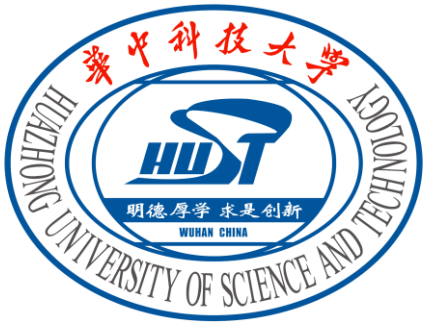
基于Java的面向对象程序设计

陈维亚

weiya_chen@hust.edu.cn

华中科技大学软件学院

第4讲：封装



目录

1. 封装的意义

❑ 封装 Encapsulation

封装是指一种将抽象性函数接口的实现细节部分包装、隐藏起来的方法。

封装可以防止一个类的代码和数据被外部类定义的代码随机访问。

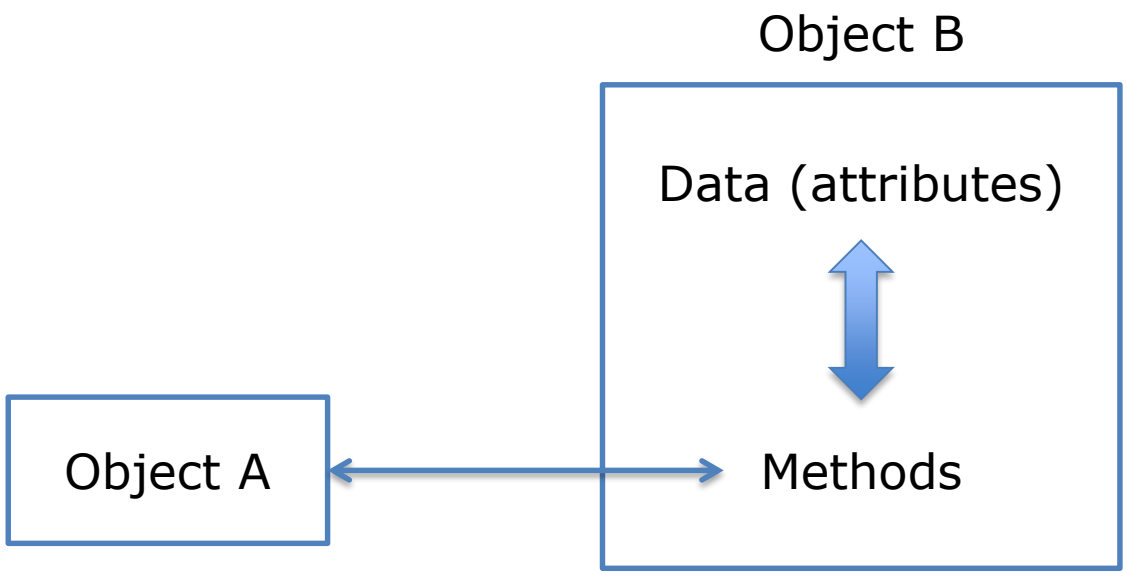
而要访问该类的代码和数据，必须通过严格的接口控制。

我不让你知道  你不需要知道

面向接口编程

1. 封装的意义

- 封装 Encapsulation



1. 封装的意义

❑ 封装的好处

从设计上讲，为了实现模块化

1. 以类为最小单元，减少修改带来的连锁效应；
2. 可以对成员变量进行更精确的控制（getter, setter）；

从使用上讲，减少了复杂度

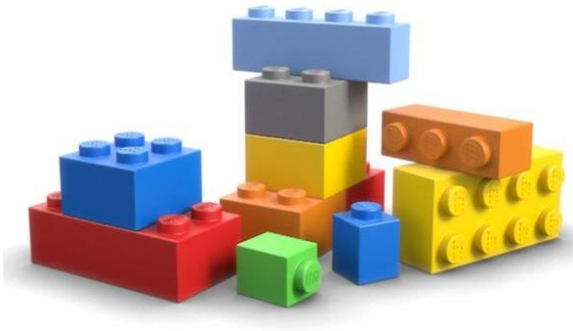
1. 代码使用者无需考虑实现细节就能直接使用一个类；
2. 更有效的协作；

良好的封装能够减少代码的耦合，抵御变化

```
public class Dog {
   String breed;
   String name;
   int age;

   void Dog(){}

   void bark(){}
   void run(){}
   void sleep(){}
}
```



1. 封装的意义

- 封装的好处

```
public class Dog {
  private String breed;
  private String name;
  private int age;

  void Dog(){}

  void bark(){}
  void run(){}
  void sleep(){}
}
```

```
public class PetStore{
  public static void main(String[]
args){
    Dog dog = new Dog();
    dog.age = 100;
  }
}
```

2. 封装的实现

❑ Java访问权限

依靠4种关键字，可修饰类、属性及方法

private: 只能被该类的对象访问。

default: 即不加任何访问修饰符，只允许在同一个包中进行访问。

protected: 只能被类本身的方法及子类访问，即使子类在不同的包中也可以访问。

public : 不仅可以跨类访问，而且允许跨包（package）访问。

权限修饰符	同一个类	同一个包	不同包的子类	不同包的非子类
Private	√			
Default	√	√		
Protected	√	√	√	
Public	√	√	√	√

2. 封装的实现

❑ Java访问权限

理想的封装状态是：

只有一些精心设计的、注释清晰的接口方法为public；

成员变量和其他的方法都是private，它们组成了一个类的内部实现，与类的外部无关

2. 封装的实现

```java
/* 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;
   }
}
```

```java
/*  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());
    }
}
```

2. 封装的实现

- this关键字

```
1  class Rectangle{
2      private int  width;
3      private int height;
4      public void setWidth(int width){
5          this.width = width;
6      }
7      public int getWidth(){
8          return width;
9      }
10     public void setHeight(int henght){
11         this.henght = height;
12     }
13     public int getHeight(){
14         return height;
15     }
16 }
```

3. 源文件的引用

❑ 源文件声明规则

当在一个源文件中定义多个类时，要特别注意这些规则：

一个源文件中只能有一个public类;

源文件的名称应该和public类的类名保持一致;

如果一个类定义在某个包中，那么package语句应该在源文件的首行;

```
package animals;
import java.io.*;
…
```

import语句和package语句对源文件中定义的所有类都有效。在同一源文件中，不能给不同的类不同的包声明。

3. 源文件的引用

❑ 源文件引用

Java包

包主要用来对类和接口进行分类。当开发Java程序时，可能编写成百上千的类，因此很有必要对类和接口进行分类。

import 语句

在Java中，如果给出一个完整的限定名，包括包名、类名，那么Java编译器就可以很容易地定位到源代码或者类。

import语句就是用来提供一个合理的路径，使得编译器可以找到某个类。

例如，下面的命令行将会命令编译器载入java_installation/java/io路径下的所有类
import java.io.*;

【练习 1】实现一个灰度图像上的点类Point，该类具有如下功能：

1) 可平移若干个像素；

2) 可计算与另一个点的欧氏距离；

3) 可判断是否在一个圆的内部；

4) 可计算与另一个点的灰度差异；

4. 章节小结

封装是什么

为啥要封装

如何使用封装

下节预告

继承