* + - * 1. 继承的概述

在继承关系中，“子类就是一个父类”，也就是说，子类可以被当作父类看带。

例如：父类是员工，子类是讲师，那么“讲师就是一个员工”。关系：is-a；

*/\*  
定义一个父类格式：  
public class 父类名称{  
 // ...  
}  
  
定义一个子类的格式：  
public class 子类名称 extends 父类名称 {  
 // ...  
}  
 \*/*

父类：  
**public class** Fu {  
 **public void** method(){  
 System.***out***.println(10);  
 }  
}

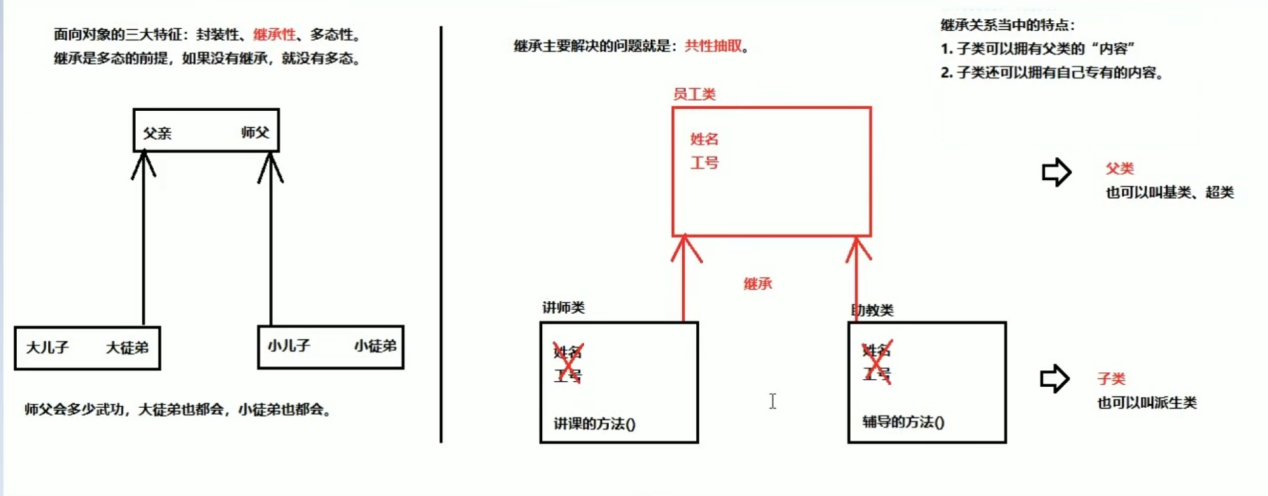
子类：

**public class** Zi **extends** Fu{

} // 子类中什么也没写，它可以继承父类的内容。

主函数：

**public class** zhu {  
 **public static void** main(String[] args) {  
 *// 创建一个子类的对象* Zi zi = **new** Zi();  
 *// Zi类当中什么也没写，但是会继承父类的method方法。* zi.method(); *// 10;* }  
}



* + - * 1. 继承的格式和访问特点

*/\*  
在父类的继承关系当中，如果成员变量重名，则创建子类对象时，访问有两种方式。  
  
直接通过子类对象访问成员变量：  
 等号左边是谁，就优先用谁，没有则找向上找。  
间接通过成员方法访问成员变量：  
 该方法属于谁，就优先用谁，没有则向上找。  
 \*/*

*【重点】：继承当成一般由子类、父类、主函数组成。*

父类：

**public class** Fu {  
 **int numFu** = 10;  
 **int num** = 100;  
 **public void** methodFu(){  
 *// 使用的是本类当中的，不会向下找子类的内容。* System.***out***.println(**num**);  
 }  
}

子类：

**public class** Zi **extends** Fu{  
 **int numZi** = 20;  
 **int num** = 200;  
 **public void** methodZi(){  
 System.***out***.println(**num**);  
 }  
}

主函数：

**public class** zhu {  
 **public static void** main(String[] args) {  
 *// 创建父类对象* Fu fu = **new** Fu();  
 System.***out***.println(fu.**numFu**); *// 只能使用父类的东西，没有任何子类的内容  
 // 创建子类对象* Zi zi = **new** Zi();  
 System.***out***.println(zi.**numFu**); *// 10;* System.***out***.println(zi.**numZi**); *// 20;  
 // 子类可以同时使用，可以使用父类和子类的东西。  
 // 等号左边是谁，就优先用谁。* System.***out***.println(zi.**num**); *// 优先子类，200  
 // System.out.println(zi.abc); 到处都没有，编译报错。  
 // 这个方法是子类的，优先用子类的，没有在向上找* zi.methodZi(); *// 200  
 // 这个方法是在父类当中定义* zi.methodFu(); *// 100* }  
}

