Object Orientation Second Story

Bok, Jong Soon
javaexpert@nate.com
https://github.com/swacademy/JavaSE

What is a Inheritance?

- **Inheritance specifies an "is a kind of" relationship**
  - Inheritance is a class relationship
  - New classes specialize existing classes

Generalization

Musician — Super class

Specialization
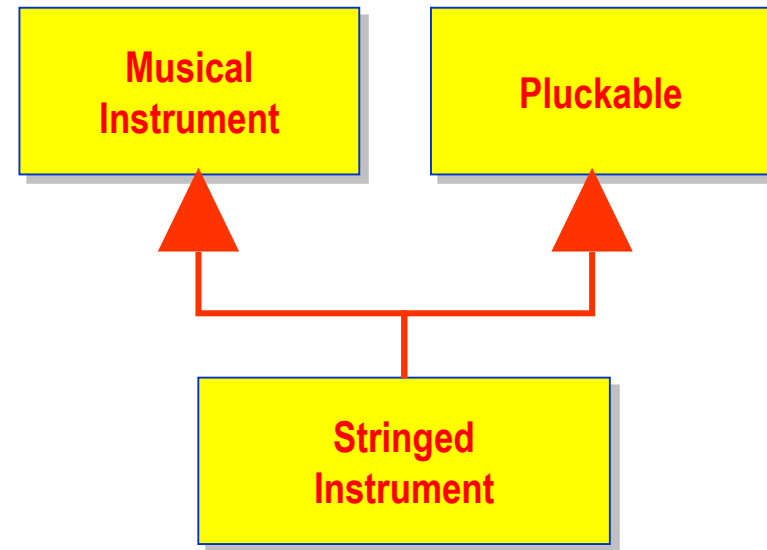
Violin Player — Sub class
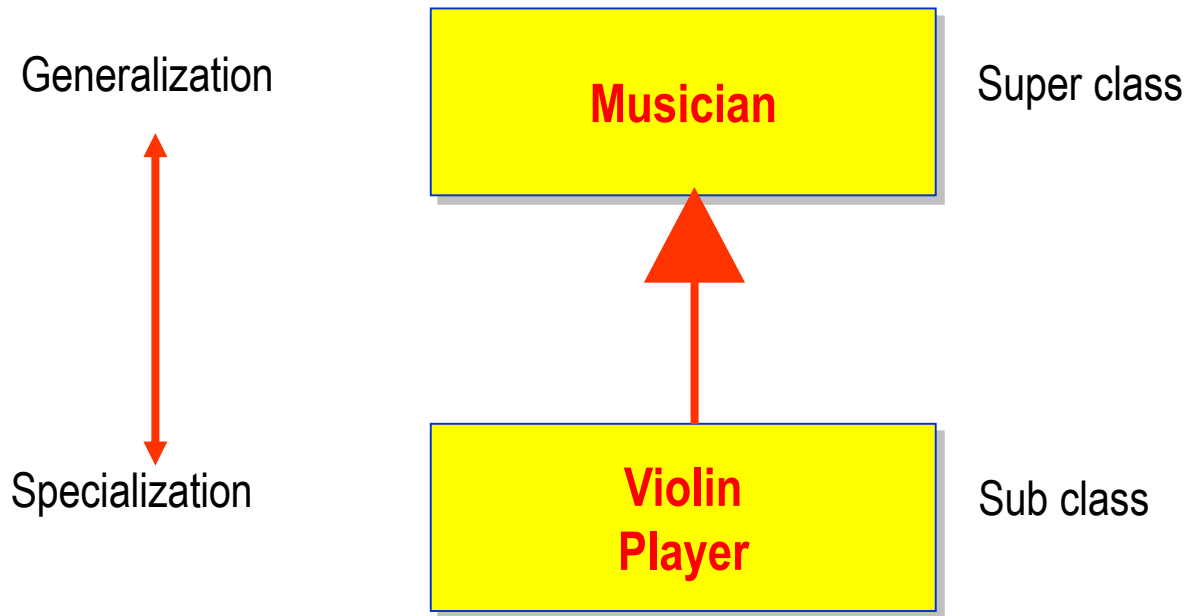
Is this a good example of inheritance ?

