2、面向对象进阶

2.1、封装 private

我们观察如下代码：

```
    class Person{
        private String name ;       // 表示姓名
        private int age ;           // 表示年龄
        void tell(){
            System.out.println("姓名：" + name + "；年龄：" + age) ;
        }
    };
    public class Demo{
        public static void main(String args[]){
            Person per = new Person() ;
            per.name = "张三" ;
            per.age = -30 ;
            per.tell() ;
        }
    };
```

以上的操作代码并没有出现了语法错误，但是出现了逻辑错误 （年龄-30岁）

在开发中， 为了避免出现逻辑错误， 我们建议对所有属性进行封装，并为其提供setter及getter方法进行设置和取得操作。

修改代码如下：

```
class Person{
    private String name ;       // 表示姓名
    private int age ;           // 表示年龄
    void tell(){
        System.out.println("姓名：" + getName() + "；年龄：" + getAge()) ;
    }
    public void setName(String str){
        name = str ;
    }
    public void setAge(int a){
        if(a>0&&a<150)
            age = a ;
    }
    public String getName(){
        return name ;
    }
    public int getAge(){
        return age ;
    }
};
public class OODemo10{
    public static void main(String args[]){

        Person per = new Person() ;
```

```
        per.setName("张三") ;
        per.setAge(-30) ;
        per.tell() ;
    }
};
```

2.2、this

在Java基础中，this关键字是一个最重要的概念。使用this关键字可以完成以下的操作：

- 调用类中的属性
- 调用类中的方法或构造方法
- 表示当前对象

2.3、static

- 概述

static表示“静态”的意思，可以用来修饰成员变量和成员方法(后续还会学习 静态代码块 和 静态内部类)。
static的主要作用在于创建独立于具体对象的域变量或者方法
简单理解：
被static关键字修饰的方法或者变量不需要依赖于对象来进行访问，只要类被加载了，就可以通过类名去进行访问。
并且不会因为对象的多次创建 而在内存中建立多份数据

- 重点：

1. 静态成员 在类加载时加载并初始化。
2. 无论一个类存在多少个对象 ，静态的属性，永远在内存中只有一份(可以理解为所有对象公用)
3. 在访问时： 静态不能访问非静态 ，非静态可以访问静态 ！

2.4、代码块

普通代码块
　　在执行的流程中 出现的 代码块， 我们称其为普通代码块。

构造代码块
　　在类中的成员代码块， 我们称其为构造代码块， 在每次对象创建时执行， 执行在构造方法之前。

静态代码块
　　在类中使用static修饰的成员代码块， 我们称其为静态代码块， 在类加载时执行。 每次程序启动到关闭 ，只会执行一次的代码块。

同步代码块
　　在后续多线程技术中学习。

面试题:
　　构造方法 与 构造代码块 以及 静态代码块的执行顺序:
　　　　静态代码块 --> 构造代码块 --> 构造方法

2.5、包

2.5.1. 包介绍

1、把功能相似或相关的类或接口组织在同一个包中，方便类的查找和使用。
2、包如同文件夹一样，不同的包中的类的名字是可以相同的，当同时调用两个不同包中相同类名的类时，应该加上包名加以区别。因此，包可以避免名字冲突。
3、包也限定了访问权限，拥有包访问权限的类才能访问某个包中的类。

2.5.2. 包的使用规则

- 包中java文件的定义：
  在.java文件的首部， 必须编写类所属哪个包， 格式：
  package 包名;

- 包的定义：
  通常由多个单词组成， 所有单词的字母小写， 单词与单词之间使用.隔开 ，一般命名为“com.公司名.项目名.模块名....”。

  规范由来：
  由于Java面向对象的特性，每名Java开发人员都可以编写属于自己的Java Package，为了保障每个Java Package命名的唯一性，在最新的Java编程规范中，要求开发人员在自己定义的包名前加上唯一的前缀。由于互联网上的域名称是不会重复的，所以多数开发人员采用自己公司在互联网上的域名称作为自己程序包的唯一前缀。例如：com.java.xxx

2.5.3 import 关键字

```
import 包名.类名;
```

2.6、权限修饰符

修饰符	类	包	子类	其他包
public	v	v	v	v
protected	v	v	v	×
default	v	v	×	×
private	v	×	×	×

2.7、main方法详解

main()方法一直写到了今天：
public static void main(String args[])
    以上的各个参数的含义如下：
        · public：表示公共的内容，可以被所有操作所调用
        · static：表示方法是静态的，可以由类名称直接调用。java StaticDemo09
        · void：表示没有任何的返回值操作
        · main：系统规定好的方法名称。如果main写错了或没有，会报错：NoSuchMethodError: main
        · String[] args：字符串数组，接收参数的

```
public class StaticDemo10{
    public static void main(String args[]){
        for(int i=0;i<args.length;i++){
            System.out.println(args[i]) ;
        }
    }
};
```

】所有的参数在执行类的时候以空格进行分割。
java StaticDemo10 1 2 3 4 5 6 7
    但是，如果现在我要输入的是以下几种参数“hello world”、“hello vince”、“hello mjw”。

因为以空格分割，所以以上的三组参数会当做六组参数输入，那么此时如果要想完成有空格的内容输入，则参数需要使用“”括起来。

```
java StaticDemo10 "hello world" "hello vince" "hello mjw"
```

2.8、单例设计模式

单例设计模式是我们学习的第一个设计模式，也是比较重要的一个设计模式，单例设计模式会伴随这你的开发生涯，不管你是初级程序员，还是以后晋级到高级程序员，你都会接触到单例设计模式，今天，我们就学习单例设计模式的两种实现方式。
单例设计模式：保证程序在内存中只有一个对象存在（被程序所共享）

单例设计模式的两种实现方式：
一、懒汉式：随着类的加载在内存中对象为null，当调用 getInstance 方法时才创建对象（延迟加载）
二、饿汉式：随着类的加载直接创建对象（推荐开发中使用）
单例设计模式的实现步骤：
1．保证一个类只有一个实例，实现方式：构造方法私有化
2．必须要自己创建这个实例，实现方式：在本类中维护一个本类对象（私有，静态）
3．必须向整个程序提供这个实例，实现方式：对外提供公共的访问方式（getInstance方法，静态）

懒汉式实现如下：
```
class Single{
    private Single(){}

    private static Single s1 = null;
    public static Single getInstance(){
        if(s1 == null){
            s1 = new Single();
        }
        return s1;
    }
}
```
饿汉式实现如下：
```
class Single2{
    private Single2(){}

    private static Single2 s = new Single2();

    public static Single getInstance(){
        return s;
    }
    void print(){
        System.out.println("Hello World!");
    }
}
```