复习：

1. 方法

A. 意义：为了重复利用

B. 明确方法的返回值类型，明确参数

C. 方法签名---方法名+参数列表，形参和实参

D. 方法的重载---同一个类中，方法名一致而参数列表不同

E. 方法的递归---方法调用了自己本身

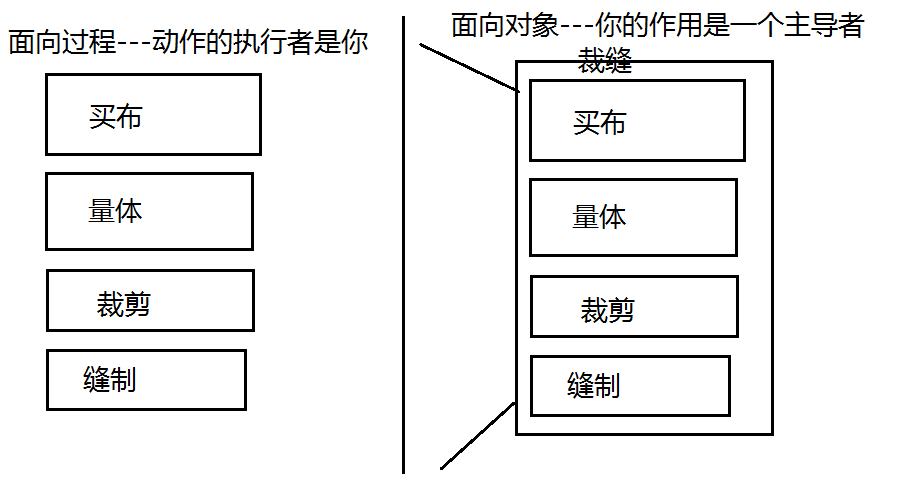
练习：n! --- 6! = 6 \* 5 \* 4 \*3 \* 2 \* 1

## 面向对象

面向对象和面向过程都是思维方式

面向过程：关注流程中的每一个步骤是如何执行的

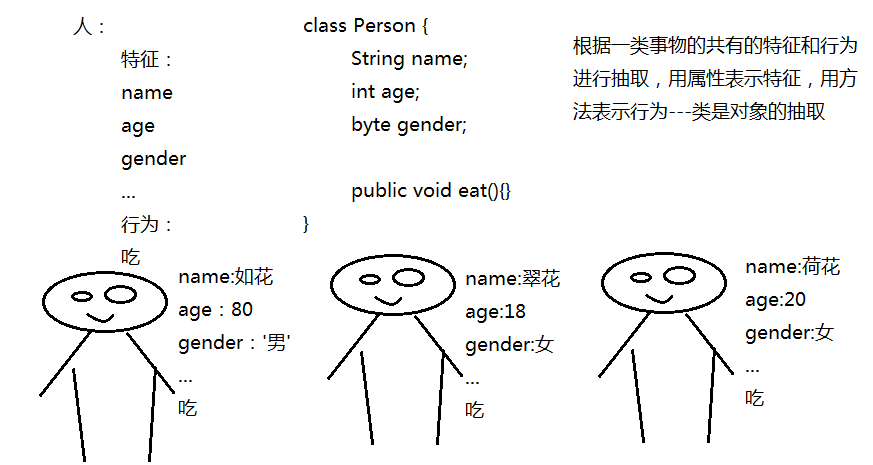
面向对象：着重找的对象，只要找到了这个对象，自然拥有了对象所有的一切功能



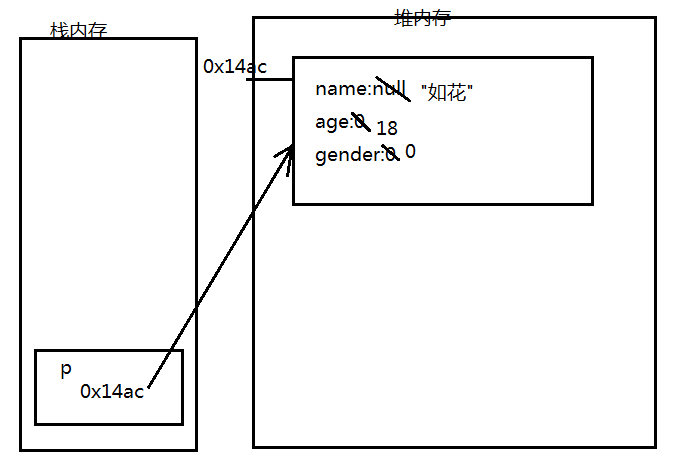
面向对象基于面向过程的。

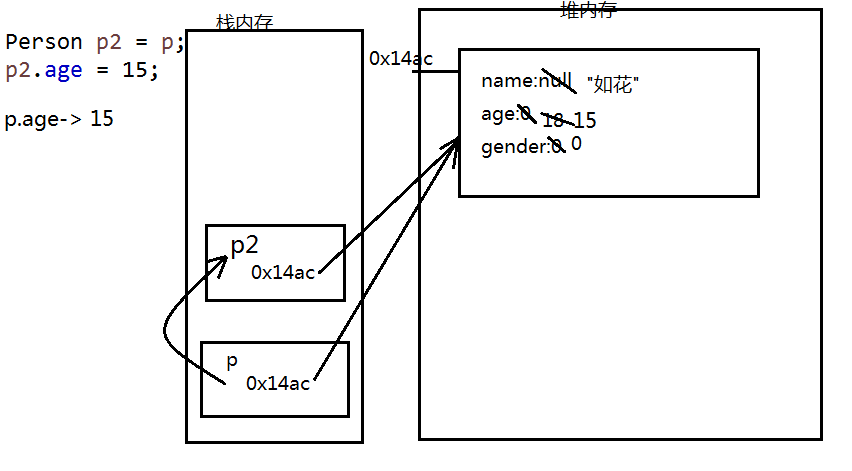
面向对象一定比面向过程好吗？---相对简单的场景，适用于面向过程；相对复杂的场景适用于面向对象