Single and Multiple Inheritance

- Single inheritance: extending from one super class
- Multiple inheritance: extending from two or more super classes
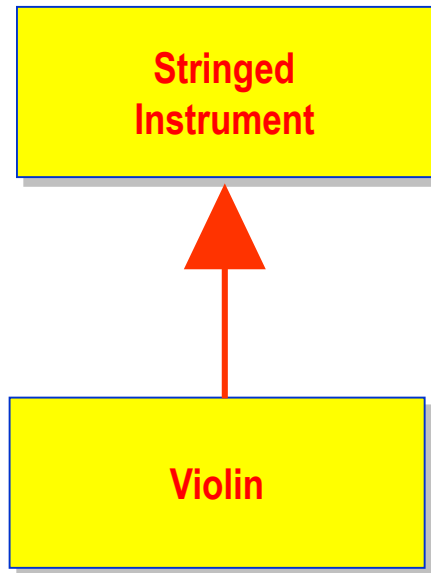
Violin has a single direct super class

Stringed Instrument has two direct super classes

Subclassing

The **Employee** class:

```
public class Employee {
    private String name;
    private double salary;
    private Date dateOfBirth;

    public String getDetails(){ … }
}
```

Employee

-name: String
-salary: double
-dateOfBirth: Date

+getDetails(): String

Subclassing (Cont.)

The **Manager** class:

```
public class Manager {
    private String name;
    private double salary;
    private Date dateOfBirth;
    private String department;

    public String getDetails() {…}
}
```
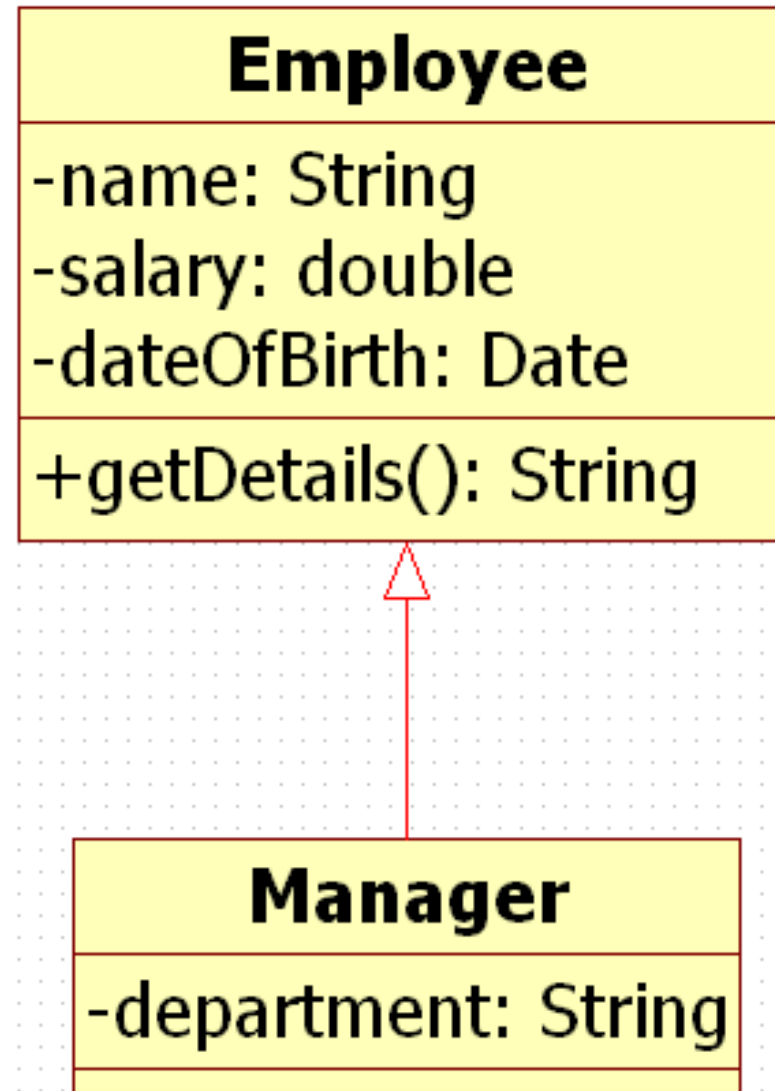

Manager
-name: String
-salary: double
-dateOfBirth: Date
-department: String
+getDetails(): String

Subclassing (Cont.)

```java
public class Employee  {
    public String name;

    public double salary;

    public Date dateOfBirth;

    public String getDetails() { … }

}


public class Manager extends Employee {
    public String department;

}
```

Subclassing (Cont.)

Inheritance

- Inheritance is the OO term referring to grouping classes together based on common theme or common attributes.

- Lets common members be defined in one class and shared by other classes

- Class inherited from superclass or parent class

- Class that inherits subclass or child class

- Use the keyword `extends`.

Single Inheritance

- When a class inherits from only one class, it is called *single inheritance*.

- Single inheritance makes code more reliable.

- **interface**s provide the benefits of multiple inheritance without drawbacks.

