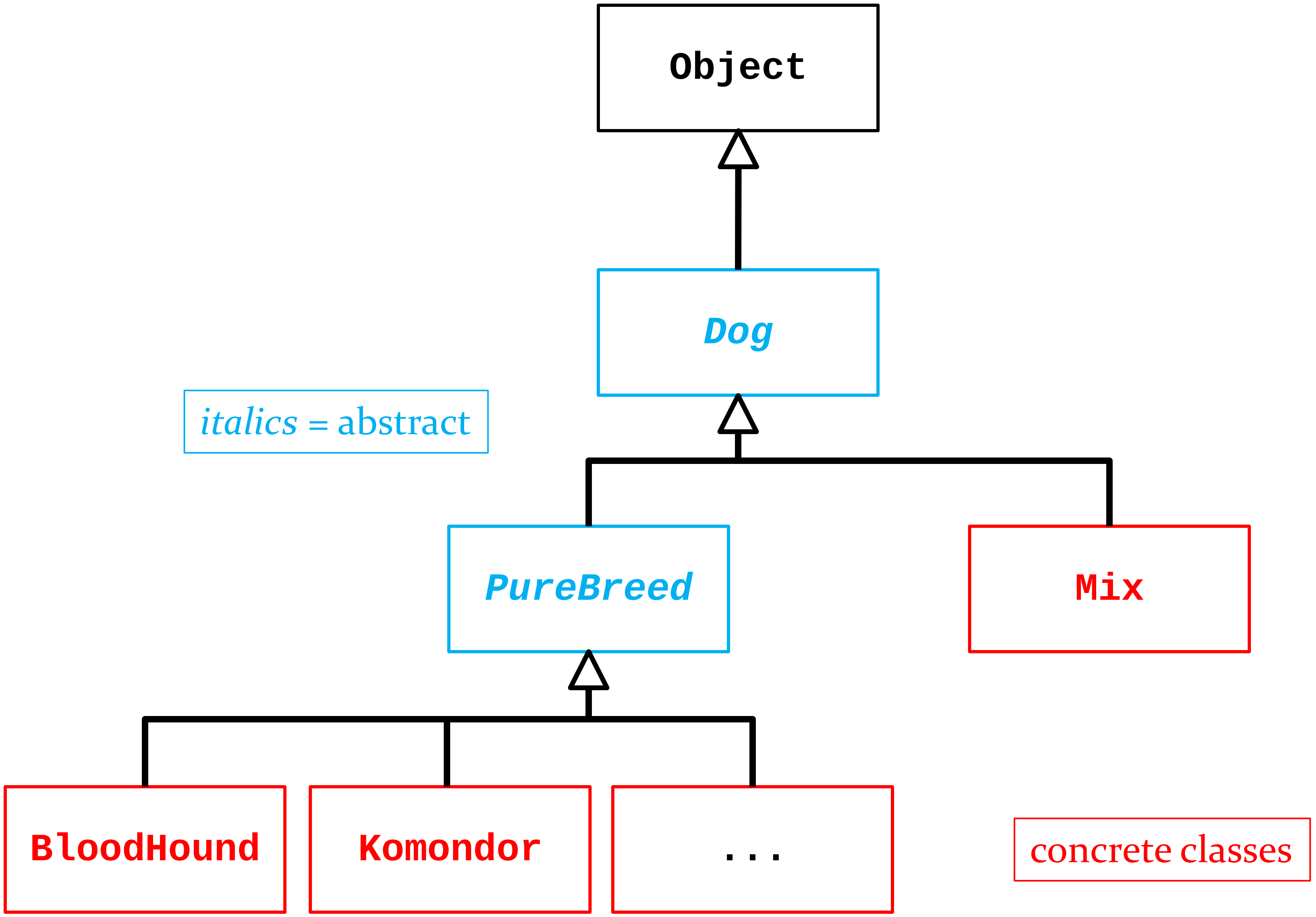
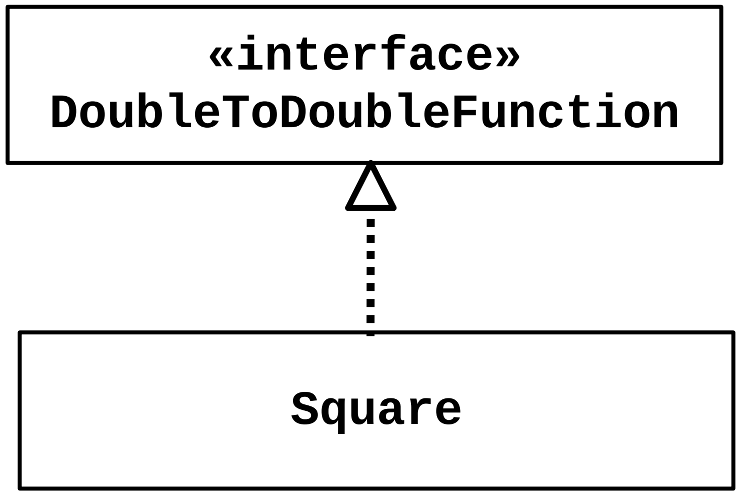
Inheritance