## 类和对象的关系



### 对象在内存中的存储





注意：引用类型赋值给的是地址

### 成员变量和局部变量

1. 定义位置：成员变量定义在类内方法外（属性）；局部变量是定义在方法或者语句内的

2. 作用范围：成员变量作用在整个类中；局部变量是只能作用在对应的方法或者语句中

3. 存储位置：成员变量存储在堆内存中，并且会自动的赋予初始值；局部变量是存储在栈内存中，不会自动赋予默认值

4. 生命周期：成员变量是在对象创建的时候存储到堆内存中，对象被回收的时候才会被释放；局部变量在方法执行的时候存储，方法执行完成之后就立即销毁

### 构造方法

在java中的任何一个类中都有构造方法。

如果一个类中没有手动定义构造方法，那么在编译的时候会自动添加一个默认的无参的构造方法

构造方法：与类同名而没有返回值类型

如果一个类中手动提供构造，则编译的时候不在添加默认的构造方法

如果一个类中只提供了含参构造，那么在创建对象的时候必须传参

构造方法的作用就是用于创建对象

### this关键字

注意：在java中，所有的非静态的方法和非静态的属性都需要通过对象来调用

this用于在本类中调用本类的方法和属性的。

this表示当前在活动的对象的引用，可以认为是一个虚拟对象。

this语句用于调用本类中的对应形式的构造方法---this语句必须放在构造方法的第一行

### 构造代码块

在类中{}包括起来的一段代码，也叫初始化代码块---在创建对象的时候，先于构造方法执行

### 局部代码块

限制变量的作用范围缩小，缩短了这个变量的生命周期，提供栈内存的利用率

## 面向对象的特征

封装、继承、多态（抽象）

### 封装

体现形式：内部类，方法，属性的私有化---将属性设置为私有的，然后通过提供对外的访问和设置的方法来间接操作这个属性，在方法内可以对属性进行限制使它的值更加符合场景要求

优势：提高代码的复用性，保证数据的合法性

### 权限修饰符

用于限制属性或者方法的使用范围的

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 本类中 | 子类中 | 同包类中 | 其他类中 |
| public | 可以 | 可以 | 可以 | 可以 |
| protected | 可以 | 可以 | 可以 | 不可以 |
| 默认 | 可以 | 同包子类可以 | 可以 | 不可以 |
| private | 可以 | 不可以 | 不可以 | 不可以 |

public>protected>默认>private

### 继承

对一些类进行抽取，将其中的共有的特征和行为提取到一个新的类中，利用extends关键字让原来的类和新的类产生关系，而这种关系叫继承。新的类称之为叫父类（超类/基类），原来的类就称之为子类（派生类）。---通过继承，子类可以使用父类中的一部分的方法和属性

在java中，支持的是单继承---一个子类只能继承一个父类，但是一个父类可以有多个子类

单继承和多继承的优劣性比较：多继承在代码的复用性上要优于单继承，但是单继承在方法调用的明确度上要好于多继承

class A {

public int m(){return 5;}

}

class B {

public int m(){return 1;}

}

class C extends A,B{}

C c = new C();

int i = c.m(); // 导致方法调用混乱

优势：提高复用性，避免方法调用的混乱

### super关键字

表示父类对象的引用，在子类中调用父类中的方法或者属性的

super语句---通过super语句来调用父类中对应形式的构造方法创建父类对象---如果方法中没有手动给定super语句，那么有一句默认的无参super()---如果父类中只提供了含参构造，那么这个时候子类的构造函数中必须手动添加对应形式的super语句 --- super语句必须放在子类构造函数的首行 --- 在创建子类对象的时候，必须先创建父类对象 --- this语句和super语句不能共存

class A {

}

class B extends A {

public B(){}

public B(int i){

this();

}

}

### 方法的重写

在父子类中，存在方法签名相同的非静态方法—方法的重写/覆盖---两等两小一大

1. 方法签名一致

class A {}

class B extends A {}

class C {

public void m(A a){}

}

class D extends C {

public void m(B b){} // 由于参数列表不一样，所以没有构成重写

}

2. 如果父类方法的返回值类型是基本类型/void，那么子类重写的方法的返回值类型要一致

class A{

public double m(){}

}

class B extends A {

public double m(){}

}

3. 子类重写的方法的权限修饰符的范围要大于等于父类中方法的权限修饰符的范围

class A{

protected void m(){}

}

class B extends A {

protected void m(){}

}

4. 如果父类的方法的返回值类型是引用类型，那么子类重写的方法的返回值类型是父类方法返回值类型的子类或者是和父类方法的返回值类型一致。

class A{}

class B extends A {}

class C {

public B m(){}

}

class D extends C {

public B m(){}

}

扩展：

class A {

private void m(){} --- 只能在本类中使用，对子类不可见

// 如果父类的方法是私有的，那么对子类的方法不起限定作用

}

class B extends A {

public int m(){return 5;}---对的

}