Week 9: Topics

- Inheritance

Inheritance in Java

- Mechanism by which a new class is created by acquiring an existing class's members

  Possibly embellishing them with new or modified capabilities

- Saves time during program development

  By basing new classes on existing proven and debugged software

Inheritance in Java (cont.)

- Rather than declaring a completely new class
  - One can designate that the new class should *inherit* the members of an existing class

- Existing class is called

  The superclass

  The new class is called the subclass

- A subclass can add its own fields and methods

  It is more specific than its superclass i.e., represents a more specialized group of objects

Superclasses and Subclasses

- A **direct superclass** is

  One from which a subclass explicitly inherits

- An **indirect superclass** is

  Any class above the direct superclass in the class hierarchy

- In Java the class hierarchy begins with the class `Object` (in package `java.lang`)

  Which every class extends directly or indirectly

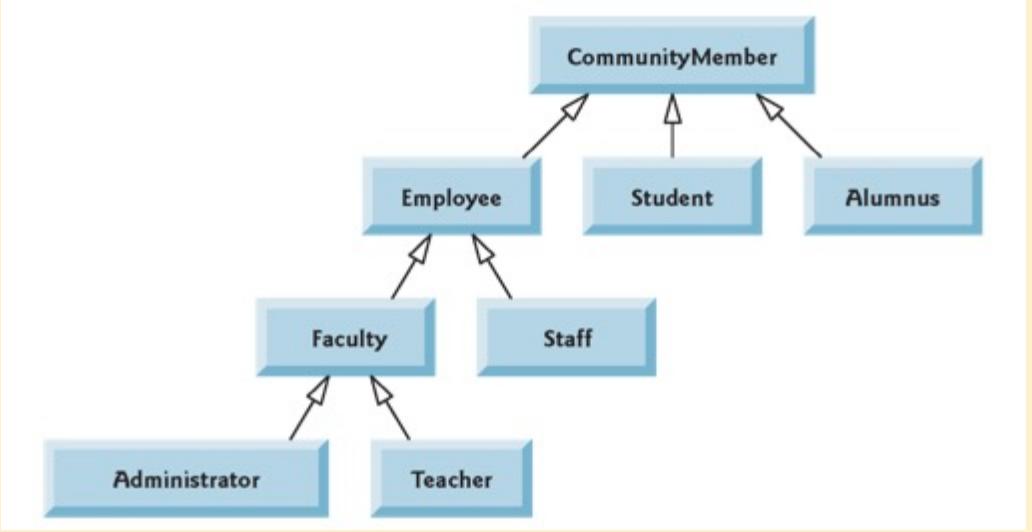
- Java supports only **single inheritance**
  - Each class is derived from exactly one superclass

Superclasses and Subclasses (cont.)