Closing Remarks

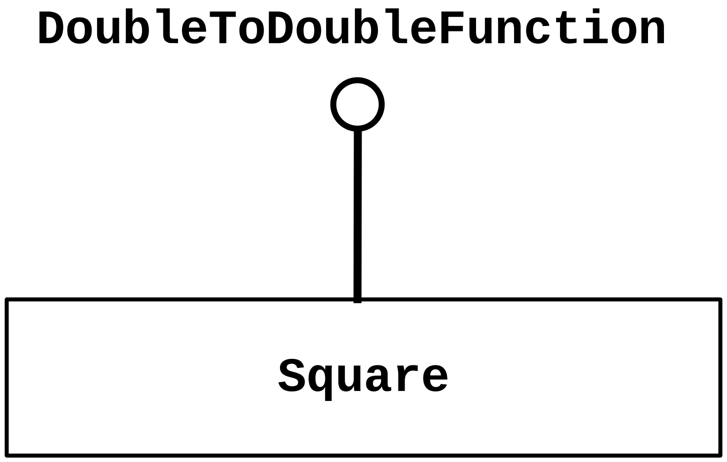

Object
Dog
italics = abstract
PureBreed
Mix
BloodHound
Komondor
...
concrete classes

UML Diagram for Interfaces



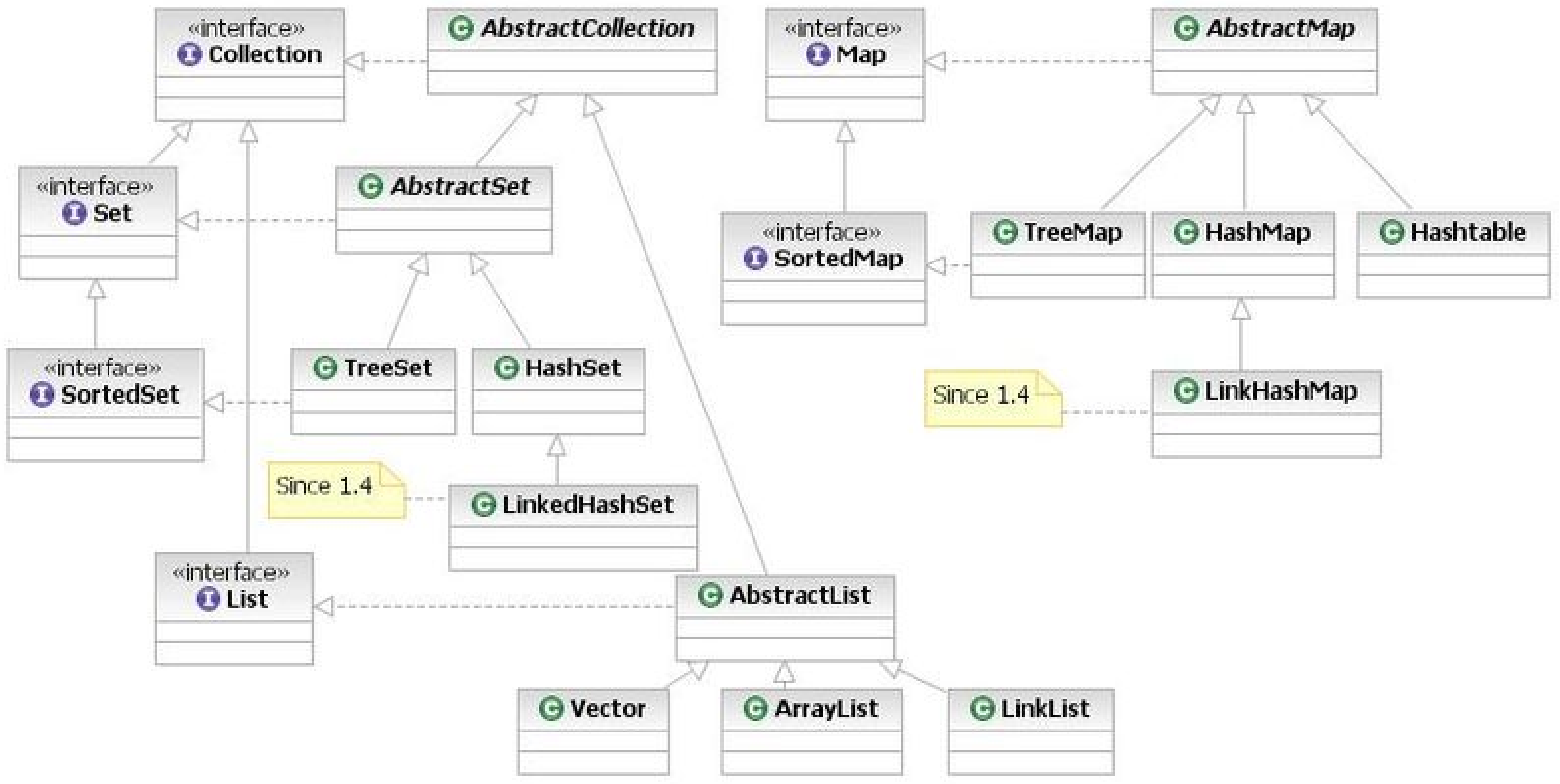
UML Diagram for Interfaces

- alternatively

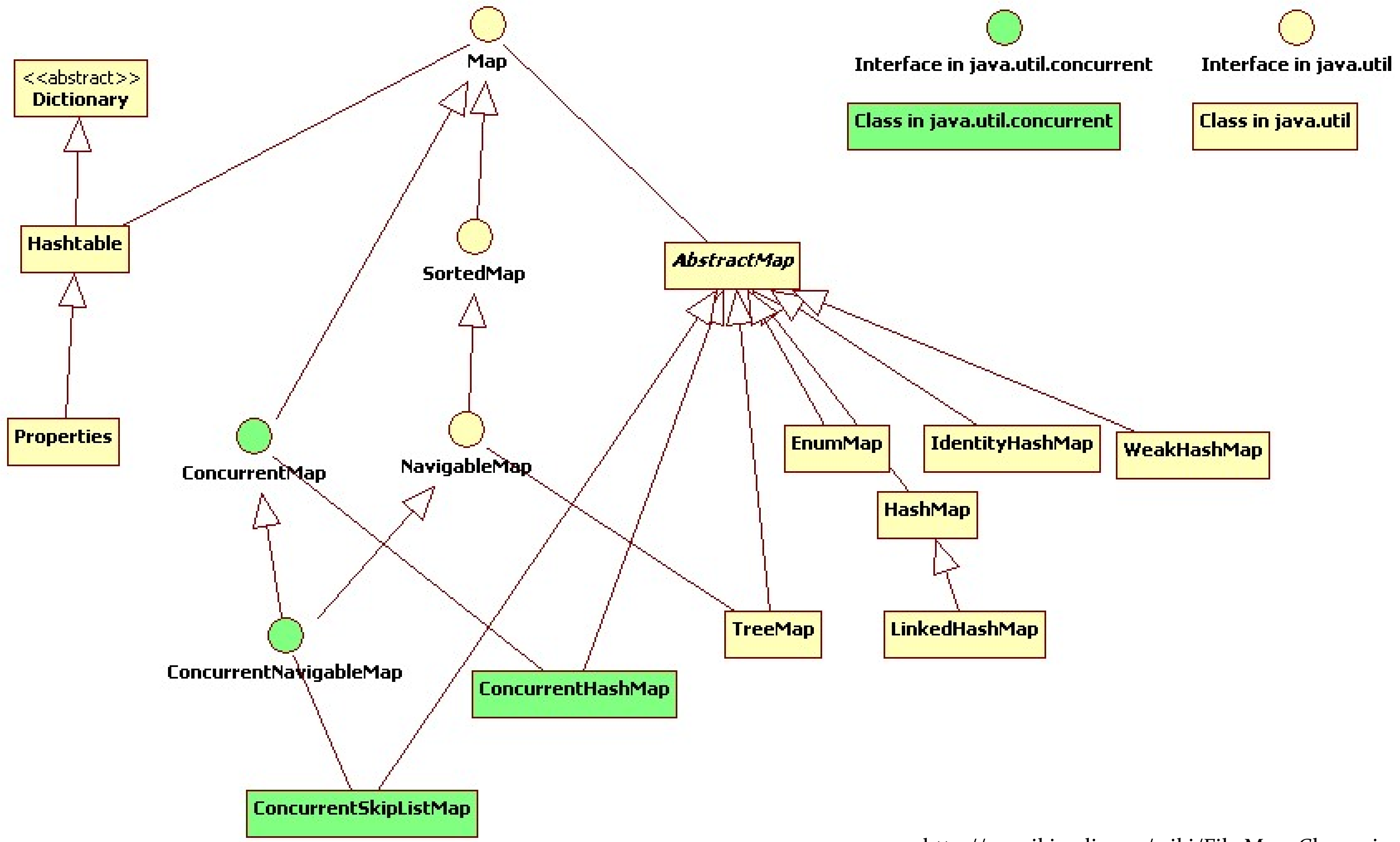


Java Collection Framework

