.0 and < 1.0" );
} // 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

// return String representation of CommissionEmployee object
@Override
// indicates that this method overrides a superclass method
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 CommissionEmployee
```



# Full Example V1.0 (2/2)

```
public class BasePlusCommissionEmployee extends CommissionEmployee
{
    private double baseSalary; // base salary per week

    // six-argument constructor
    public BasePlusCommissionEmployee( String first, String last,
        String ssn, double sales, double rate, double salary )
    {
        // explicit call to superclass CommissionEmployee constructor
        super( first, last, ssn, sales, rate );

        setBaseSalary( salary ); // validate and store base salary
    } // end six-argument BasePlusCommissionEmployee constructor

    // set base salary
    public void setBaseSalary( double salary )
    {
        if ( salary >= 0.0 )
            baseSalary = salary;
        else
            throw new IllegalArgumentException(
                "Base salary must be >= 0.0" );
    } // end method setBaseSalary

    // return base salary
    public double getBaseSalary()
    {
        return baseSalary;
    } // end method getBaseSalary

    // calculate earnings
    @Override // indicates that this method overrides a superclass method
    public double earnings()
    {
        // not allowed: commissionRate and grossSales private in superclass
        return baseSalary + ( commissionRate * grossSales );
    } // end method earnings
```

```
// return String representation of
BasePlusCommissionEmployee
@Override // indicates that this method overrides a
superclass method
public String toString()
{
    // not allowed: attempts to access private superclass
members
    return String.format(
        "%s: %s %s\n%s: %s\n%s: %.2f\n%s: %.2f\n%s: %.2f",
        "base-salaried commission employee", firstName,
        lastName, "social security number", socialSecurityNumber,
        "gross sales", grossSales, "commission rate",
        commissionRate, "base salary", baseSalary );
} // end method toString
} // end class BasePlusCommissionEmployee
```

```
public class BasePlusCommissionEmployeeTest
{
    public static void main( String[] args )
    {
        // instantiate BasePlusCommissionEmployee object
        BasePlusCommissionEmployee employee =
            new BasePlusCommissionEmployee(
                "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() );

        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 BasePlusCommissionEmployeeTest
```

# Compilation Errors

## Output

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

Exception in thread "main" java.lang.Error: Unresolved compilation problems:

The field CommissionEmployee.firstName is not visible

The field CommissionEmployee.lastName is not visible

The field CommissionEmployee.socialSecurityNumber is not visible

The field CommissionEmployee.grossSales is not visible

The field CommissionEmployee.commissionRate is not visible

at BasePlusCommissionEmployee.toString(BasePlusCommissionEmployee.java:49)

at BasePlusCommissionEmployeeTest.main(BasePlusCommissionEmployeeTest.java:33)

# Compilation Errors



```
BasePlusCommissionEmployee.java:39: commissionRate has private access in  
CommissionEmployee
```

```
    return baseSalary + ( commissionRate * grossSales );  
                           ^
```

```
BasePlusCommissionEmployee.java:39: grossSales has private access in  
CommissionEmployee
```

```
    return baseSalary + ( commissionRate * grossSales );  
                           ^
```

```
BasePlusCommissionEmployee.java:49: firstName has private access in  
CommissionEmployee
```

```
    "base-salaried commission employee", firstName, lastName,  
                                         ^
```

```
BasePlusCommissionEmployee.java:49: lastName has private access in  
CommissionEmployee
```

```
    "base-salaried commission employee", firstName, lastName,  
                                                         ^
```

```
BasePlusCommissionEmployee.java:50: socialSecurityNumber has private access  
in CommissionEmployee
```

```
    "social security number", socialSecurityNumber,  
                               ^
```

```
BasePlusCommissionEmployee.java:51: grossSales has private access in  
CommissionEmployee
```

```
    "gross sales", grossSales, "commission rate", commissionRate,  
                   ^
```