* + - * 1. 区分子类方法中重点的三种

*/\*  
局部变量： 直接写成成员变量名  
本类的成员变量： this.成员变量名  
父类的成员变量： super.成员变量名  
 \*/*

在父子类的继承关系当中，创建子类对象，访问成员方法的规则：

创建的对象是谁，就优先用谁，如果没有则向上找

注意事项：

无论是成员方法还是成员变量，如果没有都是向上找父类，绝对不会向下找子类。

父类：

**public class** Fu {  
 **int num** = 10;  
  
}

子类：

**public class** Zi **extends** Fu{  
 **int num** = 20;  
 **public void** method(){  
 **int** num = 30;

// 使用不同的前缀调用名，可以产生不同的地方的内容  
 System.***out***.println(num); *// 30 局部变量* System.***out***.println(**this**.**num**); *// 20 本类的成员变量* System.***out***.println(**super**.**num**); *// 10 父类的成员变量* }  
}

主函数：

**public class** zhu {  
 **public static void** main(String[] args) {  
 Zi zi = **new** Zi();  
  
 zi.method(); *// 30 20 10* }  
}

* + - * 1. 继承方法中的覆盖重写概念

*重写（Override)  
概念：在继承关系当中，方法的名称一样，参数列表也一样。  
重写（Override）：方法的名称一样，参数列表【也一样】。覆盖、覆写。  
重载（Override）：方法的名称一样，参数列表【不一样】。  
方法的覆盖特点：创建的是子类的对象，则优先用子类方法。*

*方法覆盖的注意事项：  
  
1. 必须保证父子之间方法的名称相同，参数列表也相同。  
@Override:写在方法面前，用来检查是不是有效的正确覆盖重写。  
这个注释就算不写，只要满足要求，也是正常的方法覆盖重写。  
  
2. 子类方法的返回值必须【小于等于】父类方法的返回值范围。  
小扩展提示：java.lang.Object类是所有类的公共最高父类（祖宗类），java.lang.String就是object的子类。  
  
3. 子类方法的权限必须【大于等于】父类的权限修饰符。  
小扩展提示：public > protected > (default) > private  
备注：（dafault)不是关键字default，而是什么都不写，留空。*

旧代码：

*// 本来的老手机***public class** phone {  
 **public void** call(){  
 System.***out***.println(**"打电话"**);  
 }  
 **public void** send(){  
 System.***out***.println(**"发短信"**);  
 }  
 **public void** show(){  
 System.***out***.println(**"显示号码"**);  
 }  
}

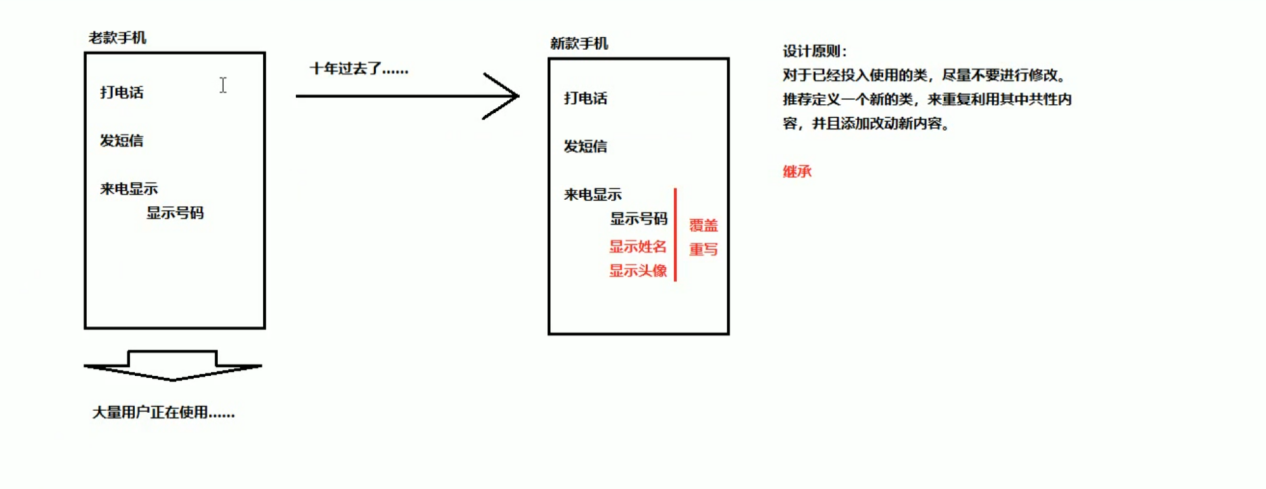
新代码：

**public class** Newphone **extends** phone{  
  
 @Override  
 **public void** show(){  
 **super**.show(); *// 把父类的show方法拿过来重复利用  
 // 自己子类再添加更多内容* System.***out***.println(**"显示姓名"**);  
 System.***out***.println(**"显示头像"**);  
 }  
}