- an old version of the Collection framework



Java Map



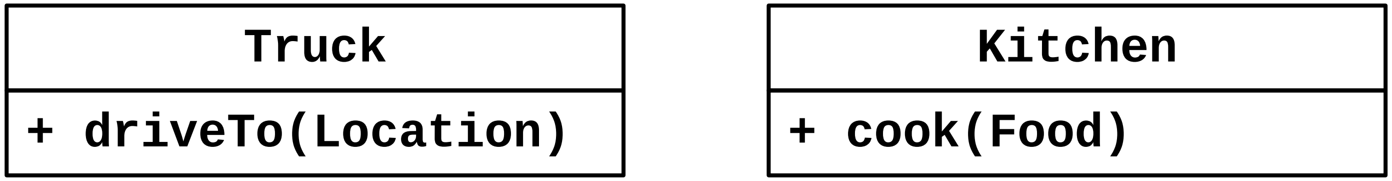
http://en.wikipedia.org/wiki/File:Map_Classes.jpg

Single Inheritance, Multiple Implements

- Java allows for only single inheritance of classes
  - a child class has only one direct parent
- Java allows for multiple implementation of interfaces
  - a class can implement multiple interfaces
  - also, a class can implement an interface through multiple paths
    - e.g., **`TreeMap`** implements **`Map`** through:
      - **`AbstractMap`** (its parent class), and
      - **`NavigableMap`**
    - why is this not a problem?

