# 继承

**1.关键字:extends**

**2.格式:public class Zi extends Fu{}**

**Fu:父类,基类,超类;**

**Zi:子类,派生类**

**3.子类可以有父类的内容,也可以有自己特有的内容**

**4.提高代码的复用性,维护性;高耦合,独立性低.所以继承的使用是要看环境看条件看关系的**

**5.继承中变量访问的特点:子类局部范围🡪子类成员范围🡪父类成员范围**

**这里的局部范围指方法中;**

**这里的成员范围指类中方法外.**

**6.关键字:super 和this类似**

**this:本类对象的引用**

**super:父类对象的引用(父类存储空间的标识)**

**三种访问方式:**

**1.直接访问;**

|  |  |  |  |
| --- | --- | --- | --- |
| **2.this** | **this.成员变量;** | **this.(...);** | **this.成员方法();** |
| **访问本类成员方法** | **访问本类构造方法** | **访问本类成员方法** |
| **3.super** | **Super.成员变量;** | **super(...);** | **super.成员方法();** |
| **访问父类成员方法** | **访问父类构造方法** | **访问父类成员方法** |

**7.继承中构造方法的访问特点**

**子类中所有的构造方法默认都会访问父类中的无参构造方法**

**8.继承中成员方法的访问特点**

**通过子类对象访问(调用)一个方法,首先在子类成员范围找,再到父类成员范围找**

**9.super内存图**

**10.方法重写 @Override 出现在子类中,重写父类方法的标识和校验**

**子类中出现了和父类一模一样的方法声明(方法名相同)**

**当子类需要父类的功能,而且子类的这个功能又有自己的一些特殊的需求,就可以重写父类的功能(方法)**

**11.方法重写的注意事项**

**父类私有方法不能被子类重写**

**子类继承父类的方法,权限修饰符不能比父类低(public>默认>private)**

**返回值类型子类小于等于父类的返回值类型<这里的小于等于指的是应用数据类型继承关系中的小于等于,不是基本数据类型的小于等同于>**

**12.继承的注意事项**

**Java中类只支持单继承,不支持多继承,支持多层继承**

**13.包的概述和使用 package**

**文件夹,对类进行分类管理**

**#黑窗口在没有具体package(文件夹)包下的编译和运行**

**自动建包:**

**编译**

**javac-d.HelloWorld.java**

**执行**

**Java com.itheima.HelloWorld**

**14.import 导包**

**格式:import 包名;**

**举例:import java.util.Scanner;**

**15.修饰符 权限修饰符**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **修饰符** | **同一个类中** | **同一个包中子类无关类** | **不同包的子类** | **不同包的无关类** |
| **private** | **✔** |  |  |  |
| **默认** | **✔** | **✔** |  |  |
| **protected** | **✔** | **✔** | **✔** |  |
| **public** | **✔** | **✔** | **✔** | **✔** |

**16.修饰符 状态修饰**

**final 最终的意思,可以修饰类,成员方法,成员变量**

**final修饰的类为最终类,不能被继承;**

**final修饰的成员方法是最终方法,不能被重写;**

**final修饰的成员变量是常量,不能被再次赋值.**

**17.final修饰局部变量**

**final修饰的局部变量是基本数据类型,其数据值不可再发生改变;**

**final修饰的局部变量是引用数据类型,其地址值不可再发生改变,但是地址里面的内容是可以发生改变的**

**18.static关键字**

**Static是静态的意思,可以修饰成员方法,成员变量**

**Static修饰的特点**

**可以被类的所有对象所共享**

**这也是判断是否使用静态的条件**

**可以被对象调名用也可以被类名调用**

**推荐使用类名调用**

**19.static访问特点**

**静态的成员方法只能访问静态**

**静态成员方法:只能访问静态的成员方法和静态的成员变量**

**非静态的成员方法:能访问静态和非静态的成员方法和变量**

**父类使用静态修饰的成员不能子类继承**

# 多态

**1.多态的前提和体现**

**有继承/实现关系**

**有方法重写**

**有父类引用指向子类对象**

**Animal a = new Cat();**

**2.多态中成员的访问特点**

**成员变量: 编译看左边,执行看左边;**

**编译看左边,执行看右边;**

**为什么成员变量和成员方法访问会有区别?**

**因为成员方法有重写,而成员变量没有**

**3.多态的好处和弊端**

**好处:提高了程序的扩展性**

**定义方法的时候,使用父类作为参数,运行的时候就使用具体的子类参与操作**

**弊端;不能使用子类的特有功能**

**4.多态中的转型**

**向上转型 如 Animal a = new Cat();**

**从子到父 父类引用指向子类对象**

**向下转型 如 Cat c = (Cat)a**

**从父到子 父类引用转为子类对象**

**5.多态转型的内存图**

**Animal Cat Dog**

**Animal a = new Cat();**

**Cat c = (Cat)a;**

**a = new Dog();**

**Dog d = (Dog)a;**

**Cat c = (Cat)a; //ClassCastException 类型转换异常**

**a 现在是Dog类型,转换成Cat是不行的**

**6.多态版猫狗案例**

**练习**