

**方法区**

**堆**

**栈**

## 执行流程

1：Animal a = new Cat();

1.1:在栈中创建区域，类型为Animal，变量名:a;

1.2:在堆中new Cat();占用一块区域。地址值：[0x3a4]

1.3:spuer()实例化父类Animal。

1.3.1:new Animal();占用一块区域，地址值:0x3ab;

1.3.2:引用着在方法区中初始化[Animal中的所有方法,该引用为：[0x754]]。

1.3.3:将Animal()引用赋给spuer();spuer引用着Animal();

1.4:在方法区中初始化Cat类的所有方法,引用值为0x343。

1.5:将0x3a4赋给栈中的变量a;a就开始引用Cat()。

2：a.eat();

2.1:通过[0x3a4]找到Cat。

2.2:编译时期：先spuer()找到Animal中的方法。如果没有，则报错。

2.3:运行时：直接在Cat中找到eat(),当Cat中没有，再去Animal中找。

2.4:将eat()方法要方法区压栈，执行，输出:SOP(猫吃鱼);

3：a = new Dog();

3.1:在堆new Dog(),开辟一块新区域，地址值:0x87xfds

3.2:spuer();实例化父类Animal

3.2.1:new Animal()开辟新区域，地址值0x33fa;

3.2.2:成员方法引用着方法区中已初始化的[0x754];

3.2.3:将Animal地址值0x33fa赋给spuer();;spuer引用着Animal();

3.3:在方法区中初始化Dog类的所有方法,引用值为0x422ac。

3.4:将[0x87xfds]赋给栈中的变量a; a不再引用Cat,而是引用着Dog;这时堆中的Cat已成为垃圾，等待JVM空闲时来回收。

4：a.eat();

4.1:通过a变量引用值找到堆中标记为0x87xfds的区域。

4.2:编译时:先进super();去检查Animal引用的方法区中有没有eat()方法,如有没有则报错。

4.3:运行时:直接去Dog方法区中找到eat();如果Dog没有，再去执行super()调用父类的eat()方法。

4.4:从方法区中将eat()压栈，执行(SOP('狗吃粮'))。

5：a.shudy();

5.1:通过a变量引用值找到堆中标记为0x87xfds的区域。

5.2:编译时:先进super();去检查Animal引用的方法区中有没有shudy()方法,结果Animal中没有shudy()方法，所以就在编译时期就报错。

6：Dog d = (Dog)a;

6.1:在栈中开辟区域，存储类型为Dog,变量名d

6.2:将a向下转型，从Animal转为Dog来引用Dog;(将a变量赋给d)

6.3:a和d都指向堆中同一个Dog对象。

7：d.eat();//从Dog方法中压栈执行eat()方法，然后弹栈;

8：d.shudy();//从Dog方法中压栈执行shudy()方法，然后弹栈;

9：Cat c = (Cat) a;

9.1:在栈中开辟一块区域，存储类型：Cat,变量名称:c

9.2:将a向下转型，将Anmail引用Dao转为Cat引用Dog;结果抛出类型转换异常。Dog不能被转为Cat;

**总结：**

**1：多态=继承+重写+父类引用子类**  、

**如： Fu ff = new Zi();**

**2：普通成员**

变量:都参考左边，因为变量不存在重写，方法中调用变量采用就近原则。

方法:编译参考左边，运行参考右边。

**3：静态成员**

变量和方法:编译运行都参考左边;  因为静态与对象无关。

成员加静态修饰的没加private，都可以被类直接调用，所以参考的都是左边。