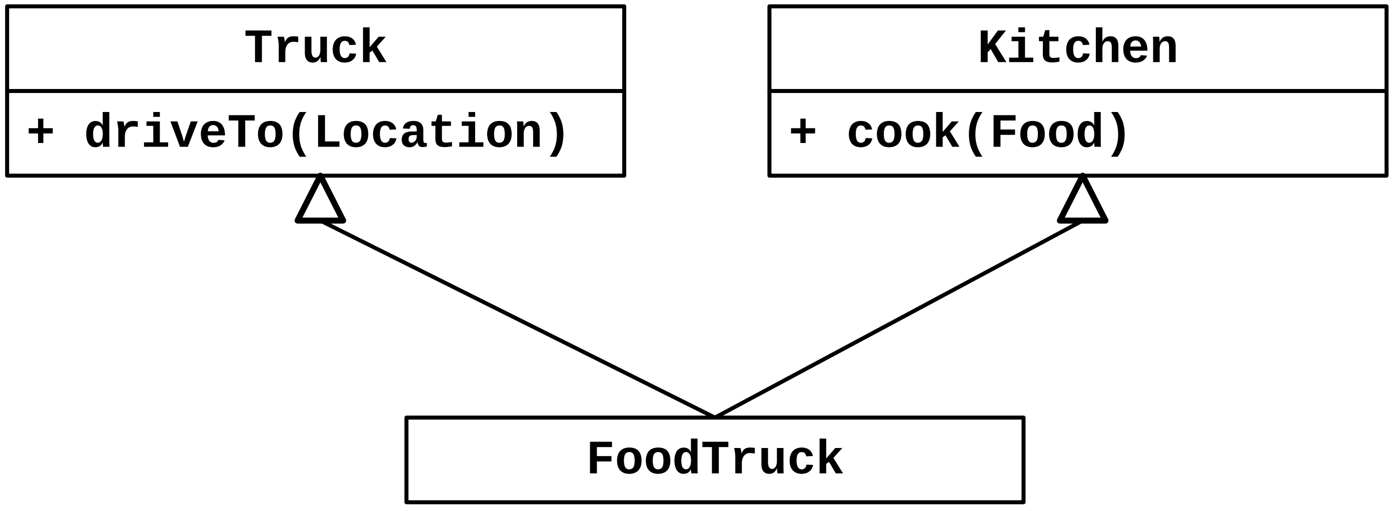
Multiple Inheritance

- some object-oriented languages support multiple inheritance
  - a child class can have more than one parent
- http://stackoverflow.com/questions/3556652/how-do-java-interfaces-simulate-multiple-inheritance
- suppose that you have unrelated **`Truck`** and **`Kitchen`** classes