- Syntax of a Java class:

```
[modifier] class class_name [extends
                    <superclass> ] {
        …
}
```

The *is a* Relationship

- A class can inherit from only one superclass at a time.
- Use the *is a* phrase to determine if a proposed inheritance link is valid.
  - "A Manager object *is an* Employee."

The *is a* Relationship (Cont.)

- Check the *is a* relationship of the following code:

```
class Cycle {
        int numberOfWheels;
        int numberOfSeats;
        float luggageCapacity;
          //and so on
}


class Boeing747 extends Cycle {
        int numberOfWings;
             //and so on
}
```

# The *is a* Relationship (Cont.)
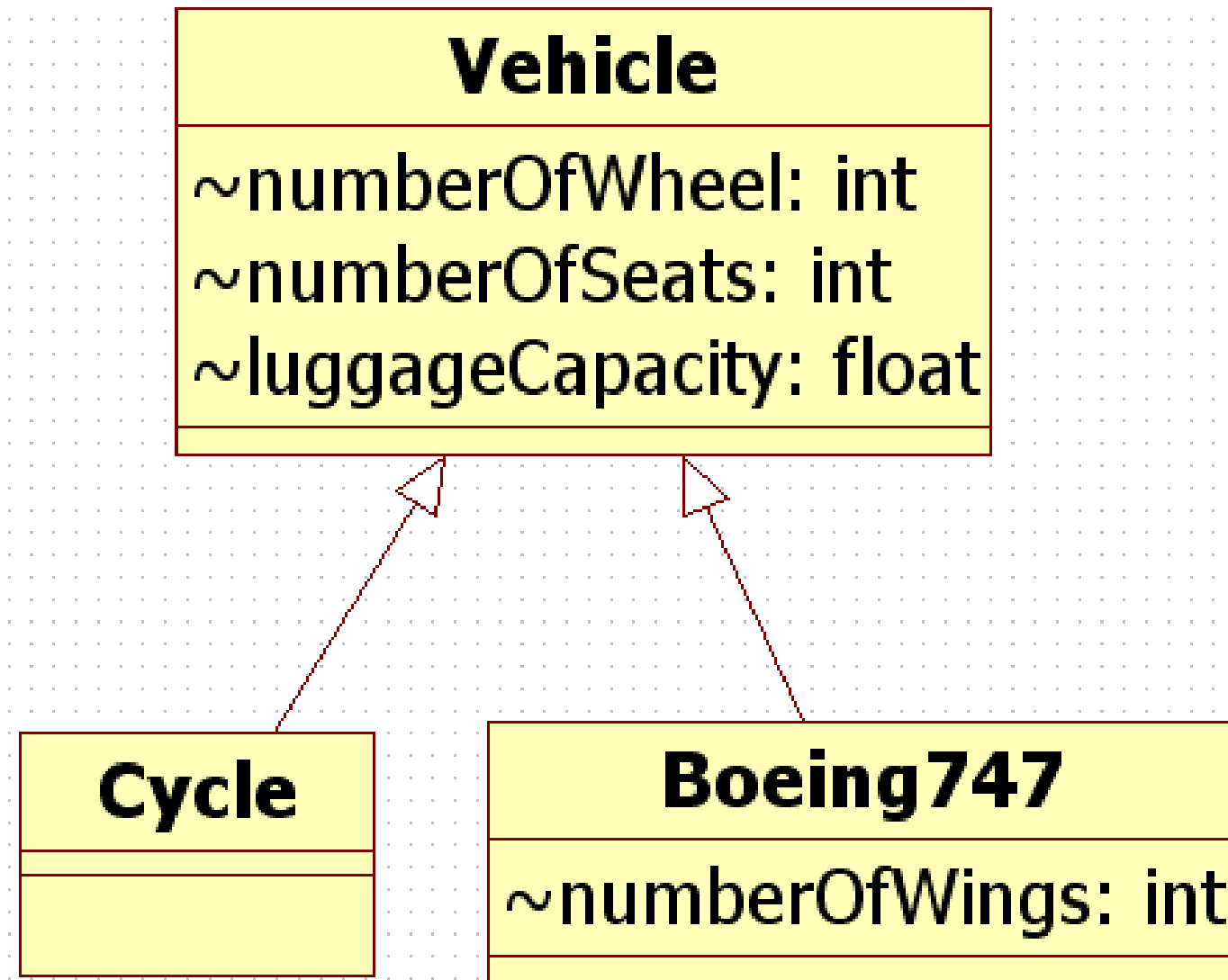
# Containment

- Write a class that contains a reference to other classes.
- Objects have to be instantiated separately, but the overall effect is syntactically and realistically improved.

```
class Cooker {
        //whatever the class does
}

class Refrigerator {
        //whatever the class does
}

class Kitchen {
        Cooker myCooker;
        Refrigerator myRefrigerator;
        //and so on
}
```