- Inheritance represents a ***is-a* relationship**

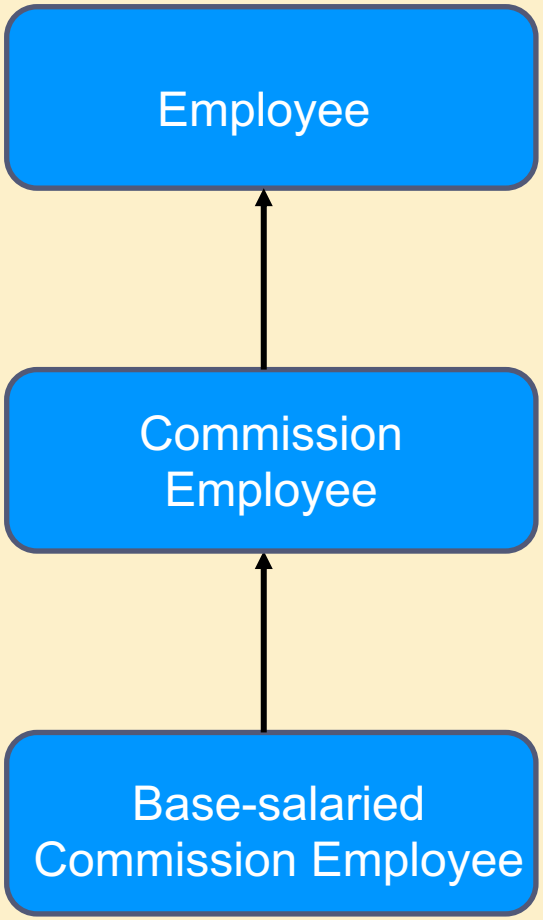
  aka a parent-child relationship

  e.g., a car is a vehicle



- An `Administrator` *is a* `Faculty` (member), *is an* `Employee`, *is a* `CommunityMember` and, of course *is an* `Object`

Relationship Between Superclasses and Subclasses



CommissionEmployee

Blackboard: Week9/CommissionEmployee

Blackboard: Week9/CommissionEmployee

Methods and Instance Variables

- Declaration indicates that class `CommissionEmployee`
  - Extends (i.e., *inherits from*) class `Object` (from package `java.lang`)
- If you do not explicitly specify which class a new class extends
  - The class extends `Object` implicitly

    You will typically not include "extends Object"
- Three instance variables are declared `final`
  - As they do not need to be modified after they are initialized

    We do not provide any corresponding set methods

Constructor

- Constructors are **not** inherited

  Class CommissionEmployee does not inherit class Object's constructor

- Java requires that the **first** task of any subclass constructor

  Is to call its superclass' constructor

- Java implicitly calls the superclass's default or *no-argument* constructor
  - `Object`'s default constructor does nothing

Blackboard: Week9/CommissionEmployee

`toString` Method and `@Override`

- Method `toString` is one of the methods that every class inherits directly or indirectly from class `Object`

  Returns a String representation of an object

- Method `toString` in `CommissionEmployee` overrides class `Object`'s `toString` method

  @Override annotation indicates that the following method declaration should override an existing superclass method

`toString` Method and `@Override`

- To override a superclass method a subclass must declare a method with the same *signature* as the superclass

  Method name, number of parameters, parameter types and order of parameter types

- Object's `toString` method takes no parameters

  CommissionEmployee declares its toString method with no parameters as well

Problem:

Write a test class `CommissionEmployeeTest` that instantiates a `CommissionEmployee` and prints details about the new employee using the `toString` method.

CommissionEmployeeTest

Blackboard: Week9/CommissionEmployee

Problem:

How would one go about extending the `CommissionEmployee` class to create a subclass `BasePlusCommissionEmployee` in which the employee is paid a base salary in addition to their commission?

BasePlusCommissionEmployee V1

Blackboard: Week9/BasePlusCommissionEmployee_V1

Blackboard: Week9/BasePlusCommissionEmployee_V1

BasePlusCommissionEmployee V1

- Note the similarity between this class and class `CommissionEmployee`

  This "copy-and-paste" approach is often error prone and time consuming

- Class `BasePlusCommissionEmployee` does not specify "`extends Object`"

- Class `BasePlusCommissionEmployee`'s constructor

  Invokes class Object's default constructor implicitly

Problem:

Extend the `CommissionEmployee` class to create a subclass `BasePlusCommissionEmployee` using the Java keyword `extends`?

BasePlusCommissionEmployee V2

Blackboard: Week9/BasePlusCommissionEmployee_V2

Inheritance Hierarchy

- Class `BasePlusCommissionEmployee` extends class `CommissionEmployee`

  Only ComissionEmployee's public and protected members are directly accessible by the subclass

- Each subclass constructor must implicitly or explicitly call one of its superclass's constructors

  By using the keyword super followed by the superclass constructor arguments

Blackboard: Week9/BasePlusCommissionEmployee_V2

Inheritance Hierarchy (cont.)

- The compiler generates an error for the call `@Override public double earnings()`

  CommissionEmployee's instance variables commissionRate and grossSalary are private

- The compiler issues additional errors for the `@Override public String toString()` call