Multiple Inheritance

- you could implement a **`FoodTruck`** using multiple inheritance

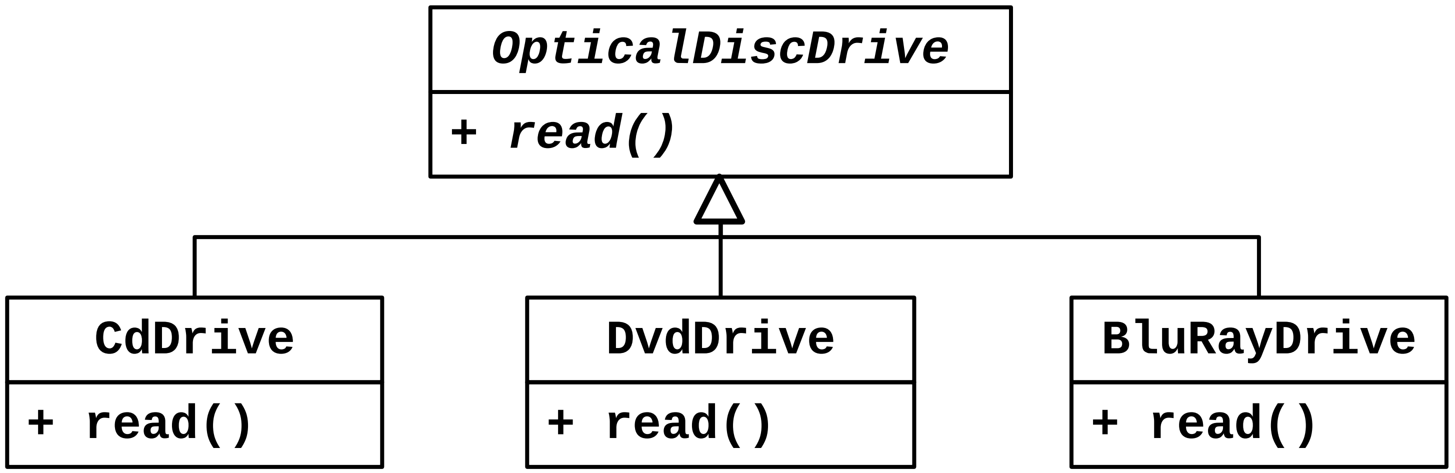


Multiple Inheritance

- a problem that the implementer must deal with when using multiple inheritance is that the same feature might be inherited from different parents

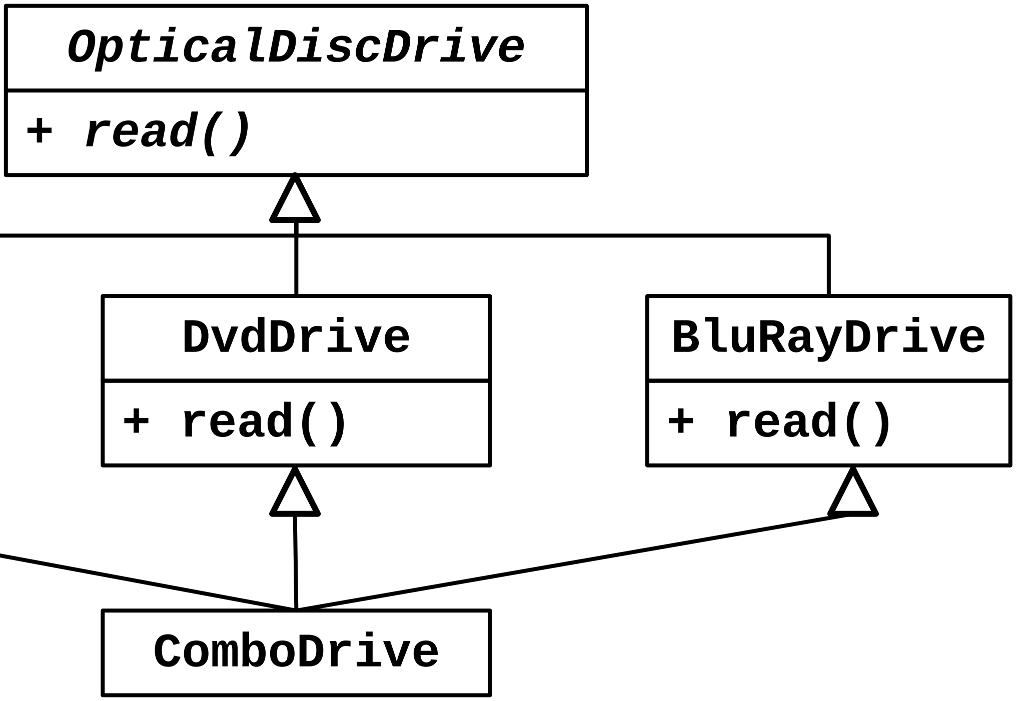
Multiple Inheritance

- suppose that you are designing software to control optical disc drives



Multiple Inheritance

- suppose you implement a **`ComboDrive`**
  - which **`read()`** runs when you invoke **`read()`** on a **`ComboDrive`**?

