Java面向对象编程2

- What?Why?How?

本章概述

- This关键字
- Static关键字
- 代码块
- Package
- Import
- 封装

this关键字

- this的作用:
  - this表示的是当前对象本身，
  - 更准确地说，this代表当前对象的一个引用。
- 普通方法中使用this。
  - 区分类成员属性和方法的形参.
  - 调用当前对象的其他方法（可以省略）
  - 位置：任意
- 构造方法中使用this。
  - 使用this来调用其它构造方法
  - 位置：必须是第一条语句
- this不能用于static方法。（讲完static，大家就知道为什么了！）

this 测试代码

```
public class TestThis {
        int a,b,c;
        TestThis(){
                System.out.println("正要new一个Hello对象");
        }

        TestThis(int a,int b){
                //Hello();     // //这样是无法调用构造方法的!
                this();   //调用无参的构造方法，并且必须位于第一行!

                a = a;//这里都是指的局部变量而不是成员变量
                this.a = a;//这样就区分了成员变量和局部变量.  这种情况占了this使用情况的大多数!
                this.b = b;
        }

        TestThis(int a,int b,int c){
                this(a,b);     //调用无参的构造方法，并且必须位于第一行!
                this.c = c;
        }


        void sing(){}

         void chifan(){
                this.sing();     //sing();
                System.out.println("你妈妈喊你回家吃饭！");
        }

        public static void main(String[] args){
                TestThis hi = new TestThis(2,3);
                hi.chifan();
        }
}
```

static 关键字

在类中，用static声明的**成员变量**为静态成员变量 ,或者叫做： 类属性，类变量.

它为该类的公用变量，属于类，被该类的所有实例共享，**在类被载入时被显式初始化，**

对于该类的所有对象来说，static成员变量只有一份。被该类的所有对象共享！！

可以使用"对象.类属性"来调用。不过，一般都是用“类名.类属性”

static变量置于方法区中！

用static声明的方法为**静态方法**

不需要对象，就可以调用(类名.方法名)

在调用该方法时，不会将对象的引用传递给它，所以**在static方法中不可访问非static的成员**。

静态方法不能以任何方式引用this和super关键字

Static示例代码

```
public class TestStatic {
        int a;
        static int width;

        static void gg(){
                System.out.println("gg");
        }
        void tt(){
                System.out.println("tt");
        }

        public static void main(String[] args){
                TestStatic hi = new TestStatic();
                TestStatic.width = 2;
                TestStatic.gg();  //gg();
                hi.gg();    //通过引用也可以访问static变量或static方法。不过，一般还是使用类名.static成员名来访问。
                gg();
        }
}
```

static关键字

- 使用static声明的成员变量称为静态变量，
- 使用static声明的方法称为静态方法
- 静态变量与静态方法又称为**类变量**和**类方法**

```
//使用static统计在类中一共产生多个对象
public class StaticDemo //声明类
{
    static int count;//声明静态属性
    public StaticDemo(){//无参构造方法
        count++;
        System.out.println("创建了"+count+"个对象");
    }
    public static void main(String[] args)
    {
        new StaticDemo();//创建匿名对象
        new StaticDemo();//创建匿名对象
        new StaticDemo();//创建匿名对象
    }
}
```

static关键字

- **静态属性的访问形式**
  - (1)对象名.属性
  - (2)类名.属性
- **静态方法**
  - 访问修饰符 **static** 返回值类型 方法名(){}
- **访问形式**
  - (1)对象名.方法名();
  - (2)类名.方法名();

常见错误1

```
public void showInfo(){
      System.out.println("姓名:"+this.name+"\t年龄:"+this.age+"\t城
市:"+this.country);
      }


public static void welcome(){
          this.showInfo();//调用本类的非静态方法
          System.out.println("欢迎大家来马士兵教育学习.....");
}
```

常见错误2

- 请指出下面代码的错误

```
class Dog {
    private String name = "旺财"; // 昵称
    private int health = 100;    // 健康值
    private int love = 0;        // 亲密度
    public void play(int n) {
        static int localv=5;
        health = health - n;
        System.out.println(name+" "+localv+" "+health+" "+love);
    }
    public static void main(String[] args) {
        Dog d=new Dog();
        d.play(5);
    }
}
```

在方法里不可以定义static变量

小结

- **static修饰与非static修饰的区别**

	static、非private修饰	非static、private修饰
属性	类属性、类变量	实例属性、实例变量
方法	类方法	实例方法
调用方式	类名.属性 类名.方法() 对象.属性 对象.方法()	对象.属性 对象.方法()
归属	类	单个对象

