Computer Science I
Inheritance
Professor Evan Korth
New York University

Road Map

- Inheritance
  - Class hierarchy
  - Overriding methods
  - Constructors
  - Modifier: protected
  - Keyword: super

Introduction to Inheritance

- Often programmers find themselves designing classes with many similarities.
  - How can we take advantage of these similarities to reduce the amount of code we write?
  - How can we take advantage of these similarities to write code which is less likely to contain errors?
  - How can we take advantage of existing code when we want to design something similar to the existing code?

Introduction to Inheritance

- Inheritance is an object oriented construct which promotes code reuse by allowing you to create a new class using an existing classes as a “starting point”.
  - You simply state which class you want to use as a base in order to inherit its members (fields and methods).
  - You can add or modify its members once you have inherited them.

Terminology

- Superclass (parent class, base class)
  - The original class which has its members inherited
  - Typically, this is the more general class
- Subclass (child class, derived class)
  - The class which inherits members from the superclass
  - In this class, we can specialize the general class by modifying and adding functionality.

Inheritance

- Abstraction
  - Focus on commonalities among objects in system
- “is-a” vs. “has-a”
  - “is-a”
    - Inheritance
    - Subclass object treated as superclass object
    - Example: Car is a vehicle
    - Vehicle properties/behaviors also car properties/behaviors
  - “has-a”
    - Composition
    - Object contains one or more objects of other classes as members
    - Example: Car has a steering wheel

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, buses, bicycles, …
    - subclass: Car
      - Smaller, more-specific subset of vehicles
Superclasses and Subclasses (Cont.)

• Inheritance examples

<table>
<thead>
<tr>
<th>Superclass</th>
<th>Subclasses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student</td>
<td>GraduateStudent, UndergraduateStudent</td>
</tr>
<tr>
<td>Shape</td>
<td>Circle, Triangle, Rectangle</td>
</tr>
<tr>
<td>Loan</td>
<td>CarLoan, HomeImprovementLoan, MortgageLoan</td>
</tr>
<tr>
<td>Employee</td>
<td>Faculty, Staff</td>
</tr>
<tr>
<td>BankAccount</td>
<td>CheckingAccount, SavingsAccount</td>
</tr>
</tbody>
</table>

Inheritance hierarchy for university CommunityMember.

Inheritance hierarchy for Shapes.

Inheritance in Java

• The keyword extends is used in Java for inheritance.
• A class that extends another class is said to be a subclass. It is also known as a child class or a derived class.
• The class it extends is called a superclass. Sometimes the superclass is called a base class or a parent class.
• A subclass can also modify its method members. This is referred to as overriding.
• A subclass inherits the members of its superclass.
• Syntax:
  public class Student extends CommunityMember

Overriding methods

• Often a subclass does not want to have the same exact method implementations as its superclass. In this case, the subclass can override the methods that it wants to change.
• To override a method, you must re-declare the method in the subclass. The new method must have the exact same signature as the superclass’ method you are overriding.
• Only non-private methods can be overridden because private methods are not visible in the subclass.
• Note: You cannot override static methods.
• Note: You cannot override fields.
• In the above two cases, when you attempt to override them, you are really hiding them.
Overload vs Override

- Overloading a method refers to having two methods which share the same name but have different signatures.
- Overriding a method refers to having a new implementation of a method with the same signature in a subclass.

Keyword super

- The keyword super can be used in a subclass to refer to members of the subclass’ superclass. It is used in two ways:
  - To invoke a constructor of the superclass (more about this on the next slide). For example (note: the keyword new is not used):
    - super ();
    - super (params);
  - To call a superclass’ method (it is only necessary to do so when you override (more on overriding soon too) the method). For example:
    - super.superclassMethodName();
    - super.superclassMethodName(params);

Inheritance and constructors

- Constructors do not get inherited to a subclass. Instead, the subclass’ constructor automatically calls it’s superclass’ zero parameter constructor before it executes any of the code in it’s own constructor.
- If you do not want to use the parent’s zero parameter constructor, you must explicitly invoke one of the superclass’ other constructors with the keyword super.
  - If you do this, the call to the superclass’ constructor must be the first line of code in the subclass’ constructor. e.g:
    ```java
    Student (String name) {
      super (?params?);
      ...
    }
    ```
  - If the superclass has no zero parameter constructor, the first line of any subclass constructor MUST explicitly invoke another superclass constructor.

Indirect Inheritance

- It is possible in Java to have an arbitrarily long line of ancestors (i.e. many superclasses on your way to Object)
- The members you inherit from classes above your superclass are said to be indirectly inherited.
- It does not matter where a class’ members come from – a client program always uses similar syntax to access the fields and methods (whether you define it in a class, inherit it directly from your superclass or indirectly from another class in the hierarchy).