Java中关于子类实例化和多态调用的经典示例

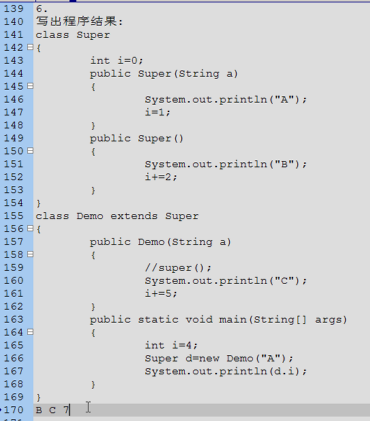
1. **实例化过程自我描述（正确）：**
2. **首先，从最高父类往下逐渐加载进内存，（Object类是JVM机运行时便加载进来），加载的时候先对成员变量进行默认初始化然后执行静态代码块，静态初始化时对于final修饰的或者static修饰的成员变量（常量）直接赋予设定值， 故最先是成员变量的默认初始化然后执行静态代码块，此过程过程从最高父类往下执行；这是从上往下。**
3. **然后，利用super（）；以及同一个类中利用this（）；逐步上升，除了这两个语句，其他没有执行任何代码；这是从下往上。**
4. **等到调用到最高父类的构造方法后，最高父类即object类，对每一个构造方法都一样，首先对当前类的成员变量进行显式初始化，紧接着执行当前类的构造代码块，然后执行当前构造方法中的从第二句开始的代码（ 第一句为显式或隐式的super语句或者this语句），就这样从高父类，一级一级地往下执行；这是从上往下走。**
5. **最终，完成了所有的初始化工作，创建了最低的子类的对象。**
6. **注意：（书写代码时，必须把所有的变量定义写在构造代码块和静态代码块的前面，否则出错，这也许就是为什么会先执行默认初始化或者显式初始化的原因吧，其次再执行静态代码块或者构造代码块了）。**
7. **对象的实例化过程中，父类构造方法对成员方法和成员变量的调用要与多态的向上转型后父类引用对成员变量和成员方法的调用类比起来：**

**（1）父类构造方法对成员变量的调用，调用的就是当前构造方法所在的类的成员变量，即父类中的成员变量，而父类中构造方法对成员方法的调用，同多态的一样，编译时看父类中是否有此方法，执行时从子类中寻找重写的方法，若没有重写，直接调用父类的方法；至于调用方法中又调用成员变量，则此成员变量附属此方法的所属，如果执行的方法是属于子类的，则方法中调用的变量就从子类中先寻找，若没有再利用父类的，若执行的方法是父类的，调用的变量也是父类的。**

**（2）多态中，对象的向上转型后，父类引用调用只有在调用非静态成员方法时，需要先观看子类的，如果子类没有，去父类中寻找，但是对于成员变量和静态方法而言，都是只看父类中的，子类中有与没有都没有任何关系。**

**（3）从上面看出父类的构造方法中对成员变量和成员方法的调用是一致的。重点是掌握父类构造方法或者父类引用又调用的方法中又调用成员变量的所属问题。所调用的方法是谁的，方法中调用的变量就是谁的。**

**第一题：**



解析：主方法中i是局部变量，没有用到。 由于子类中没有自己i变量，所以是直接利用父类的，故直接利用父类的变量i 时 已经为2，再加5，得到7。最终利用父类引用调用变量，显然调用的是父类中的变量值，但是此时子类中的只有这么一个父类的变量，所以直接就是这个唯一的变量。

变式题目：

**public** **class** Test **extends** FU{

**int i = 9; //(1)存在，（2）删除**

**public** Test(String str) {

i += 6;

System.***out***.println("子类构造方法 ： i = "+i);

}

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

**int** i = 8;

FU f = **new** Test("a");

Test t = (Test)f;

System.***out***.println("f.i = "+f.i +" t.i = "+ t.i );

}

}

**class** FU{

**int** i = 0;

**public** FU(){

i += 2;

System.***out***.println("父类构造方法 ： i = "+i);