代码块

- **概念:**使用"{}"括起来的一段代码
- **分类：**根据位置可分类
- 普通代码块→直接在方法或语句中定义的代码块
- 构造代码块→直接写在类中的代码块
- 静态代码块→使用static声明的代码块
- 同步代码块→**多线程的时候会学到**

静态初始化块

- 如果希望加载后，对整个类进行某些初始化操作，可以使用static初始化块。
- 类第一次被载入时先执行static代码块；类多次载入时，static代码块只执行一次；Static经常用来进行static变量的初始化。
- 是在类初始化时执行，不是在创建对象时执行。
- 静态初始化块中不能访问非static成员。

```
public class TestStaticBlock {

        static {
                System.out.println("此处，可执行类的初始化工作！");
        }

        public static void main(String[] args) {
                System.out.println("main方法中的第一句");
        }
}
```

pakage

- 为什么需要package?
  - 为了解决类之间的重名问题。
  - 为了便于管理类：合适的类位于合适的包！
- package怎么用?
  - 通常是类的第一句非注释性语句。
  - 包名：域名倒着写即可，再加上模块名，并与内部管理类。
- 注意事项：
  - 写项目时都要加包，不要使用默认包。
  - com.gao和com.gao.car，这两个包没有包含关系，是两个完全独立的包。只是逻辑上看起来后者是前者的一部分。

JDK中的主要包

java.lang

包含一些Java语言的核心类，如String、Math、Integer、System和Thread，提供常用功能。

java.awt

包含了构成抽象窗口工具集（abstract window toolkits）的多个类，这些类被用来构建和管理应用程序的图形用户界面(GUI)。

java.net

包含执行与网络相关的操作的类。

java.io

包含能提供多种输入/输出功能的类。

java.util

包含一些实用工具类，如定义系统特性、使用与日期日历相关的函数。

Import

为什么需要import?

如果不适用import，我们如果用到其他包的类时，只能这么写：java.util.Date，代码量太大，不利于编写和维护。通过import可以导入其他包下面的类，从而可以在本类中直接通过类名来调用。

import怎么使用?

```
import  java.util.Date;
import  java.util.*;   //导入该包下所有的类。会降低编译速度，但不会降低运行速度。
```

注意要点：

java会默认导入java.lang包下所有的类，因此这些类我们可以直接使用。

如果导入两个同名的类，只能用包名+类名来显示调用相关类：

```
java.util.Date date  = new  java.util.Date();
```

import static

- 静态导入的作用：用于导入指定类的静态属性
- JDK5.0后增加！
- 如何使用：
  - **import static** java.lang.Math.*;//导入Math类的所有静态属性
  - **import static** java.lang.Math.PI;//导入Math类的PI属性
  - 然后，我们可以在程序中直接使用：System.*out*.println(*PI*);

为什么要使用封装

- 下面代码有什么缺陷?

```
Dog d = new Dog();
d.health = -1000;
```

属性随意访问，不合理的赋值

- 如何解决上面设计的缺陷?

