Principles of Computer Science II

Prof. Nadeem Abdul Hamid

CSC 121A – Spring 2005

Lecture Slides 2 -

OO Review, Scope, and Inheritance

Object-Oriented Programming

- Everything is an object
- Program is a bunch of objects
  - Tell each other what to do by sending messages (calling methods)
- Each object has its own memory made up of other objects
- Every object has a type
  - Each object is an *instance* of a class
- All objects of a particular type can receive the same messages

Definition of an Object

- ***An object has state, behavior and identity.*** ***[Booch]***
  - State: internal data (fields or instance variables)
  - Behavior: methods
  - Identity: each object uniquely distinguished from others; *i.e.* has a unique address in memory

Scope

- Determines the visibility and lifetime of variables
- In Java, scope determined by placement of curly braces {}

```
{  int x = 12;
  // Only x available
  {
    int q = 96;
    // Both x & q available
  }
  // Only x available
  // q "out of scope"
}
```


Object Lifetimes

- Not the same as primitive variables

```
{
   String s = "a string";
} // end of scope
```

- The reference, s, vanishes at the end of scope
- String object pointed to is still there
- Unlike other languages, don't worry about cleaning up memory
- Java uses a *garbage collector* to figure out which objects are no longer in use and reclaim their memory

Access Specifiers

- Every class member may have an access specifier before it: public/protected/private
- Package access
  - Also known as "friendly": default access which applies when you don't specify access
  - All classes in the same package have access to that class member (field/method)
  - To all classes outside the package, the member appears private

Access Specifiers (cont.)

- public: member is available to everyone
- private: only the class containing the member can access it

```
class Sundae {
  private Sundae() {}
  static Sundae makeASundae() {
    return new Sundae();
  }
}

public class IceCream {
  public static void main(String[] args) {
    //! Sundae x = new Sundae();
    Sundae x = Sundae.makeASundae();
  }
}
```

- protected: access granted to derived classes

Accessibility Table

External access	public	protected	(default) package	private
Same package	yes	yes	yes	no
Derived class in another package	yes	yes (inheritance only)	no	no
User code	yes	no	no	no

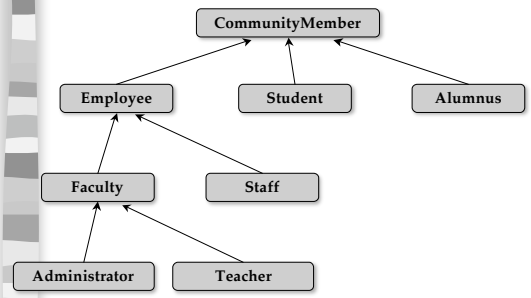
Inheritance

- Primary feature of OO programming for *software reuse*
- Define a new class by taking features of an existing class and modifying or extending them
  - Existing class: superclass
  - New, derived class: subclass
- A subclass is more specific than its superclass
- Subclasses and superclasses form a class hierarchy
  - Java class hierarchy starts with class `Object`

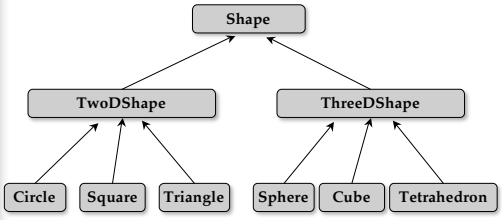
Inheritance Relationships

- "Is-a" vs. "Has-a" relationship
- "Has-a" relationship
  - Determines the fields of a class
  - `Employee` *has a* `Name`, `SSN`, `PayRate`, *etc.*
  - `Car` *has a* `SteeringWheel`, `FuelTank`, *etc.*
- "Is-a" relationship
  - Represented using inheritance
  - `HourlyEmployee` *is an* `Employee`, ...
  - `Car` *is a* `Vehicle`, `Boat` *is a* `Vehicle`, ...

Hierarchy for University Community Members



Shapes Hierarchy



CommissionEmployee Class

```
// CommissionEmployee.java
// Class represents employee paid on commission
//
// Nadeem Abdul Hamid (based on Deitel & Deitel Ch. 9)
// CSC 121 - Spring 2005
//

public class CommissionEm
    private String name;
    private double sales;    // gross weekly sales
    private double rate;     // commission percentage

    public CommissionEmployee( String name, double sales,
                               double rate ) {
        this.name = name;
        setSales( sales );   // validate and store gross sales
        setRate( rate );     // validate and store commission rate
    } // end CommissionEmployee constructor

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

    // return name
    public String getName() { return name; }
```

CSC121 Standard header format

CommissionEmployee Class

```
// CommissionEmployee.java
// Class represents employee paid on commission
//
// Nadeem Abdul Hamid (based on Deitel & Deitel Ch. 9)
// CSC 121 - Spring 2005
//

public class CommissionEmployee extends Object {
    private String name;     // full name
    private double sales;    // gross weekly sales
    private
    public
        this.name = name;
        setSales( sales );   // validate and store gross sales
        setRate( rate );     // validate and store commission rate
    } // end CommissionEmployee constructor

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

    // return name
    public String getName() { return name; }
```

Every Java class directly or indirectly inherits Object's methods (explicitly or implicitly) -- only included here for demonstration purposes

