Before we begin…

- Lectures
  - Focus on the big picture – the high-level concepts
  - Have some important details, clarifications, and corrections
  - Provide some slides, but will focus on code examples
- Assignments
  - Focus on doing – the application of concepts and details with coding and tools
  - Do not focus on checking if you read the books or watched the lectures (exams will)
- Exams
  - Focus on the books, lectures, and some applications of the material
  - Do not focus on coding, but there may be some on it
- This is the trade-off
  - You have fewer assignments, and they focus on applying, which I know you like
  - But you must study the books and watch the lectures to do well
- Recommendation: Keep a personal study journal (See Course Resources)

Three Characteristics of an Object

State – An object's properties and current values; its internal condition at a given time.

Class members, data members, attributes, fields

Behavior – The actions or operations an object can perform; how it responds to interactions.

Class methods

Sometimes called "implementation"

Identity – A object's unique identifier (class type + internal identifier) that distinguishes it from another.

Sometimes referred to as "type," but type is only one component of it

Sometimes referred to as the "address," but Java does let you see the address

Represented by a hash code

Relationships Types Between Classes

Dependence ("uses-a") - A class uses another class

A class uses or manipulates another class

Example: The Order uses the Account to check a customer's credit status

Aggregation ("has-a") – A class contains another class

A class has an instance of another class

Example: An Employee has an Address

Inheritance ("is-a") – A more specialized version of a general class

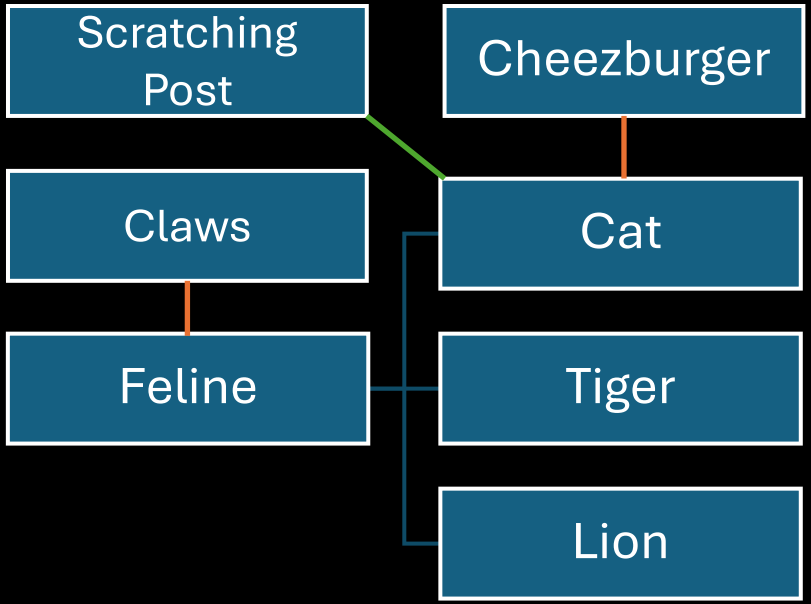
Relationship expressed by a hierarchy

Example: A Student is a Person

Inheritance Hierarchical Relationship

- Expresses the “is-a” relationship
- Parent-child relationship – one parent, many children
- Superclass (base class) and subclass (derived class)
- Allows the implementation of the inheritance concept
- Subclasses gain state, behavior, and identity from the superclass
- Usage depends on requirements
- Should be natural to what you are representing

Relationship Types Among Classes



Types of Inheritance

- Single – A class inherits from one parent class
- Multilevel – A class inherits from a parent class, which itself inherits from another class
- Hierarchical – Multiple classes inherit from the same parent class
- Multiple - A class inherits from more than one parent class
- Hybrid - A combination of two or more types of inheritance

Aspects of Inheritance

Inheritance of State

- Inherits direct access to the public and protected class members
- Inherits package-private access to class members depending on packaging
- Does not inherit private access to class members
- Inherits state initialization behavior (subclass must call a superclass constructor)

Inheritance of Behavior (or Implementation)

- Inherits direct access to the public and protected methods
- Inherits package-private access to methods depending on packaging
- Does not inherit private access to methods
- Does not inherit constructors (subclass must call a superclass constructor)

Inheritance of Type

- Inherits the superclass type
- Inherits the interface type
- Does not inherit the internal identifier

Inheritance of Behavior

Overloading – same name, different signature (name + parameters)

