

Basics of Inheritance



Chapter 5 – Section 3



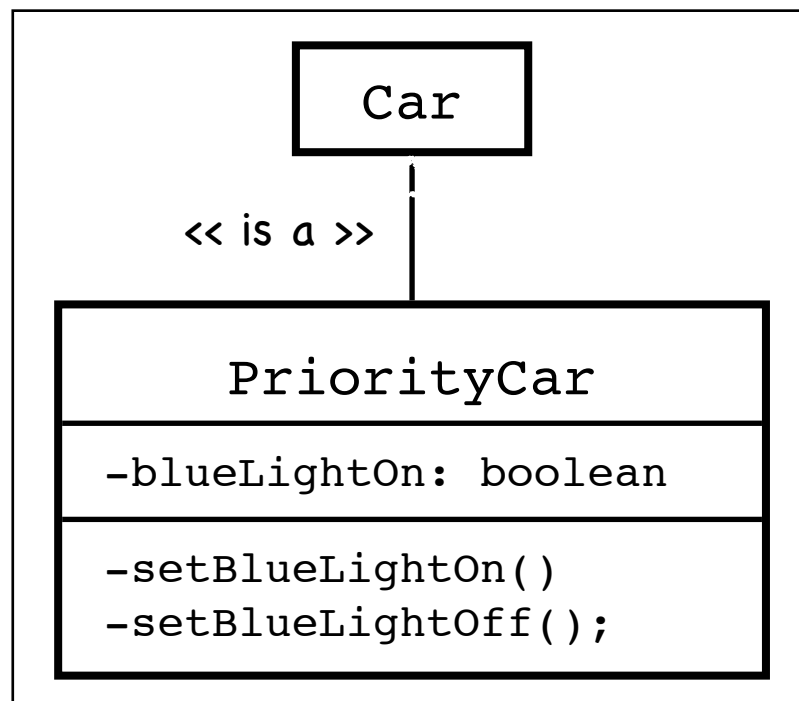
Table of contents

- ☉ Inheritance: the "IS A" relationship
- ☉ The special case of constructors
- ☉ The three roots of Inheritance
 - ☉ Extension
 - ☉ Overriding
 - ☉ Overloading



The three roots of inheritance

Extension: adding a member



```
public class PriorityCar
    extends Car {
    private boolean blueLightOn;

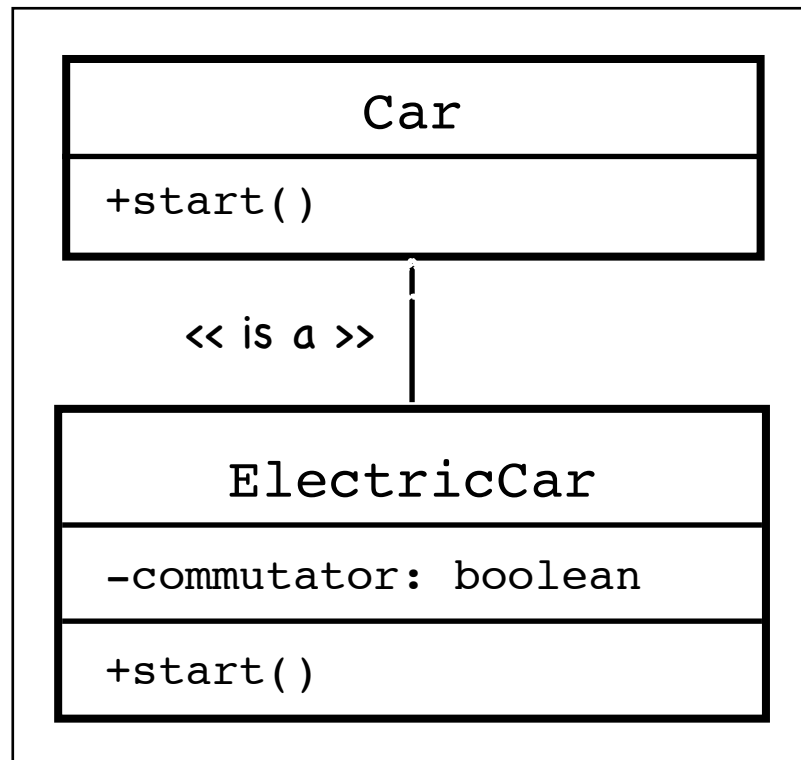
    public void setBlueLightOn() {
        - do something with hardware
        blueLightOn = true;
    }

    public void setBlueLightOff() {
        - do something with hardware
        blueLightOn = false;
    }
}
```



The three roots of inheritance

● Overriding: modifying a member



```
public class ElectricCar
    extends Car {
    private boolean commutator;

    public void start() {
        - do something with hardware
        commutator = false;
    }
}
```

