封装性

# 引言

封装的基础实现。

# 具体内容

在Java中所有面向对象的概念都是以类与对象的关系为主的。那么下面首先通过一个具体的程序来研究一下，为什么需要提供有封装性。

|  |
| --- |
| class Person {  String name;  int age;  public void tell() {  System.out.println("姓名：" + name + "，年龄：" + age);  }  }  public class TestDemo {  public static void main(String[] args) {  Person per = new Person();  per.name = "A";  per.age = -30;  per.tell();  }  } |

现在的代码没有语法错误，但是从现实的角度来讲，这个代码是有错误的，而且问题严重。这样的错误严格来讲就属于业务错误。那么造成这种出错的原因是什么？

最大的错误在于当前类中的属性可以直接被类外部的对象所直接调用。所以，此时就认为这样的操作是属于不安全的操作。

那么现在最需要解决的问题是将内部的属性保存起来，即：不让外部直接操作。为此在Java中提供了一个private关键字，利用此关键字额可以实现封装。

|  |
| --- |
| class Person {  private String name;  private int age;  public void tell() {  System.out.println("姓名：" + name + "，年龄：" + age);  }  } |

加入了private之后，就表示此时name与age两个属性只能够被Person类所访问。

如果使用了private定义的属性，类的外部不能够直接进行访问了，所以安全性是最高的。

那么如果现在需要通过对象操作类中的属性，在Java中就有了一个明确的要求：可以使用setter、getter方法设置或取得封装属性内容，以private String name；封装属性为例；

* 设置属性：public void setName(String nam)；
* 取得数据：public String getName();

范例：修改程序，使得外部可以访问私有属性

|  |
| --- |
| class Person {  private String name;  private int age;  public void tell() {  System.out.println("姓名：" + name + "，年龄：" + age);  }  public void setName(String name) {  this.name = name;  }  public void setAge(int age) {  this.age = age;  }  public String getName() {  return name;  }  public String getAge() {  return age;  }  } |

开发原则：以后只要是类中的属性全部使用private封装，封装后的属性必须严格按照要求编写setter、getter方法。

如果非要进行一些检测操作，则可以修改setter方法。（只是现在临时的说明）

|  |
| --- |
| class Person {  private String name;  private int age;  public void tell() {  System.out.println("姓名：" + name + "，年龄：" + age);  }  public void setName(String name) {  this.name = name;  }  public void setAge(int age) {  if (age >= 0 && a <= 250) {  this.age = age;  }    }  public String getName() {  return name;  }  public String getAge() {  return age;  }  }  public class TestDemo {  public static void main(String[] args) {  Person per = new Person();  per.setName("A");  per.setAge(-30);  per.tell();  }  } |

# 总结

1. private声明的属性只能够被类的内部所访问；
2. private声明的属性必须有对应的setter、getter方法，而且方法名称要求按照严格标准编写。