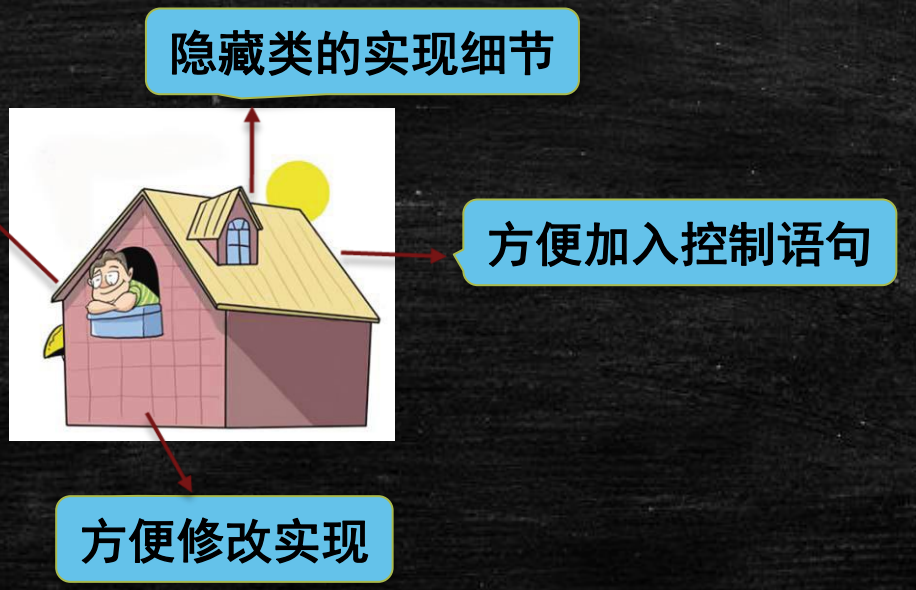
使用封装

什么是封装

- 面向对象三大特征之一 ——封装
  - 封装的概念

封装：将类的某些信息隐藏在类内部，不允许外部程序直接访问，而是通过该类提供的方法来实现对隐藏信息的操作和访问

  - 封装的好处



隐藏/封装(encapsulation)

为什么需要封装？封装的作用和含义？

我要看电视，只需要按一下开关和换台就可以了。有必要了解电视机内部的结构吗？有必要碰碰显像管吗？

我要开车，....

