Java面向对象封装

封装的概念

封装具体实现步骤

  示例代码如下

封装具体语法

封装的概念

Java封装是一种面向对象的编程方法，它的主要思想是将类的内部细节（包括数据和方法）隐藏起来，对外只提供公共的访问接口，从而保证了程序的安全性和稳定性。

封装具体实现步骤

1、声明类的成员变量时，使用private关键字进行修饰，将变量设为私有属性（只能在类的内部访问）。

2、为每个私有属性提供getter和setter方法，用来读取和修改私有属性。

3、getter方法用于获取私有属性的值，setter方法用于设置私有属性的值。在方法内部，可以添加一些控制语句用于判断赋值是否合法。

示例代码如下

```java
public class Student {
    private String name; // 学生姓名
    private int age; // 学生年龄

    // 设置姓名
    public void setName(String name) {
        this.name = name;
    }

    // 获取姓名
    public String getName() {
        return name;
    }

    // 设置年龄
    public void setAge(int age) {
        if (age > 0 && age < 120) { // 判断年龄是否合法
            this.age = age;
        } else {
            System.out.println("输入的年龄不合法！");
        }
    }

    // 获取年龄
    public int getAge() {
        return age;
    }
}
```

在上面的代码中，我们将name和age属性都设为了private，然后分别为它们提供了getter和setter方法，以便在外部可以访问和修改这些属性。同时，在setAge()方法中添加了一些控制语句，用于判断输入的年龄是否合法。

使用封装的好处是，一方面可以保护数据的安全性，不会因为误操作而导致数据被破坏；另一方面，也方便了代码的维护和升级，如果后续需要修改属性的实现方式，只需要修改相应的getter和setter方法即可，对外部程序不会造成影响。

封装具体语法

```java
访问修饰符 类型 变量名;

// 或者

访问修饰符 返回值类型 方法名(参数列表) {
    // 方法体
}
```

其中，访问修饰符可以是public、protected、private、default四种之一，分别表示访问权限从高到低。对于类的成员变量和方法，通常将其设置为private，表示只能在当前类中被访问，外部程序无法直接访问。为了使外部程序也能够访问这些属性，可以提供getter和setter方法来获取和修改私有属性。

另外，Java中的封装还可以使用关键字this来表示当前对象，以便在方法内访问当前对象的属性或调用其他方法。例如：this.age 表示当前对象的年龄属性，this.setName() 表示调用该对象的setName()方法。

总之，Java中的封装将类的数据和方法进行了包装和隐藏，对外只提供了公共的访问接口，从而保证了程序的安全性和稳定性。