



Unit 05

Inheritance



CMPS 251, Fall 2020, Dr. Abdulaziz Al-Ali

Check point (from last Unit)

- ▶ How can we store 100 books in our Store class?
- ▶ Suppose you want to print the titles of the books. How would you do that if?
 - ▶ The bookArr is public ?
 - ▶ The bookArr is private ?

Objectives

- ▶ Introduce the concept of inheritance in Java
- ▶ Define the superclass and subclass
- ▶ Show some basic code examples

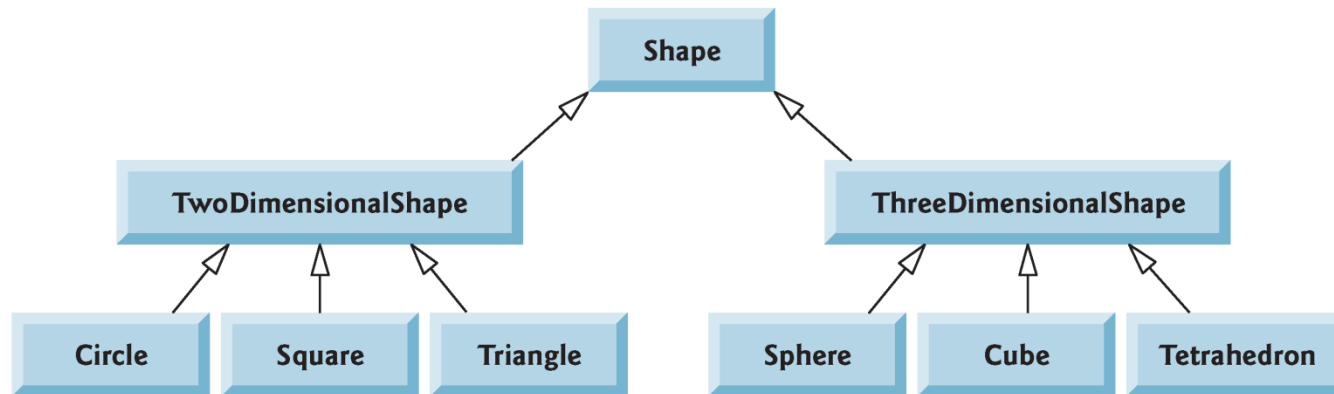
Basic Concepts

- ▶ **Inheritance** is a type of software reuse
- ▶ When creating a new class, you can declare that it should **inherit** all the attributes and methods from another class
- ▶ Existing class is called the **superclass**
- ▶ New class is called the **subclass**

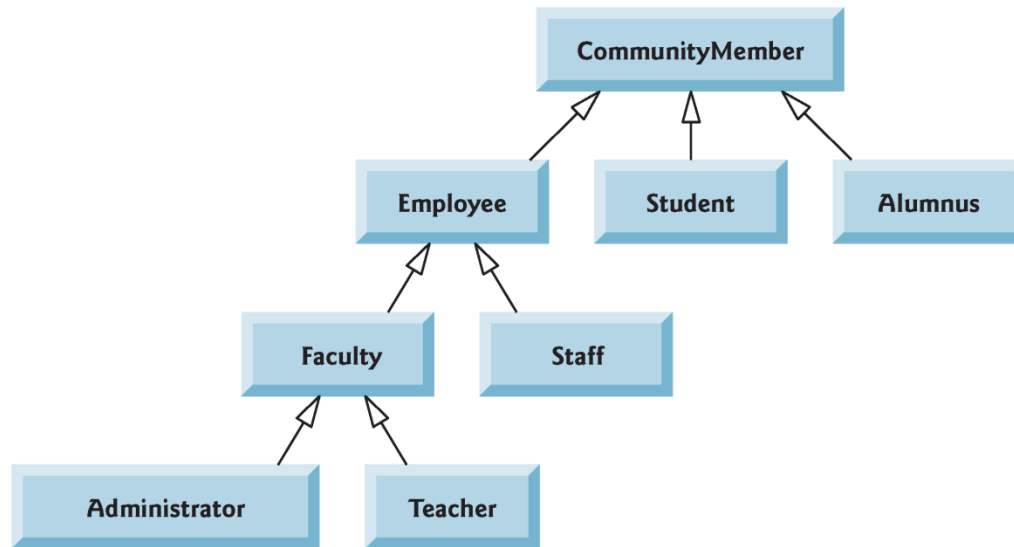
Basic Concepts

- ▶ The subclass inherits all methods and attributes from the superclass
- ▶ The subclass can also define its own attributes and methods
- ▶ A class can only inherit from one superclass
 - ▶ Single inheritance
- ▶ Constructors are not inherited
 - ▶ (But superclass constructors can be called from the subclass)

Basic Graphical Example



Basic Graphical Example



Basic Code Example

Inheritance!



