day07----封装

封装

封装：是指隐藏对象的属性和实现细节，仅对外提供公共访问方式。

所有逻辑操作都要在方法中

private：表示私有的意义。可用于修饰成员变量和成员方法。

被它修饰完毕后的内容，在其他类中是不能直接访问的。

私有的东西只能在自己的类中访问

private 只能修饰成员变量和成员方法

private 仅仅是封装的一种体现。因为类、函数等其实也是一个封装的体现。

```
class Student
{
    String name;
    private int age;
    public void setAge(int a)
    {
        if(a<0 || a>200)    System.out.println("年龄有问题");
        else age = a;
    }

    public int getAge()
    {    return age;    }

    public void show()
    {     System.out.println(name+"***"+age);    }
}
```

封装的思想：

1、因为逻辑判断只能在方法中，通过设置方法对数据进行方法校验。

2、封装隐藏了属性和实现细节，就只对外提供一个接口

3、函数也是封装的一种体现

一个方法可以访问所属类的所有对象的私有数据

```
class Employee {
	private String name;
	public void setName(String name) {		this.name = name;	}
	public String getName()	{		return name;	}
}

public class test {
 public static void main(String[] args) {
	Employee harry = new Employee();
	harry.setName("herry");
	Employee boss = new Employee();
	boss.setName("boss");
	System.out.println(harry.getName().equals(boss.getName()));

 }

}
```

构造方法

构造方法格式及特点：

A:方法名必须和类名一致

B:没有返回值类型　（void 也不行）

C:没有具体的返回值

作用：　给对象进行初始化。

注意事项：

A:如果没，则用系统默认
　　public Person(){}
B:若有则系统不再给默认构造方法
　　推荐做法：请永远手动给出无参构造。
C:构造方法重载　(接受参数不同而已)

给成员变量赋值：

A:通过 set 方法。推荐使用。

B:通过构造方法。

C:定义时手动赋值

```
public Person(String n)    //不要加返回值类型
{
    name = n;
}
```

this

this 关键字： this 代表其所在方法所属对象的引用。

哪个对象调用方法，在方法内部就有一个隐含的 this 代表这个对象。

this 只能用在方法中　谁调用，this 就代表谁。

应用场景：

用于解决局部变量隐藏成员变量值的问题。

因为调用的时候是就近原则

局部变量是定义在方法中的变量，main 方法也是方法

局部变量可见范围内不能在定义局部变量

代码块

代码块：就是由{}括起来的代码。

局部代码块：定义在方法中的代码块。作用是：让变量尽可能早的从内存中消失，提高效率。

构造代码块：定义在方法外的代码块。作用是：把所有构造方法中的共同的内容定义在构造代码块中。

局部代码块中不能再定义相同变量

```
class Dog{
        int age;
        //age = 1;    //Error
        {
          System.out.println(age);//out:0
        }
        Dog (int age){
          this.age = age;
          System.out.println(age);
        }
}

public class test2 {
      public static void main(String[] args) {
        int age = 1;
        {
            //int age = 2;//Error  局部代码块内不能定义已有的变量
            int a = 2;
            System.out.println(a);    //out:2
        }
        //System.out.println(a); //Error,出了代码块就释放了 a
        Dog d = new Dog(3);
      }
}
```

static

可以修饰成员变量和成员方法

类的组成：

成员变量 -- 外在描述
构造方法 -- 对数据进行初始化
成员方法 -- 一些功能

static 修饰成员变量和成员方法。

特点：

A:随着类的加载而加载

B:优先于对象存在

C:为什么要使用静态呢?

如果某个内容是被所有对象所共享，那么，该内容就应该用静态修饰。

没有被静态修饰的内容，其实是属于对象的特殊描述。

D:可以直接被类名调用

静态方法中是没有 this 关键字的，因为静态方法随着类的加载而加载。所以它会优先于对象（使用 this 的）存在，若用了则会出现找不到对象，也就是会出错。故静态方法只能访问静态成员。

静态的调用（方法、成员）：类名调用或对象调用都可以

```
class Dog{
 int a;
 static int b;
 public static void print(){
      System.out.println(a);   //error，不能在静态方法中访问实例域
      System.out.println(b);   //ok
 }
}
```

如果不导包，则会默认在文件当前位子找相关的类文件

制作帮助文档

查看 API

字段----成员变量　　（即属性）

如果没有构造方法，则说明它的构造方式用了 private 修饰了，只能通过类名调用

如果有构造方法，则非静态的要创建对象，再使用对象调用

而静态的也可以通过类名，也可以通过对象调用

Math

在 java.lang.Math 中（只有 lang 包不需要导）

1、

```
static double                random()
```

返回一个大于等于 0，小于 1 的 double 数

2、abs（--）　返回绝对值

3、

4、max (a,b)　---返回最大值

min（a,b）---返回最小值

产生 1-100 之间的随机数：

int number = (int)(Math.random()*100)+1;