Java 抽象类 Java 接口

Java 封装

在面向对象程式设计方法中，封装（英语：Encapsulation）是指一种将抽象性函式接口的实现细节部份包装、隐藏起来的方法。

封装可以被认为是一个保护屏障，防止该类的代码和数据被外部类定义的代码随机访问。

要访问该类的代码和数据，必须通过严格的接口控制。

封装最主要的功能在于我们能修改自己的实现代码，而不用修改那些调用我们代码的程序片段。

适当的封装可以让程式码更容易理解与维护，也加强了程式码的安全性。

封装的优点

- 1. 良好的封装能够减少耦合。
- 2. 类内部的结构可以自由修改。
- 3. 可以对成员变量进行更精确的控制。
- 4. 隐藏信息，实现细节。

实现Java封装的步骤

1. 修改属性的可见性来限制对属性的访问（一般限制为private），例如：

```
public class Person {
private String name;
private int age;
}
```

这段代码中，将 **name** 和 **age** 属性设置为私有的，只能本类才能访问，其他类都访问不了，如此就对信息进行了隐藏。

2. 对每个值属性提供对外的公共方法访问，也就是创建一对赋取值方法，用于对私有属性的访问，例如：

```
public class Person{
private String name;
private int age;

public int getAge(){
return age;
}

public String getName(){
return name;
}

public void setAge(int age){
this.age = age;
}
```

```
public void setName(String name){
this.name = name;
}
}
```

采用 **this** 关键字是为了解决实例变量（private String name）和局部变量（setName(String name)中的name变量）之间发生的同名的冲突。

实例

让我们来看一个java封装类的例子：

EncapTest.java 文件代码：

```
/* 文件名: EncapTest.java */
public class EncapTest{
private String name;
private String idNum;
private int age;
public int getAge(){
return age;
}
public String getName(){
return name;
}
public String getIdNum(){
return idNum;
}
public void setAge( int newAge){
age = newAge;
}
public void setName(String newName){
name = newName;
}
public void setIdNum( String newId){
idNum = newId;
}
}
```

以上实例中public方法是外部类访问该类成员变量的入口。

通常情况下，这些方法被称为getter和setter方法。

因此，任何要访问类中私有成员变量的类都要通过这些getter和setter方法。

通过如下的例子说明EncapTest类的变量怎样被访问：

RunEncap.java 文件代码：

```
/* F文件名 : RunEncap.java */
public class RunEncap{
public static void main(String args[]){
EncapTest encap = new EncapTest();
encap.setName("James");
encap.setAge(20);
```

```
encap.setIdNum("12343ms");
System.out.print("Name : " + encap.getName()+
" Age : "+ encap.getAge());
}
}
```

以上代码编译运行结果如下:

```
Name : James Age : 20
```

Java 抽象类 Java 接口

点我分享笔记