**·this调用本类属性**

this可以算是Java里面比较复杂的关键字，因为this的使用形式上决定了它的灵活性，在程序中可以实现以下三类结构的描述

·当前类中的属性: this.属性;

·当前类中的方法（普通方法，构造方法）: this(), this.方法名称()

·描述当前对象;

使用 this 调用当前类属性

public Person(String n ,int a){ //有参构造

name = n;

age = a;

}

此时n 和 a 代表不明确 我们想将其换成name 和 age 那么此时 当我们想要调用当前类的属性时 需要用this

public Person(String name ,int age){ //有参构造

this.name = name;

this.age = age;

}

使用this调用方法除了调用属性之外，this也可以实现方法的调用，但是对于方法的调用就必须考虑构造与普通方法。

·构造方法调用 （this()）:使用关键字 new 实例化对象的时候 才会调用构造方法；

·普通方法调用（ this.方法名称() ）:实例化对象产生之后就可 以调用普通方法。

范例: （普通方法调用）

public Person (String name,int age){

this.setName(name); //加上this为标准写法

setName(name); //加与不加都表示本类方法

}

范例: （构造方法调用） ：假设现在类中一共有三个构造方法，但是要求不管调用那个方法，都必须，都执行一句输出语句“一个新的Person类方法”

传统做法实现:

class Person {

private String name ; //个人姓名

private int age ; //年龄

//构造方法重载

public Person(){

System.out.println("\*\*\* 一个新的Person类对象实例化了");

}

//构造方法重载

public Person(String name){

System.out.println("\*\*\* 一个新的Person类对象实例化了");

}

//构造方法重载

public Person(String name , int age){

System.out.println("\*\*\* 一个新的Person类对象实例化了");

}

public void tell (){

System.out.println("姓名" + name + ",年龄" + age)

}

//以下省略属性的setter与getter方法

...

}

public classs JavaDemo {

public static void main (String args[]){

Person per = new Person("王五",18);

per.tell();

}

}

优化后的代码:

class Person {

private String name ; //个人姓名

private int age ; //年龄

//构造方法重载

public Person(){

System.out.println("\*\*\* 一个新的Person类对象实例化了");

}

//构造方法重载

public Person(String name){

this(); //调用无参

this.name = name;

}

//构造方法重载

public Person(String name , int age){

this.(name) //调用单参构造方法

this.age = age;

}

public void tell (){

System.out.println("姓名" + name + ",年龄" + age)

}

//以下省略属性的setter与getter方法

...

}

**·对于本类构造方法的互相调用需要注意以下几点重要问题:**

·构造方法必须在实例化新对象的时候调用 ，所以 this() 语句 必须放在构造方法的首行（也就是说 this() 只能也只允许放在构造方法首行）

·构造方法互相调用时请保留程序的出口,别形成死循环;

错误示例:

public Person(){

//出现了死循环

this("name",age);

System.out.println("\*\*\* 一个新的Person类对象实例化了");

}

public Person(String name){

this(); //调用无参

this.name = name;

}

public Person(String name , int age){

this.(name) //调用单参构造方法

this.age = age;

}

在上述代码中当参数达到4个甚至更多时，我们将会存在更多的构造方法（单参，无参，两参，三参，...），在这些构造方法里会存在大量的赋值操作，而我们可以使用 this(); 进行代码的优化;在前期的学习中我们将更加注重如何去消除代码的重复性;

eg:（三参 name age sex）

public Person(){

this("无名氏", 18 , "男");

}

public Person(String name){

this(name,18,"男");

}

public Person(String name , int age){

this.(name ,age ,"男");

}

public Person(String name , int age ,String sex){

this.(name ,age , sex);

}

关于this的一个案例：详情请到 (src/this关键字) 文件夹中寻找;（JavaDemo.java）

在以后进行项目开发的过程中，简单java类都将作为一个重要的组成部分存在，简单的来说简单java类无处不在，并且有可能组成一系列变化。所谓的简单java类就是描述一类信息的程序类: 描述一本书 ，描述一个人 ，一辆车;并且在 这个类中没有什么特别复杂的逻辑操作，只是作为一种信息存储的媒介存在;对于简单java类而言，其核心的开发结构如下:

·类名一定要有意义，可以明确的描述某一类事务

·类之中的属性都必须使用private进行封装，同时封装后的属性必须有getter setter 方法

·类之中可以提供无数构造方法，但是必须要保留无参构造方法；

·类之中不允许出现任何输出语句，所有内容必须返回;

·【非必须】可以提供一个获取对象详细信息的方法，命名为 getinfo()