The *has a* Relationship

- Validate containment relationships with the *has a* phrase.
  - "My Kitchen *has a* Cooker."
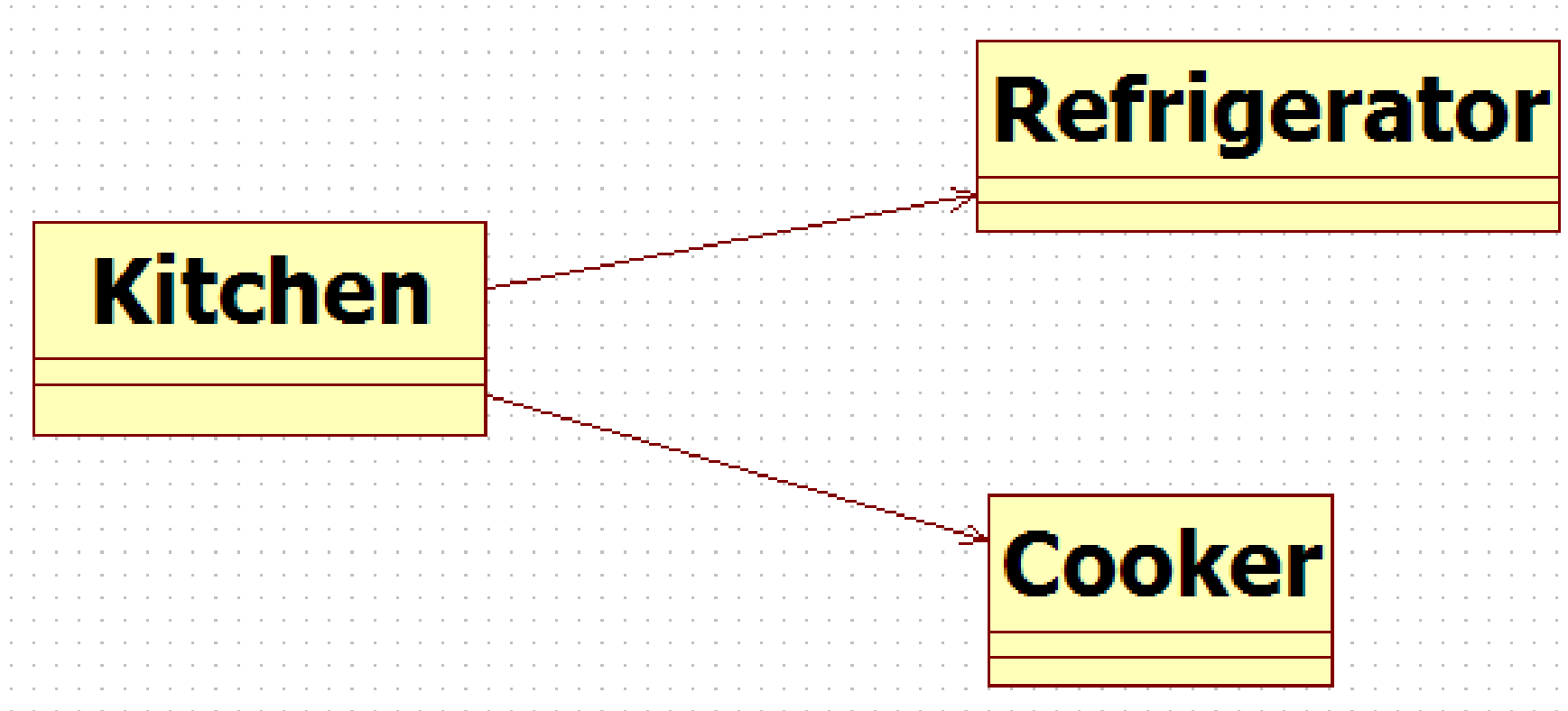
Constructors Are Not Inherited

- A subclass inherits all methods and variables from the superclass(parent class).

- A subclass does not inherit the constructor from the superclass.

- Two ways to include a constructor are:
  - Use the default constructor.
  - Write one or more explicit constructors.

The `super` Keyword

- `super` is used in a class to refer to its superclass.

- `super` is used to refer to the member of superclass, both data attributes and methods.

- Behavior invoked does not have to be in the superclass ; it can be further up in the hierarchy.

Invoking Parent Class Constructors

- In many circumstances, the default constructor is used to initialize the parent object.
- If used, you must place **super** or **this** in the first line of the constructor.

```
public class Employee {
        String name;
        public Employee(String name) {
                this.name = name ;
        }
}
public class Manager extends Employee{
        String department;
        public Manager(String s, String d){
                super(s);
                department = d;
        }
}
```