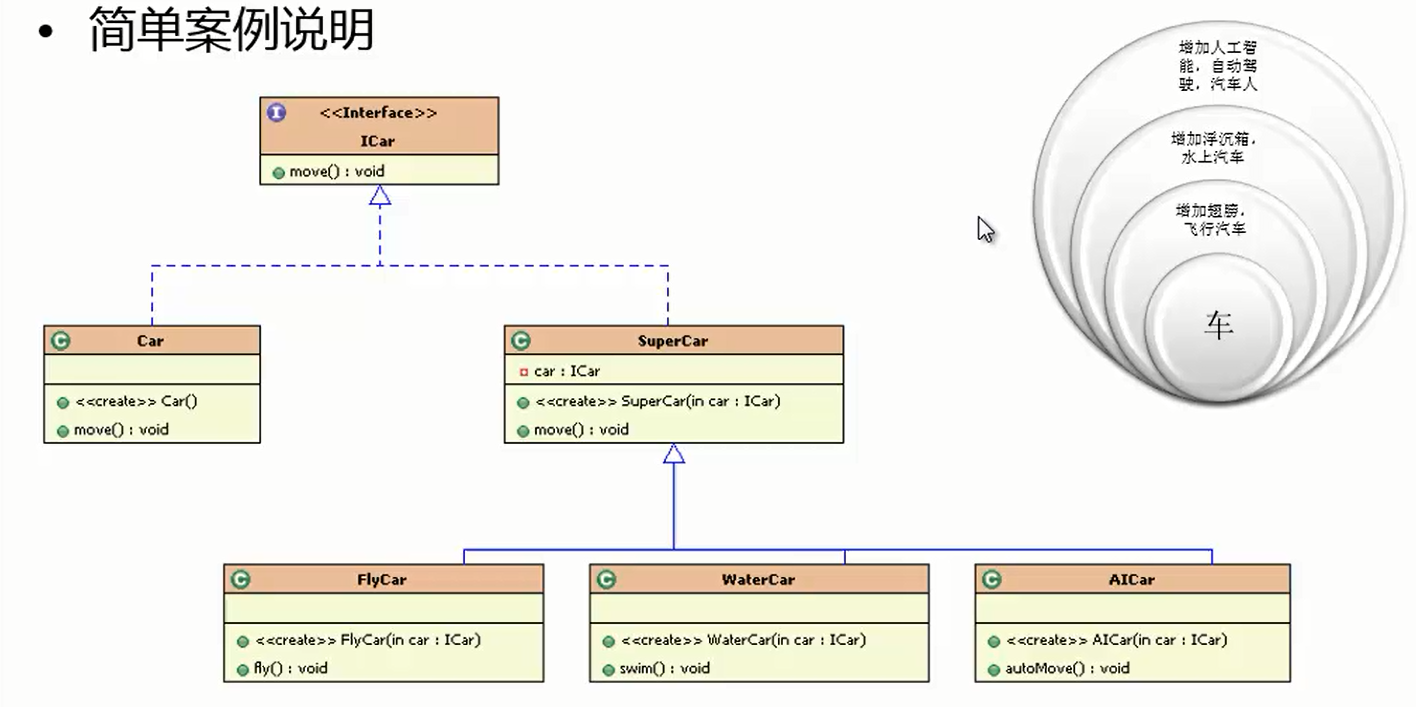
我曾经以为应该用继承处理一切。后来领教到运行时扩展，远比编译时期的继承威力大。  
**本章可以称为“给爱用继承的人一个全新的设计眼界”。**我们即将再度探讨典型的继承滥用问题。

**在本章将会学到如何使用对象组合的方式，做到在运行时装饰类。一旦你熟悉了装饰的技巧，你讲能在不修改任何底层代码的情况下，给你的（或别人的）对象赋予新的职责。**  
 装饰器模式是一种用于代替继承的技术,无需通过继承增加子类就能扩展对象的新功能。

由于业务的扩展产生很多的子类，所以我们可以通过修饰模式，从而进行组合扩展。

代码实现如下：

**package** decorater;  
  
**public interface** ICar {  
 **void** move();  
}

**package** decorater;  
  
**public class** Car **implements** ICar {  
  
 @Override  
 **public void** move() {  
 System.***out***.println(**"陆地上跑"**);  
 }  
}

**package** decorater;  
  
**public class** SuperCar **implements** ICar {  
  
 **protected** ICar **car**;  
  
 **public** SuperCar(ICar car) {  
 **this**.**car** = car;  
 }  
  
 @Override  
 **public void** move() {  
 **car**.move();  
 }  
}

**package** decorater.superCar;  
  
**import** decorater.ICar;  
**import** decorater.SuperCar;  
  
**public class** AICar **extends** SuperCar {  
  
 **public** AICar(ICar car) {  
 **super**(car);  
 }  
  
 @Override  
 **public void** move() {  
 **super**.move();  
 System.***out***.println(**"智能遥控"**);  
 }  
}

**package** decorater.superCar;  
  
**import** decorater.ICar;  
**import** decorater.SuperCar;  
  
**public class** FlyCar **extends** SuperCar{  
  
 **public** FlyCar(ICar car) {  
 **super**(car);  
 }  
  
 @Override  
 **public void** move() {  
 **super**.move();  
 System.***out***.println(**"天上飞"**);  
 }  
}

**package** decorater.superCar;  
  
**import** decorater.ICar;  
**import** decorater.SuperCar;  
  
**public class** WaterCar **extends** SuperCar {  
  
 **public** WaterCar(ICar car) {  
 **super**(car);  
 }  
  
 @Override  
 **public void** move() {  
 **super**.move();  
 System.***out***.println(**"水上游"**);  
 }  
}

**package** decorater;  
  
**import** decorater.superCar.AICar;  
**import** decorater.superCar.FlyCar;  
**import** decorater.superCar.WaterCar;  
  
**public class** test {  
 **public static void** main(String[] args) {  
 ICar car = **new** Car();  
 car.move();  
 System.***out***.println(**"+++++++++++普通车++++++++++++++"**);  
 WaterCar waterCar = **new** WaterCar(car);  
 waterCar.move();  
 System.***out***.println(**"+++++++++++水上游的超跑++++++++++++++"**);  
 AICar aiCar = **new** AICar(waterCar);  
 aiCar.move();  
 System.***out***.println(**"+++++++++++人工智能的水上游超跑++++++++++++++"**);  
 SuperCar superCar = **new** FlyCar(aiCar);  
 superCar.move();  
 System.***out***.println(**"+++++++++++集齐所有功能的超跑++++++++++++++"**);  
 }  
}

结果：

陆地上跑

+++++++++++普通车++++++++++++++

陆地上跑

水上游

+++++++++++水上游的超跑++++++++++++++

陆地上跑

水上游

智能遥控

+++++++++++人工智能的水上游超跑++++++++++++++

陆地上跑

水上游

智能遥控

天上飞

+++++++++++集齐所有功能的超跑++++++++++++++