本文内容是封装的梳理和总结，本文内容包括：

1 封装

1.1 什么是封装

封装是用来设置**类的成员**在**什么地方**可以**被访问**。

- 类的成员：是指包括构造函数，实例属性，实例方法，静态属性，静态方法
- 什么地方：是指类的内部、同包中的其他类，同包或子类，任何包中的任何类
- 被访问：是指对方法的调用，对属性值的读取，对属性值的修改

1.2 如何实现封装

实现封装需要使用访问修饰符，访问修饰符共四个

- private ： 类内可访问
- 默认：同包可访问
- protected：同包或子类可访问
- public：任意类可访问

1.3 getter和setter方法

对于属性的读写设置，一个典型的做法是

- 将属性设置为private，如此一来，只要不在类内就无法读写

- 若提供public修饰的get方法，就可以读取属性值，若不提供就不能读取属性值
- 若提供public修饰的set方法，就可以修改属性值，若不提供就不能修改属性值

把这种公有方法封装私有属性称为getter和setter

1.4 来个例子

需求：

- 有电影类Movie
- 包括电影名称属性name，类外允许读取，不允许修改
- 包括票价属性price，类外可读取，可修改
- 编写代码，实现对Movie属性的封装

分析：

- 电影名称是只读属性，只提供get方法即可
- 电影价格是读写属性，提供get和set方法即可

代码：

```
public class Movie {
    private String name;  //属性私有化
    private double price; //属性私有化

    // 构造函数初始化属性值
    public Movie(String name,double price){
        this.name = name;
        this.price = price;
    }

    // 公有get方法，读取电影名称
    public String getName() {
        return name;
    }

    // 公有get方法，读取电影价格
    public double getPrice() {
        return price;
    }

    // 公有set方法，修改电影价格
    public void setPrice(double price) {
```

```
        this.price = price;
    }
}
```

解析：

- name属性只有get方法，没有set方法，实现只读属性
- price属性有get方法和set方法，实现读写属性

1.5 再来个例子

需求：

- 有一个类Factory，要求Factory类外不允许实例化该类，类外若想获得Factory类的对象，需要调用该类公有的静态方法createInstance()，该方法可以返回Factory类的对象。
- 编写代码实现该需求

分析：

- 对于类外不允许实例化Factory的要求，实现方法是将Factory的构造函数私有化，因为私有的构造函数使得类外是不能调用的，也就实现了类外不允许实例化Factory类。
- 在公有静态createInstance()方法中实例化Factory类，并返回实例化的对象，因为createInstance()方法在Factory的类内，类内是可以调用私有构造函数的。

代码：

- Factory类

```
public class Factory {
    // 构造函数私有化，类外就无法调用了，因此就实现了类外不能创建该类的对象
    private Factory(){

    }
    // 公有静态方法，返回Factory对象
    public static Factory createInstance(){
        return new Factory();//此处在Factory的类内，能够调用private的构造函数
    }
}
```

- 测试类

```
public class Test {
    public static void main(String[] args) {
        // 调用公有静态方法，获取Factory对象
        Factory factory = Factory.createInstance();
    }
}
```

解析：

- 构造函数私有后，类外不能new对象了
- 通过公有的静态方法可以获取对象

类似于本例这种对象不是new出来的，而是类名调用公有静态方法获得的，这种使用场景很多，希望以后你遇到这种情况时，能够想起这个例子。