隐藏对象内部的复杂性，只对外公开简单的接口。便于外界调用，从而提高系统的可扩展性、可维护性。

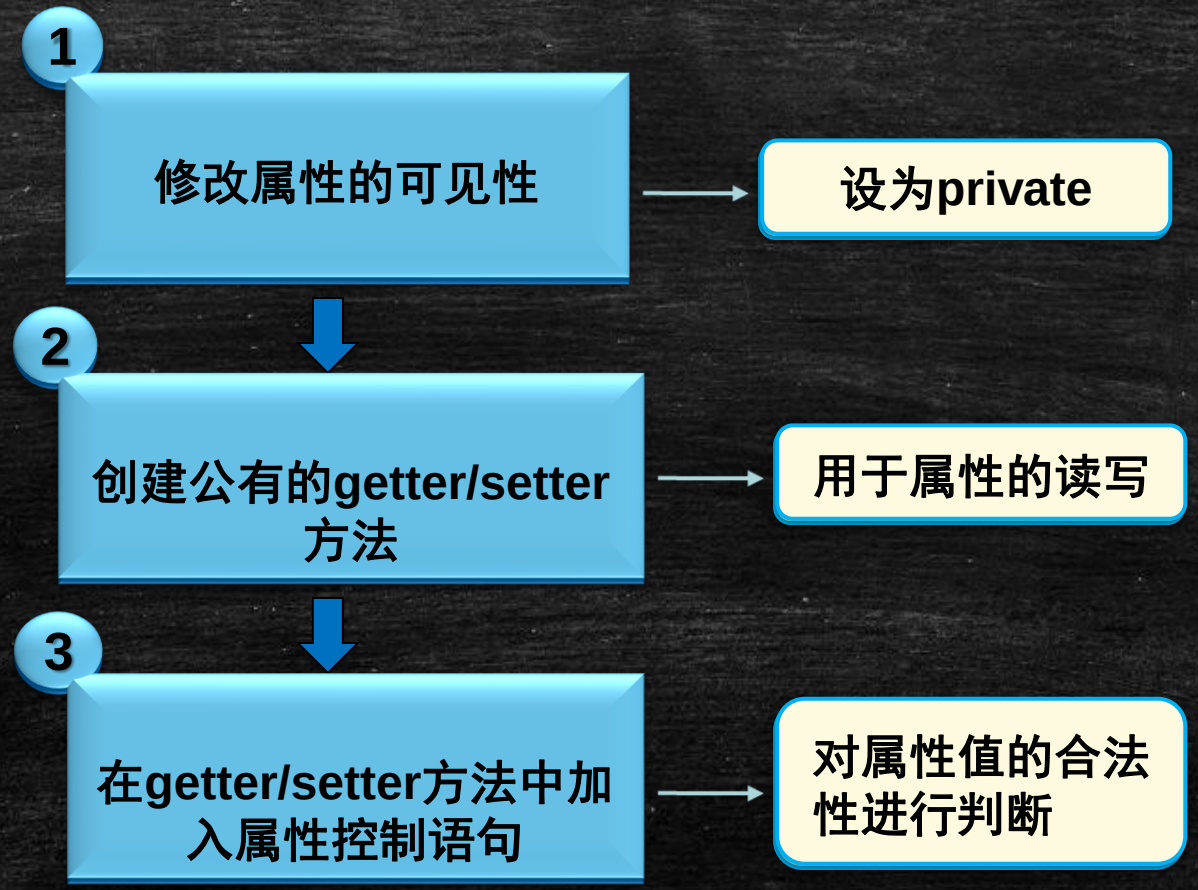
我们程序设计要追求“**高内聚，低耦合**”。

高内聚就是类的内部数据操作细节自己完成，不允许外部干涉；

低耦合：仅暴露少量的方法给外部使用。

如何使用封装

- 封装的步骤



小结

```
class Dog {
    private String name = "旺财"; // 昵称
    private int health = 100; // 健康值
    private int love = 0;   // 亲密度
    private String strain = "拉布拉多犬"; // 品种
    public int getHealth() {
        return health;
    }
    public void setHealth (int health) {
        if (health > 100 || health < 0) {
            this.health = 40;
            System.out.println("健康值应该在0和100之间，默认值是40");
        } else
            this.health  =  health;
    }
    // 其它getter/setter方法
}
```

this代表当前对象

Dog
- name:String - health:int - love:int - strain:String
+ print():void + setHealth():void + getHealth():String … …

面向对象的三大特征

- 继承 inheritance
  - 子类 父类
  - 子类可以从父类继承属性和方法
  - 子类可以提供自己单独的属性和方法
- 封装/隐藏encapsulation
  - 对外隐藏某些属性和方法
  - 对外公开某些属性和方法
- 多态 polymorphism
  - 为了适应需求的多种变化，使代码变得更加通用！
- 面向过程只有封装性（功能的封装，而没有数据的封装），没有继承和多态

使用访问控制符，实现封装

- 成员（成员变量或成员方法）访问权限共有四种：
  - public 公共的
    - 可以被项目中所有的类访问。(项目可见性)
  - protected 受保护的
    - 可以被这个类本身访问；同一个包中的所有其他的类访问；被它的子类（同一个包以及不同包中的子类）访问
  - default / friendly 默认的/友好的（包可见性）
    - 被这个类本身访问；被同一个包中的类访问。
  - private 私有的
    - 只能被这个类本身访问。（类可见性）
- 类的访问权限只有两种
  - public 公共的
    - 可被同一项目中所有的类访问。(必须与文件名同名)
  - default / friendly 默认的/友好的
    - 可被同一个包中的类访问。

使用访问控制符，实现封装

	同一个类	同一个包中	子类	所有类
private	*			
default	*	*		
protected	*	*	*	
public	*	*	*	*

封装要点：

- 类的属性的处理：
  1. 一般使用private. （除非本属性确定会让子类继承）
  2. 提供相应的get/set方法来访问相关属性. 这些方法通常是public，从而提供对属性的读取操作。 （注意：boolean变量的get方法是用：is开头!）
- 一些只用于本类的辅助性方法可以用private，
  希望其他类调用的方法用public

总结

- 方法调用中的参数传递（重中之重）
  - 基本数据类型的参数传递：不能改变参数的值
  - 引用数据类型的参数传递：不能改变参数的值
- this
  - This代表当前对象自身的引用（必须new）
  - This可以修饰属性，区别成员变量和局部变量
  - This修饰方法
  - This修饰构造方法（必须是第一条语句）
- static
  - static变量：只有一份，属于类，可以类名. Static变量
  - static方法: 类名. Static方法，不能出现this和super
  - static代码块：只执行一次，最早执行的（类第一次调用）
- package import
  - 包：作用
  - 导入：import com.bjsxt.oop.*;
  - 静态导入： import static java.lang.Math.PI;

上机练习1——设计Dog和Penguin类

- 需求说明：
  - 运用面向对象思想抽象出Dog类和Penguin类，画出对应类图

类型	属性				行为
狗	昵称	健康值	亲密度	品种	输出信息
企鹅	昵称	健康值	亲密度	性别	输出信息

  - 根据类图编写Dog类和Penguin类
  - 添加默认构造方法

上机练习2——打印Dog信息2-1

- 需求说明：
  - 根据控制台提示信息选择领养宠物（狗），
    - 输入昵称、品种、健康值
    - 打印宠物信息
  - 要保证健康值的有效性（在1到100之间）

上机练习3——Dog类的带参构造方法

- 需求说明：
  - 增加带参构造方法

```
Dog(String name，String strain)
```

  - 修改Test类，使用带参构造方法创建对象

上机练习4——操作企鹅性别属性

- 需求说明：
  - 给Penguin类提供SEX_MALE和SEX_FEMALE两个静态常量，分别取值“Q仔”或“Q妹”
  - 修改Test类，使用静态常量对性别进行赋值
  - 修改企鹅的性别只能取值“雄”或“雌”，通过修改静态变量实现该需求

完时间：15分钟