**·成员属性封装**

在类之中的组成就是属性和方法，一般而言方法是对外提供服务的，所以不会进行封装处理，而对于属性由于其需要较高的安全性，所以往往需要对其进行保护，这个时候就需要采用属性封装对属性进行保护

属性不封装之下的问题:

class Person {

String name ; //个人姓名

int age ; //年龄

public void tell (){

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

}

}

public class JavaDemo {//主类

public Static void main (String args[]){

Person per = new Person();

per.name = "张三" ; //在类外部修改属性

per.age = -18 ; //在类外部修改属性

per.tell() ;

}

}

此时在person 类中提供的name与gae 两个属性并没有进行封装处理，这样外部就可以直接调用了，但是有可能所设置的数据是错误的数据。如果想解决这样的问题就可以利用 private 关键字对属性进行封装。

eg: 对属性进行封装（而一旦封装之后外部将不能直接访问，注: 对外部不可见，但是对类的内部是可见的，那么如果要想让外部的程序可以访问封装的属性，则在java开发中提供有如下要求:）

·【setter 与 getter 】设置或取得属性

class Person {

private String name ; //个人姓名

private int age ; //年龄

public void tell (){

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

}

//以下省略属性的setter与getter方法

...

//可在setter与getter中进行检测，来达到保护属性的效果

}

public class JavaDemo {//主类

public Static void main (String args[]){

Person per = new Person();

per.name = "张三" ; //在类外部修改属性

per.age = -18 ; //在类外部修改属性

per.tell() ;

}

}