```
BasePlusCommissionEmployee.java:51: commissionRate has private access in  
CommissionEmployee
```

```
    "gross sales", grossSales, "commission rate", commissionRate,  
                                                         ^
```

7 errors

# Full Example V2.0 (1/2)

```
public class CommissionEmployee extends Object
{
    protected String firstName;
    protected String lastName;
    protected String socialSecurityNumber;
    protected double grossSales; // gross weekly sales
    protected double commissionRate; // commission percentage

    // five-argument constructor
    public CommissionEmployee( String first, String last, String ssn,
        double sales, double rate )
    {
        // implicit call to Object constructor occurs here
        firstName = first;
        lastName = last;
        socialSecurityNumber = ssn;
        setGrossSales( sales ); // validate and store gross sales
        setCommissionRate( rate ); // validate and store commission rate
    } // end five-argument CommissionEmployee constructor

    // set first name
    public void setFirstName( String first )
    {
        firstName = first; // should validate
    } // end method setFirstName

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

    // set last name
    public void setLastName( String last )
    {
        lastName = last; // should validate
    } // 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
    }
}
```

```
    public String getSocialSecurityNumber()
    {
        return socialSecurityNumber;
    } // end method getSocialSecurityNumber

    // set gross sales amount
    public void setGrossSales( double sales )
    {
        if ( sales >= 0.0 )
            grossSales = sales;
        else
            throw new IllegalArgumentException(
                "Gross sales must be >= 0.0" );
    } // end method setGrossSales

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

    // set commission rate
    public void setCommissionRate( double rate )
    {
        if ( rate > 0.0 && rate < 1.0 )
            commissionRate = rate;
        else
            throw new IllegalArgumentException(
                "Commission rate must be > 0.0 and < 1.0" );
    } // 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

    // return String representation of CommissionEmployee object
    @Override
    // indicates that this method overrides a superclass method
    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 CommissionEmployee
```

# Full Example V2.0 (2/2)

```
public class BasePlusCommissionEmployee extends CommissionEmployee
{
    private double baseSalary; // base salary per week

    // six-argument constructor
    public BasePlusCommissionEmployee( String first, String last,
        String ssn, double sales, double rate, double salary )
    {
        // explicit call to superclass CommissionEmployee constructor
        super( first, last, ssn, sales, rate );

        setBaseSalary( salary ); // validate and store base salary
    } // end six-argument BasePlusCommissionEmployee constructor

    // set base salary
    public void setBaseSalary( double salary )
    {
        if ( salary >= 0.0 )
            baseSalary = salary;
        else
            throw new IllegalArgumentException(
                "Base salary must be >= 0.0" );
    } // end method setBaseSalary

    // return base salary
    public double getBaseSalary()
    {
        return baseSalary;
    } // end method getBaseSalary

    // calculate earnings
    @Override // indicates that this method overrides a superclass method
    public double earnings()
    {
        // not allowed: commissionRate and grossSales private in superclass
        return baseSalary + ( commissionRate * grossSales );
    } // end method earnings

    // return String representation of BasePlusCommissionEmployee
    @Override // indicates that this method overrides a superclass method
    public String toString()
    {
        // not allowed: attempts to access private superclass members
        return String.format(
            "%s: %s %s\n%s: %s\n%s: %.2f\n%s: %.2f\n%s: %.2f",
            "base-salaried commission employee", firstName, lastName,
            "social security number", socialSecurityNumber,
            "gross sales", grossSales, "commission rate", commissionRate,
            "base salary", baseSalary );
    } // end method toString
} // end class BasePlusCommissionEmployee
```

```
public class BasePlusCommissionEmployeeTest
{
    public static void main( String[] args )
    {
        // instantiate BasePlusCommissionEmployee object
        BasePlusCommissionEmployee employee =
            new BasePlusCommissionEmployee(
                "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() );