Person.java

```
public class Person {  
    private String name;  
  
    public Person(String name) {  
        this.name = name;  
    }  
}
```

Customer

```
public class Customer extends Person {  
    public String phone;  
  
    public Customer(String n, String p) {  
        super(n);  
        this.phone = p;  
    }  
}
```

Employee.java


```
public class Employee extends Person {  
    public double salary;  
  
    public Employee(String n, double s) {  
        super(n);  
        this.salary = s;  
    }  
}
```


Is-a Relationship

- ▶ Subclasses should have a **is-a** relationship with the superclass
- ▶ A Customer is a Person
- ▶ An Employee is a Person

Constructors and Inheritance

- ▶ Constructors are not inherited
- ▶ Subclass constructors must call a superclass constructor
 - ▶ If you don't, Java will automatically call the superclass's default constructor



What if the default constructor doesn't exist? Try it!

Check point

- ▶ What is Inheritance?
- ▶ What is superclass and subclass?
- ▶ Which word do we use to indicate inheritance in the code?
- ▶ What is inherited in subclasses from super classes?
- ▶ When inside a subclass, can constructors of the superclass be called?

Demo Time!

- ▶ See the *glasses* package in the sample code.
 - ▶ See **TODO** items 1-10

Extra Checkpoint

- ▶ Why did the code complain right after making the MedicalGlasses class extend Glasses?

Method Overriding and Inheritance

- ▶ Often times the subclass should replace a superclass method with its own version
- ▶ Subclass version should have the same signature (name, return type, and arguments) as the superclass
- ▶ It is good practice to use the `@Override` annotation

Method Override Example

Employee.java

```
public class Employee extends Person {  
    public double salary;  
  
    public Employee(String n, double s) {  
        super(n);  
        this.salary = s;  
    }  
  
    public double getPayAmount()  
        return this.salary;  
    }  
}
```

CommissionEmployee.java

```
public class CommissionEmployee extends Employee {  
    double commission;  
    int sales;  
  
    public CommissionEmployee(String n, double salary,  
                               double commission, int sales) {  
        super(n, salary);  
        this.commission = commission;  
        this.sales = sales;  
    }  
  
    @Override  
    public double getPayAmount() {  
        return salary + commission * sales;  
    }  
}
```

More on Overriding

- ▶ You can't override with a more restrictive access modifier
 - ▶ Public method can't become private, for example
- ▶ Private attributes in the superclass can't be directly accessed in the subclass
 - ▶ Use getters and setters
- ▶ Protected and public attributes can be directly accessed

Check point

- ▶ Which of the following is true?
 - ▶ If the parent has a public method, we can override it as protected in the child class
 - ▶ If the parent has a protected method, we can override it as public in the child class

Check point

- ▶ What does method *Overriding* mean? Any rules about overriding?
- ▶ What does method *Overloading* mean?

Demo Time!

- ▶ See the *glasses* package in the sample code.
 - ▶ See `TODO` items 11-22

Using the Subclass as the Superclass

- ▶ Objects of a sub-class can be stored in references to a superclass
 - ▶ This is commonly done in lists and arrays
- ▶ You can check to see if an Object of type superclass is actually a subclass using the **instanceof** operator

Check point

- ▶ Can we override a *private* method from the parent?
(NO, because private methods are not seen by the child)
- ▶ Can we override a *protected* method from the parent and make it *public*?
- ▶ Can we override a *protected* method from the parent and make it *package* access?
- ▶ How can you call the parent version of an overridden method from the child class?

Summary

- ▶ Inheritance is a type of software re-use
- ▶ Attributes and methods are inherited but not constructors
- ▶ Each constructor of the subclass **MUST** call the super class constructor.
- ▶ A class may only inherit directly from one parent
- ▶ An object of the subclass, can be stored in a reference variable of the super class.