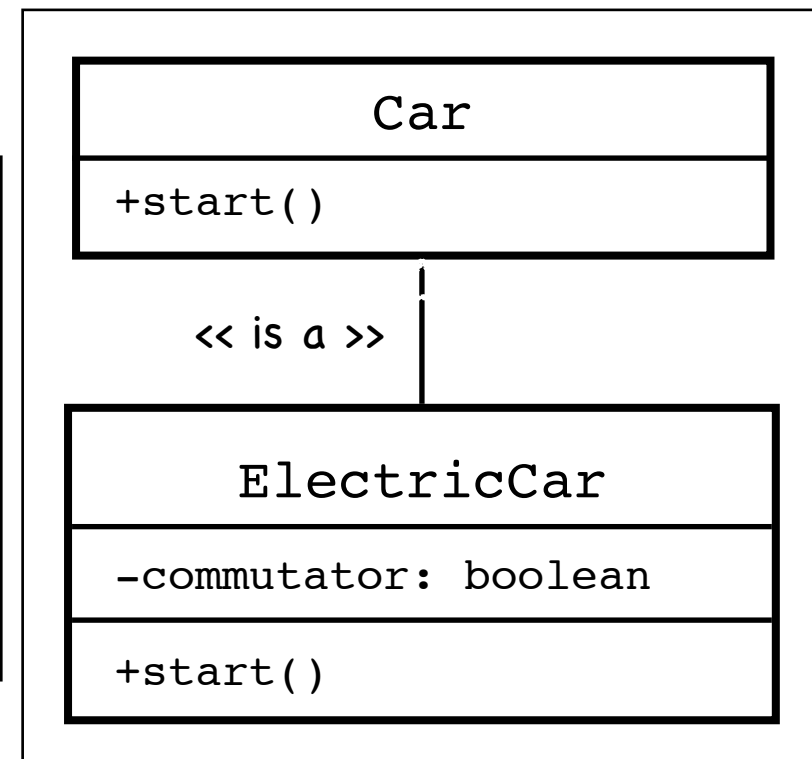


The three roots of inheritance

● Overriding: modifying a member, with reuse

```
public class ElectricCar
    extends Car {
    private boolean commutator;

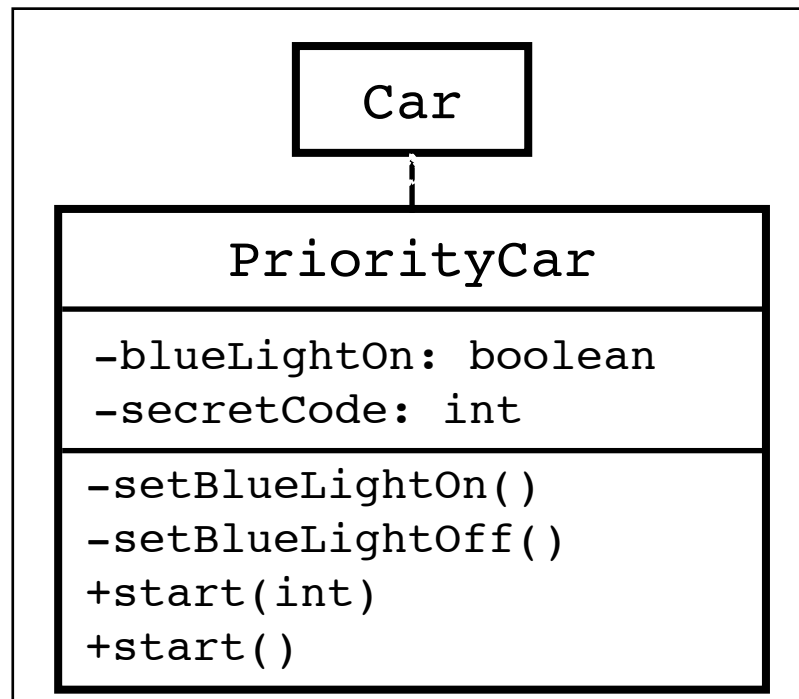
    public void start() {
        super.start();
        commutator = false;
    }
}
```





The three roots of inheritance

Overloading: duplicating a member



```
public class PriorityCar
    extends Car {
    private boolean secretCode;
    ...
    public void start (int code){
        if (code==secretCode)
            super.start;
    }
    public void start() {
        System.exit(1); // not allowed
    }
}
```

Warning: mistakes...



How to prevent inheritance ?

☉ If we do not want it possible: final

```
public class Car {  
    ...  
    public final void start() {  
        isStarted = true;  
    }  
    ...  
}
```

```
public class PriorityCar  
    extends Car {
```





```
    ...  
    public void start() {  
        ...  
    }  
    ...  
}
```

start() in PriorityCar cannot override start() in Car;
overridden method is final



Conclusion (partial)

Interests of inheritance

-  Programming by extension
-  Specialization
-  Overriding
-  Reuse