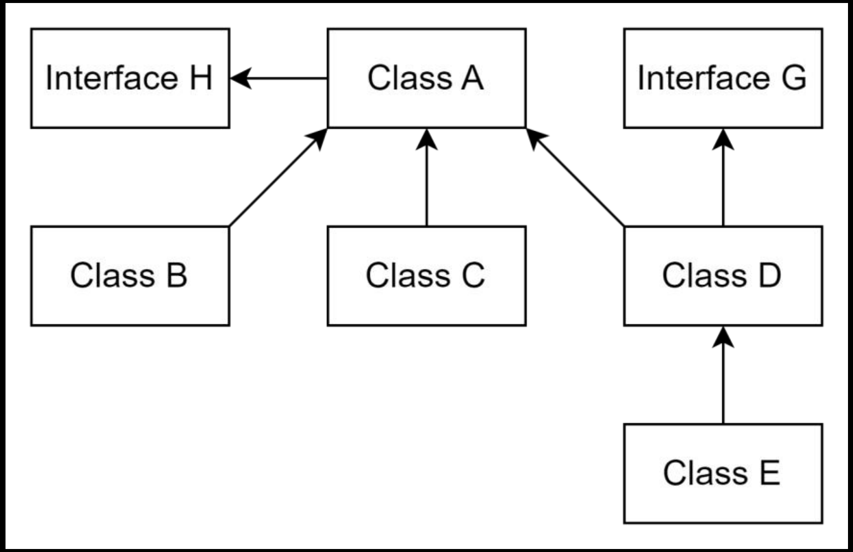
- Occurs only within the same class
- Cannot have two methods with the same signature
- Not involved in inheritance

Overriding

- Subclass method with the same signature as a superclass method
- Calls to the method will invoke the subclass version
- Subclass can call the superclass version using *super*
- Only occurs between inheriting classes
- Overloaded methods in a superclass are inherited
- Can be prevented using final

Polymorphism

- Ability of an object to take on many forms
- An object can pass more than one “is-a” test
- All Java objects are polymorphic
  - Cast up into the superclass type
  - Cast down back into the subclass type
  - Cannot polymorph into sibling types
- Types of polymorphism
  - Compile-time
  - Runtime



Polymorphism Types

Compile-Time

- Static binding or early binding
- The compiler resolves method calls
- Method overloading
- Fast execution because it has already determined the method to execute
- Inheritance not involved

Runtime

- Dynamic binding or late binding
- Calls resolved during execution
- Method overriding
- Slower execution because the determination of methods occurs during execution
- Inheritance involved

Multiple Inheritance

Allowing a class to have more than one superclass

- “Java does not support multiple inheritance.”

Consider the three aspects: state, behavior, and type

• Multiple inheritance of state	statement above is true
• Multiple inheritance of behavior	statement above is true
• Multiple inheritance of type	statement above is false

Multiple Inheritance – The Diamond Problem

Suppose: Class D has method something()

- Class A and Class B inherit something()

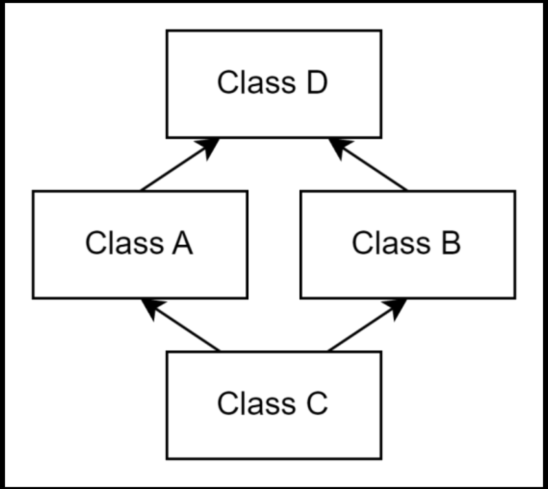
Remember:

- A method has a signature
- Only one method with that signature can be in a class
- Class A and Class B inherit something() – no problem

But which method does Class C inherit?

- Class A? Class B? Class D?

Challenge: Think about how multiple inheritance would affect class members.



Multiple Inheritance

Interfaces are not objects

- There are no constructors
- There are no class members, only static constants
- Abstract methods have no implementation and must be overridden
- Default methods only exist if the class does not define one
- Private interface methods only exist to support default methods
- There is a type, but not an identity, because they are not instantiated

You cannot have two types that are the same

Multiple Inheritance

A class…

- Can “extend” only one other class
- Can “implement” any number of interfaces
- Cannot “extend” an interface
- Cannot “implement” a class

An interface…

- Can “extend” any number of interfaces
- Cannot “implement” an interface
- Cannot “extend” a class
- Cannot “implement” a class