主函数：

**public class** zhuphone {  
 **public static void** main(String[] args) {  
 phone oppo = **new** phone();  
 oppo.call(); *// 打电话* oppo.send(); *// 发短信* oppo.show(); *// 显示号码* Newphone vivo = **new** Newphone();  
 vivo.call(); *// 打电话* vivo.send(); *// 发信息* vivo.show(); *// 显示号码 显示姓名 显示头像。*

// 其中show方法继承旧版代码中的show的内容。



* + - * 1. super关键字的三种用法

super关键字的三种用法：

1. 在子类的成员方法中，访问父亲的成员变量。
2. 在子类的成员方法中，访问父亲的成员方法。
3. 在子类的构造方法中，访问父类的构造方法。

super关键字用来访问父亲内容，而this关键字用来访问本类的内容。

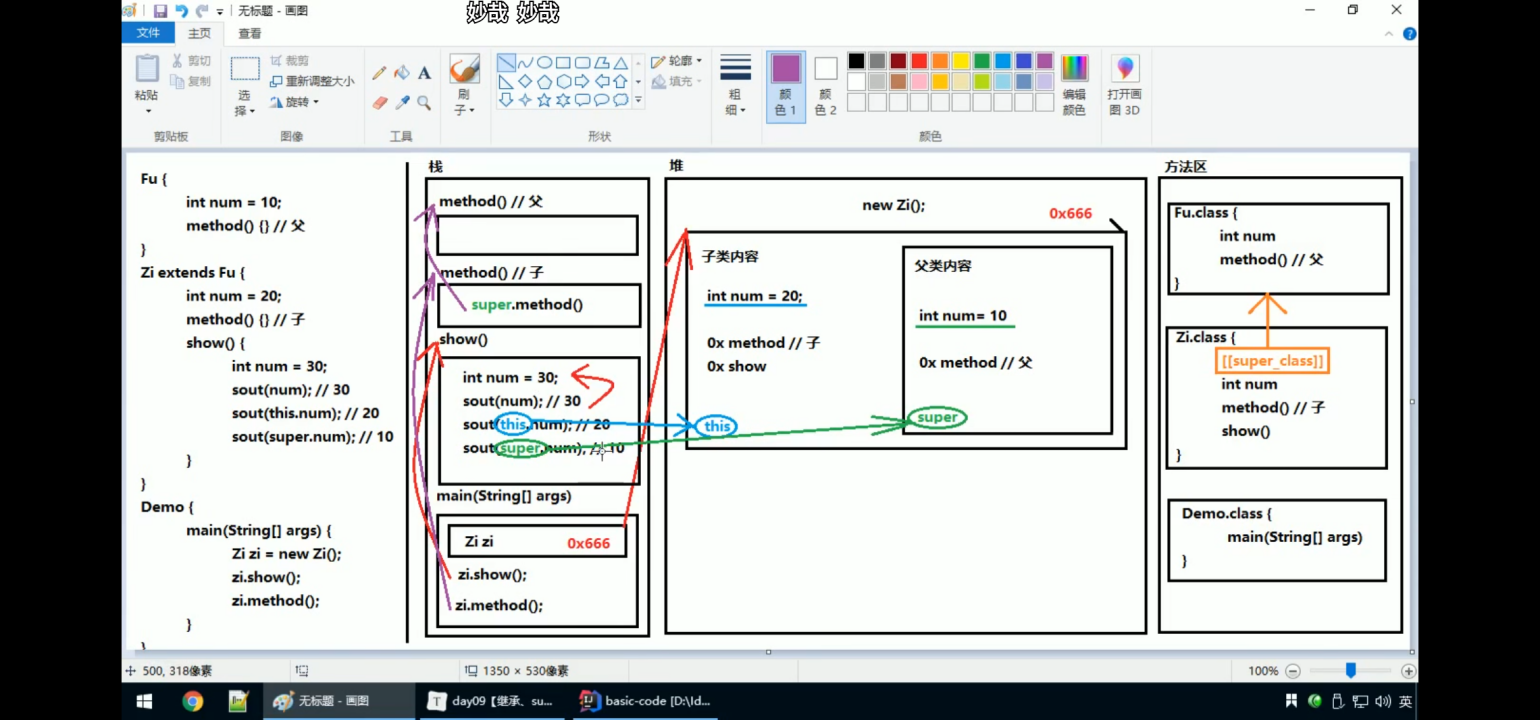
this关键字的三种用法：

1. 在本类的成员方法中，访问本类的成员变量。
2. 在本类的成员方法中，访问本类的另一个成员方法。
3. 在本类的构造方法中，访问本类的另一个构造方法。

在第三种用法当中要注意：

1. this(...)调用也必须是构造方法的第一个语句，唯一一个。
2. super和this两种构造调用，不能同时使用。

this与super关键字图解



* + - * 1. Java继承的三个特点