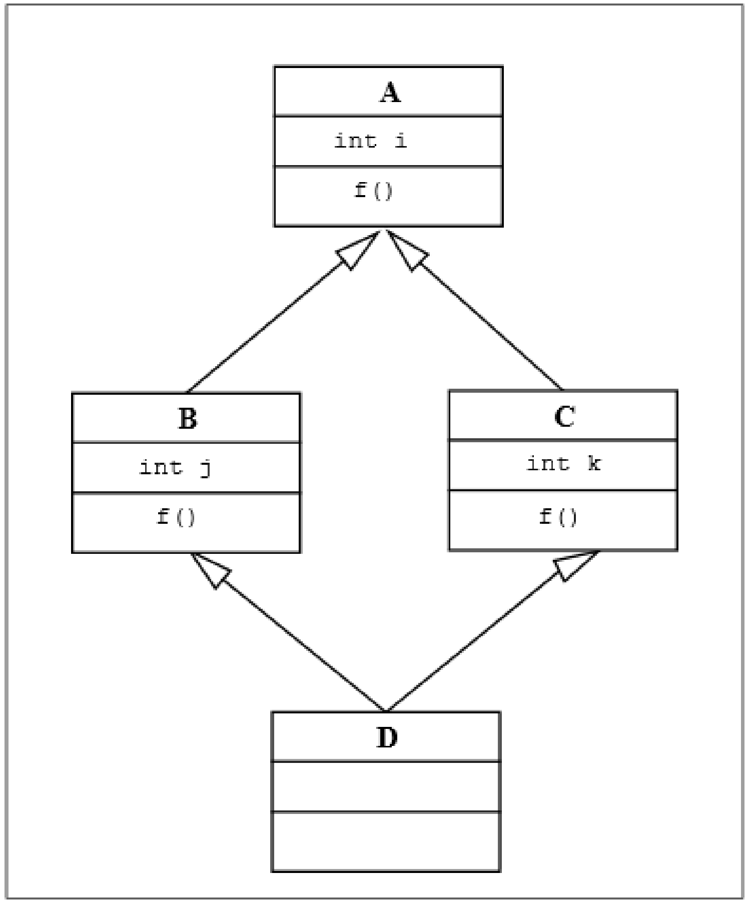


Multiple Inheritance

- common fields inherited from two or more parents are also problematic

- does **`D`** inherit two copies of **`i`**? Or **`B`**'s copy of **`i`**? Or **`C`**'s copy of **`i`**?