        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 BasePlusCommissionEmployeeTest
```

# Example V2.0 Output

## Output

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

# Full Example V3.0 (1/2)

```
public class CommissionEmployee
{
    private String firstName;
    private String lastName;
    private String socialSecurityNumber;
    private double grossSales; // gross weekly sales
    private double commissionRate; // commission percentage

    // five-argument constructor
    public CommissionEmployee( String first, String last, String ssn,
        double sales, double rate )
    {
        // implicit call to Object constructor occurs here
        firstName = first;
        lastName = last;
        socialSecurityNumber = ssn;
        setGrossSales( sales ); // validate and store gross sales
        setCommissionRate( rate );
    }

    // set first name
    public void setFirstName( String first )
    {
        firstName = first; // should validate
    } // end method setFirstName

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

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

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

```
    public void setSocialSecurityNumber( String ssn )
    {
        socialSecurityNumber = ssn; // should validate
    }

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

    public void setGrossSales( double sales )
    {
        if ( sales >= 0.0 )
            grossSales = sales;
        else
            throw new IllegalArgumentException(
                "Gross sales must be >= 0.0" );
    }

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

    public void setCommissionRate( double rate )
    {
        if ( rate > 0.0 && rate < 1.0 )
            commissionRate = rate;
        else
            throw new IllegalArgumentException(
                "Commission rate must be > 0.0 and < 1.0" );
    } // end method setCommissionRate

    public double getCommissionRate()
    {
        return commissionRate;
    }

    public double earnings()
    {
        return getCommissionRate() * getGrossSales();
    } // end method earnings

    @Override // indicates that this method overrides a superclass
    method
    public String toString()
    {
        return String.format( "%s: %s %s\n%s: %s\n%s: %.2f\n%s: %.2f",
            "commission employee", getFirstName(), getLastName(),
            "social security number", getSocialSecurityNumber(),
            "gross sales", getGrossSales(),
            "commission rate", getCommissionRate() );
    } // end method toString
} // end class CommissionEmployee
```

# Full Example V3.0 (2/2)

```
public class BasePlusCommissionEmployee extends CommissionEmployee
{
    private double baseSalary; // base salary per week

    // six-argument constructor
    public BasePlusCommissionEmployee( String first, String last,
        String ssn, double sales, double rate, double salary )
    {
        super( first, last, ssn, sales, rate );
        setBaseSalary( salary ); // validate and store base salary
    } // end six-argument BasePlusCommissionEmployee constructor

    // set base salary
    public void setBaseSalary( double salary )
    {
        if ( salary >= 0.0 )
            baseSalary = salary;
        else
            throw new IllegalArgumentException(
                "Base salary must be >= 0.0" );
    } // end method setBaseSalary

    // return base salary
    public double getBaseSalary()
    {
        return baseSalary;
    } // end method getBaseSalary

    // calculate earnings
    @Override // indicates that this method overrides a superclass method
    public double earnings()
    {
        return getBaseSalary() + super.earnings();
    } // end method earnings

    // return String representation of BasePlusCommissionEmployee
    @Override // indicates that this method overrides a superclass method
    public String toString()
    {
        return String.format( "%s %s\n%s: %.2f", "base-salaried",
            super.toString(), "base salary", getBaseSalary() );
    } // end method toString
} // end class BasePlusCommissionEmployee
```

```
public class BasePlusCommissionEmployeeTest
{
    public static void main( String[] args )
    {
        // instantiate BasePlusCommissionEmployee object
        BasePlusCommissionEmployee employee =
            new BasePlusCommissionEmployee(
                "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() );

        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 BasePlusCommissionEmployeeTest
```



# Example V3.0 Output

## Output

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

# Summary



- What is Inheritance?
- Subclass and Superclass
- “is-a” vs. “has-a” relationships
- Simple Inheritance Example
- Strategy for Coding with Inheritance
- protected Members
- Constructors in Subclasses
- Full Example with Inheritance