·字类实例化流程

范例:

class Person{

public Person(){

System.out.println("【实例化Person对象】");

}

}

class Student{

public Student(){

System.out.println("【实例化Student对象】");

}

}

public class JavaDemo{

public static void main(String args[]){

new Student();

//输出结果为:

【实例化Person对象】

【实例化Student对象】

}

分析上述程序可以知道，当实例化Students时会自动使用 supper() 语句 也就是说会自动调用父类无参构造方法

也只会默认调用无参构造方法，且super() 方法只能放在 字类构造方法的第一行

super() ;表示的就是子类调用父类的构造方法，所以这里写于不写是一样的

但是如果父类没有无参构造那么在这里就必须使用 super()明确调用有参构造

class Person{

public Person(String name){

System.out.println("【实例化Person对象】");

}

}

class Student{

public Student(String name ){

super(name);

System.out.println("【实例化Student对象】");

}

}

public class JavaDemo{

public static void main(String args[]){

new Student();

//输出结果为:

【实例化Person对象】

【实例化Student对象】

}

结论: 无论如何，在实例化字类对象的时候一定会实例化父类对象，目的时为所有的属性都可以进行空间分配

super和this都可以进行构造方法的调用，并且都要放在首行， super调用父类 this调用本类

所以super 和 this 不能同时出现

·继承的相关限制

java之中不允许多重继承，只允许多层继承

范例:(错误)

class A{}

class B{}

class C extends A,B{} //多重继承

有些语言可以，多重继承的目的是为了继承多个类之中的方法和属性

(正确)

class A{}

class B extends A{}

class C extends B{} //多层继承

实际开发中不应该超过三层， 但理论是可以的，因为在超过三层后，会出现逻辑不清晰等问题

子类继承父类实际上可以继承父类的所有操作结构，但对于私有操作属于隐式继承，但对于非私有操作都是显示继承

隐式继承无法直接访问，但可以通过public方法间接访问