**class** Fu{

Fu(){

System.***out***.println("fu run ...");

show();

}

**public** **void** show() {

// **TODO** Auto-generated method stub

System.***out***.println("fu run again...");

}

}

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

**int** num=8;

Zi(){

//super();

System.***out***.println("zi run...."+num);

}

**public** **void** show(){

System.***out***.println("zi run...."+num);

}

}

**public** **class** Extends {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Zi z=**new** Zi();

z.show();

}

}

**运行结果：**



问题1，父类子类构造函数调用的顺序？

从运行结果可以看出，构造函数的调用是先从父类开始，如果父类也是继承grandfather，那么就先调用grandfather的构造函数。这里为了使代码更加清晰的指出问题2，没有演示grandfather 类及构造函数的调用。出现这种顺序的原因是构造函数的第一行有super（）语句，所以一进入构造函数就跳转至父类的构造函数。C++的顺序与Java一致，，c++的析构函数的执行顺序是反过来的，这里没有演示。

原因：子类继承了父类，获取了父类中的内容，在获取父类属性之前，需要知道父类是如何初始化属性的。所以在构造对象时先运行父类构造函数。

问题2既然是先执行父类的构造函数，那么为什么调用子类的show方法？而且调用子类的

方法却又不能打印num属性？

简单说一下内存中的情况。首先main（）方法进栈，然后变量z进栈，由于Zi继承Fu，所以Fu类中的方法构造函数及show（）方法先进方法区，同样的Zi类中的构造函数及show（）方法也进入方法区。接着new Zi（）;进入堆，于是执行Zi（）的构造函数，一进入函数体遇到super（）；又跳转至父类的构造函数，于是fun run....打印了。接着执行show（），由于父类中的this是z的引用，所以就在z的方法中找show（）并执行。那么为什么不是打印zi run....8呢？因为Zi（）中的num属性只是做了默认的赋0,特定的赋值是在super（）执行完之后执行的。所以super（）成了分水岭，于是super（）后面的打印语句打印出zi run...8,z.show()也打印出了zi run...8。