链滴

关于 java 继承的哪些事

作者：vcjmhg

原文链接：https://ld246.com/article/1577603909598

来源网站：链滴

引言

本文结合一个例子来说明继承实现的基本原理。

基类Base代码如下所示：

```
public class Base {
    public static int s;
    private int a;
    static {
        System.out.println("基类静态代码块，s:"+s);
        s=1;
    }
    {
        System.out.println("基类实例代码块，a:"+a);
        a=1;
    }
    public Base(){
        System.out.println("基类构造方法，a:"+a);
        a=2;
    }
    protected void step(){
        System.out.println("base s:"+s+",a:"+a);
    }
    public void action(){
        System.out.println("start");
        step();
        System.out.println("end");
    }
}
```

注意： Base包含一个静态变量s，一个实例变量a，一段静态初始化代码块，一段实例初始化代码块一个构造方法，两个方法step和action。

子类Child代码如下所示：

```
public class Child extends Base{
    public static int s;
    private int a;
    static {
        System.out.println("子类静态代码块,s:"+s);
        s=10;
    }
    {
        System.out.println("子类实例代码块,a:"+a);
        a=10;
    }
    public Child(){
        System.out.println("子类构造方法，a:"+a);
        a=20;
    }

    @Override
    protected void step() {
        System.out.println("child s:"+s+",a:"+a);
    }
}
```

注意： 子类Child继承了Base，也定义了和基类同名的静态变量s和实例变量a并且重写了方法step。

测试的main方法代码如下所示:

```
public static void main(String[] args) {
    System.out.println("------------- new Child()");
    Child c=new Child();
    System.out.println("\n-- c.action");
    c.action();
    Base b=c;
    System.out.println("\n --- b.action()");
    System.out.println("\n --- b.s:"+b.s);
    System.out.println("\n --- c.s:"+c.s);
    }
```

执行结果如下：

下边我们逐过程来解释下其背后发生了什么，并解释我们所提出的问题。

类的加载过程

在java中，所谓的类加载指的是将类的相关信息加载到内存中。在java中类是动态加载的，当第一次用这个类的时候才会加载，而且在加载一个类的时候会查看其父类是否被加载，如果没有则会加载其类。

一个类的信息主要包含以下几个部分：

类的加载过程顺序如下：

1. 分配内存保存类的信息
2. 给类变量赋默认值

注意： 数字类变量默认值都是0，boolean默认值是false，char是\u0000，引用型变量默认值都是null。

3. 加载父类
4. 设置父子关系
5. 执行类的初始化代码

以我们的例子来说，我们这里有三份类信息，分别是Child、Base、Object，内存布局如下图所示：

对象的创建过程

在类加载之后，new Child()就会创建Child对象，创建Child对象过程包括：

1. 分配内存
2. 对所有实例变量赋默认值
3. 执行实例初始化代码

在该部分分配的内存包括本类和所有父类的实例变量，不包含任何静态变量（因为类加载过程中这部变量的内存已经分配完成）。**实例的初始化代码执行先从父类开始，父类执行完成之后再执行子类。**需要注意的是在任何类执行初始化代码之前，任何实例变量都会被赋默认值。

但需要注意的是，**每一个对象除了保存类的实例变量之外，还保存着实际类信息的引用。**

方法调用的过程

在该部分我们分析前边所提出的问题，首先我们先来看c.action()，这句代码的执行过程如下：

1. 查看c的对象类型，找到Child类型，在Child类中找action方法，发现没有然后到父类中寻找。
2. 在父类Base中找到了方法action，开始执行action方法；
3. action先输出start，然后发现需要调用step()方法，就从Child类型中去寻找step()方法；
4. 在Child类型中找到了Child方法，执行Child类中的step()方法，输出Child类中的两个变量s(=10),a=20)的值；
5. 继续执行action方法输出end

在此处我们可以发现，**寻找要执行的实例方法的时候，是从对象的实际类型信息开始查找的，找不到时候在查找父类的类型信息。**

接着我们来看b.action()的执行过程，这句话实际上输出的结果和c.aciton()方法输出的结果是一样的这我们称之为动态绑定，而动态绑定实现的机制就是根据对象的实际类型信息查找要执行的方法，子型中查找不到才会查找父类。这里因为b和c执行的是相同的对象，所有执行的结果是一样的。

而对于变量部分我们发现，其访问过程是静态绑定的，即无论对于类变量还是实例变量访问的时候，问的实际变量都和其对象类的类型绑定，如b.s和c.s分别访问的是Base.s和Child.s。例子中实例变量是private类型的，如果是public我们会发现b.a访问的是Base类定义的实例变量a，而c.a访问的是对中Child类定义的实例变量a。

小结

在该部分我们分析的类和对象加载的过程，以及在过程中方法和代码块的执行顺序，我们可以得到以结论。

方法和代码的执行顺序：

1. 父类的类初始化代码块（static代码块）
2. 子类的类代码初始化块
3. 父类的实例初始化代码块
4. 父类的构造方法
5. 子类的实例初始化代码块
6. 子类的构造方法

重载方法的执行逻辑：

寻找要执行的实例方法的时候，是从对象的实际类型信息开始查找的，找不到的时候在查找父类的类信息。