}

}

结果分析（1）

父类构造方法 ： i = 2

子类构造方法 ： i = 15

f.i = 2 t.i = 15

（2）删除那一句：

父类构造方法 ： i = 2

子类构造方法 ： i = 8

f.i = 8 t.i = 8

总结：对于变量而言，对象实例化中，会把父类和子类的变量都封装进来，如果重名会用不同的标志加以区分，然后根据调用的引用类型进行选择调用，说白了，就是相当于两个变量了；但是如果子类中没有与父类变量重名，此时无论子类还是父类的引用都是在操作这个父类的变量，子类或者父类引用对其进行加减操作，则都会影响其值得变化。因为此时子类引用和父类引用指向的是一个变量。

第二题：

**public** **class** MianShiTi **extends** FuClass {

**int** num = 80;

**public** MianShiTi(String str) {

System.***out***.println("ZiClass Constructor, " +"num = " + num +" 子类中的变量值");

num += 9;

System.***out***.println("ZiClass :"+ "num + 9 = " + num + " this.num = "+ num);

eat(); //调用的是子类中的非静态方法

}

**static** **void** run() {//静态方法的重写

System.***out***.println("ZiClass is Running");

}

**void** eat() {//非静态方法的重写

System.***out***.println("ZiClass " + num +" 这个值是子类中的变量值，若是0，则说明还没有显式初始化");

*run*();//子类中调用静态方法，显然应该是子类中的静态方法

}

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

**int** num = 3;//局部变量只能在此方法中使用

MianShiTi m = **new** MianShiTi("A");

FuClass fc = m;//向上转型

System.***out***.println("下面显示出多态的性质：");

System.***out***.println("（1）\*\*\*\*\*\*多态的调用成员变量\*\*\*\*\*\*\*\*\*\*\*\*");

System.***out***.println("m.num = "+m.num + " fc.num = " + fc.num +"父类引用只调用父类中的变量值"+ " 局部变量 num = " + num);

System.***out***.println("（2）\*\*\*\*\*\*多态的调用静态方法\*\*\*\*\*\*\*\*\*\*\*\*");

m.*run*();

fc.*run*();

System.***out***.println("（3）\*\*\*\*\*\*多态的调用非静态方法\*\*\*\*\*\*\*\*\*\*\*\*");

m.eat();

fc.eat();

}

}

**class** FuClass {

**int** num = 90;

{//静态代码块

System.***out***.println("num = "+num + " 加上this也是一样 " + "this.num = "+**this**.num);

}

FuClass() {

System.***out***.println("FuClass : num = "+num);

num += 7;

System.***out***.println("FuClass : "+"num + 7 = "+ +num);

*run*();//调用静态方法，从结果看出调用的是父类的静态方法，此方法属于父类，则其中调用的变量是父类中的变量

eat();//调用非静态方法，从结果看出调用的是子类的重写的方法，此方法属于子类，则其中调用的变量就是子类中的变量

}

**static** **void** run() { //静态方法

System.***out***.println("FuClass is Running");

}

**void** eat() { //非静态方法

System.***out***.println("Eating Test");

System.***out***.println("EAT " + num);

}

}

结果：

num = 90 加上this也是一样 this.num = 90

FuClass : num = 90

FuClass : num + 7 = 97

FuClass is Running

ZiClass 0 这个值是子类中的变量值，若是0，则说明还没有显式初始化

ZiClass is Running

ZiClass Constructor, num = 80 子类中的变量值

ZiClass :num + 9 = 89 this.num = 89

ZiClass 89 这个值是子类中的变量值，若是0，则说明还没有显式初始化

ZiClass is Running

下面显示出多态的性质：

（1）\*\*\*\*\*\*多态的调用成员变量\*\*\*\*\*\*\*\*\*\*\*\*

m.num = 89 fc.num = 97父类引用只调用父类中的变量值 局部变量 num = 3

（2）\*\*\*\*\*\*多态的调用静态方法\*\*\*\*\*\*\*\*\*\*\*\*

ZiClass is Running

FuClass is Running

（3）\*\*\*\*\*\*多态的调用非静态方法\*\*\*\*\*\*\*\*\*\*\*\*

ZiClass 89 这个值是子类中的变量值，若是0，则说明还没有显式初始化

ZiClass is Running

ZiClass 89 这个值是子类中的变量值，若是0，则说明还没有显式初始化

ZiClass is Running