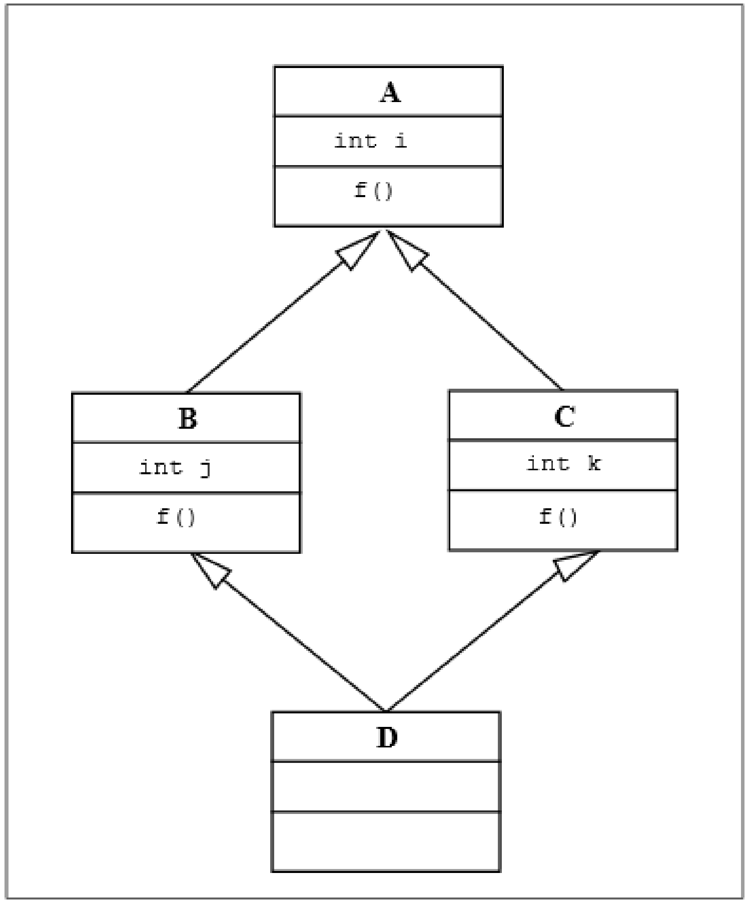
Figure 1: Deadly Diamond of Death



Multiple Inheritance

- multiple inheritance would also complicate Java's object model
- does **D** have two **A** subobjects? Or **B**'s **A** subobject? Or **C**'s **A** subobject?

Figure 1: Deadly Diamond of Death



http://objectmentor.com/resources/articles/javacpp.pdf

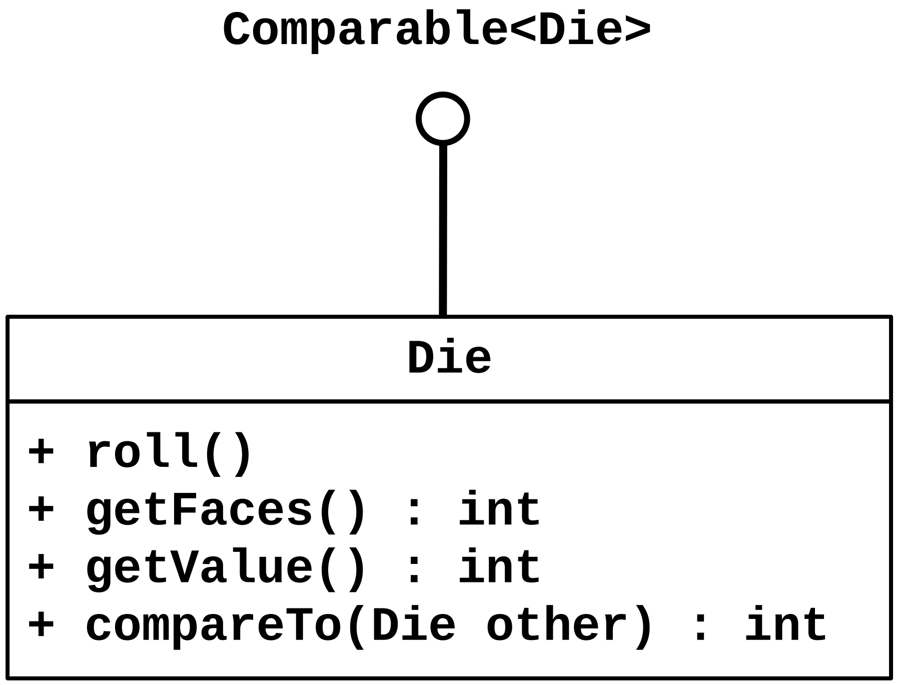
Exercises for the Student

- how could you implement **`FoodTruck`** in Java?
- how could you implement **`ComboDrive`** in Java?

Abusing Inheritance

- inheritance allows a child class to reuse fields and methods from its parent classes
  - i.e., it is a mechanism for code reuse
- it can be very tempting to use inheritance to reuse code from a class that does something similar to what you want your class to do
  - e.g., consider implementing **`BoggleDie`** by extending **`Die`**

UML Diagram for Die



Abusing Inheritance

- another example:
  - suppose that you need a list of integers that is always in sorted order
    - i.e., whenever you add an integer to the list, the list reorders its elements so that the list is in sorted order
  - let's try extending **`ArrayList<Integer>`**