CommissionEmployee Class

```
// CommissionEmployee.java
// Class represents employee paid on commission
//
// Nadeem Abdul Hamid (based on Deitel & Deitel Ch. 9)
// CSC 121 - Spring 2005
//

public class CommissionEmployee extends Object {
    private String name;     // full name
    private double sales;    // gross weekly sales
    private double rate;     // commission percentage

    public CommissionEmployee( String name, double sales,
                               double rate ) {
        this.name = name;
        setSales( sales );   // validate and store gross sales
        setRate( rate );     // validate and store commission rate
    } // end CommissionEmployee constructor

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

    // return name
    public String getName() { return name; }
```

```
    // set gross sales amount
    public void setSales( double sales ) {
        this.sales = ( sales < 0.0 ) ? 0.0 : sales;
    }

    // return gross sales amount
    public double getSales() { return sales; }

    // set commission rate
    public void setRate( double rate ) {
        this.rate = ( rate > 0.0 && rate < 1.0 ) ? rate : 0.0;
    }

    // return commission rate
    public double getRate() { return rate; }

    // calculate earnings
    public double earnings() { return rate * sales; }

    // return String representation of CommissionEmployee object
    public String toString() {
        return "commission employee: " + name + "\n" +
            "gross sales: " + sales + "\n" +
            "commission rate: " + rate;
    } // end method toString
} // end class CommissionEmployee
```

conditional operator (ternary)

style to help match braces

```
//
// CommissionEmployeeTest.java
// Testing class CommissionEmployee
//
// Nadeem Abdul Hamid (based on Deitel & Deitel Ch. 9)
// CSC 121 - Spring 2005
//

public class CommissionEmployeeTest {

    public static void main( String args[] ) {
        // instantiate CommissionEmployee object
        CommissionEmployee empl
            = new CommissionEmployee( "Sue Jones", 10000, 0.06 );

        // get commission employee data
        System.out.println( "Employee info obtained by get methods: \n" );
        System.out.println( "Name is " + empl.getName() );
        System.out.println( "Gross sales are " + empl.getSales() );
        System.out.println( "Commission rate is " + empl.getRate() );

        empl.setSales( 500 );
        empl.setRate( .1 );

        System.out.println( "\nUpdated employee info obtained by toString:\n\n"
                            + empl + "\n" );
    } // end main

} // end class CommissionEmployeeTest
```

```
public class BasePlusCommissionEmployee {
    private String name;     // full name
    private double sales;    // gross weekly sales
    private double rate;     // commission percentage
    private double salary;   // base salary per week

    public BasePlusCommissionEmployee( String name, double sales,
                                       double rate, double salary ) {
        this.name = name;
        setSales( sales );   // validate and store gross sales
        setRate( rate );     // validate and store commission rate
        setSalary( salary ); // validate and store salary
    } // end BasePlusCommissionEmployee constructor

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

    // return name
    public String getName() { return name; }

    // set gross sales amount
    public void setSales( double sales ) {
        this.sales = ( sales < 0.0 ) ? 0.0 : sales;
    }

    // return gross sales amount
    public double getSales() { return sales; }

    // set commission rate
    public void setRate( double rate ) {
        this.rate = ( rate > 0.0 && rate < 1.0 ) ? rate : 0.0;
    }
```

```
    // return commission rate
    public double getRate() { return rate; }

    // set weekly salary
    public void setSalary( double salary ) {
        this.salary = ( salary < 0.0 ) ? 0.0 : salary;
    }

    // return weekly salary
    public double getSalary() { return salary; }

    // calculate earnings
    public double earnings() { return salary + ( rate * sales ); }

    // return String representation of BasePlusCommissionEmployee object
    public String toString() {
        return "commission employee: " + name + "\n" +
            "gross sales: " + sales + "\n" +
            "commission rate: " + rate + "\n" +
            "base salary: " + salary;
    } // end method toString

} // end class BasePlusCommissionEmployee
```



```
public class BasePlusCommissionEmployee2 extends CommissionEmployee {
    private double salary;   // base salary per week

    public BasePlusCommissionEmployee2( String name, double sales,
                                        double rate, double salary ) {
        super( name, sales, rate );
        setSalary( salary ); // validate and store salary
    } // end BasePlusCommissionEmployee2 constructor

    // set weekly salary
    public void setSalary( double salary ) {
        this.salary = ( salary < 0.0 ) ? 0.0 : salary;
    }

    public double getSalary() { return salary; }     // return weekly salary

    // calculate earnings
    public double earnings() {
        // !!! not allowed: rate and sales are private in superclass
        return salary + ( rate * sales );
    }

    // return String representation of BasePlusCommissionEmployee object
    public String toString() {
        // !!! not allowed: attempts to access private superclass members
        return "commission employee: " + name + "\n" +
            "gross sales: " + sales + "\n" +
            "commission rate: " + rate + "\n" +
            "base salary: " + salary;
    } // end method toString
} // end class BasePlusCommissionEmployee2
```

```
public class BasePlusCommissionEmployee3 extends CommissionEmployee {

    private double salary;   // base salary per week

    public BasePlusCommissionEmployee3( String name, double sales,
                                        double rate, double salary ) {
        super( name, sales, rate );
        setSalary( salary ); // validate and store salary
    } // end BasePlusCommissionEmployee3 constructor

    // set weekly salary
    public void setSalary( double salary ) {
        this.salary = ( salary < 0.0 ) ? 0.0 : salary;
    }

    // return weekly salary
    public double getSalary() { return salary; }

    // calculate earnings
    public double earnings() {
        return getSalary() + super.earnings();
    }

    // return String representation of BasePlusCommissionEmployee object
    public String toString() {
        return "base-salaried " + super.toString() + "\n" +
            "base salary: " + salary;
    } // end method toString

} // end class BasePlusCommissionEmployee3
```