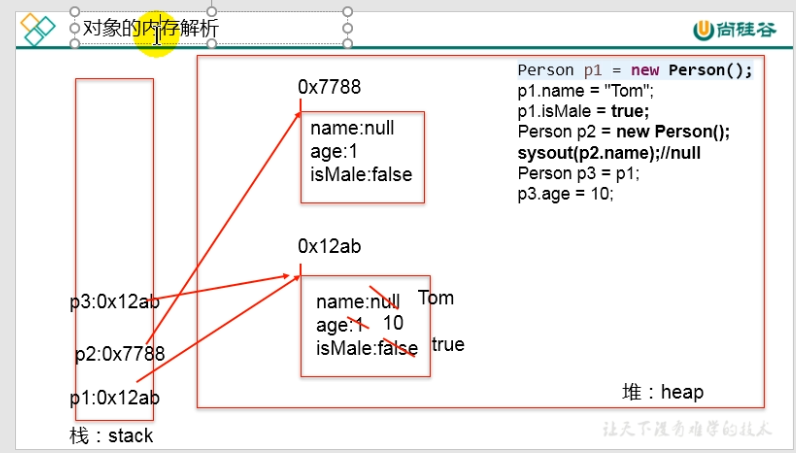
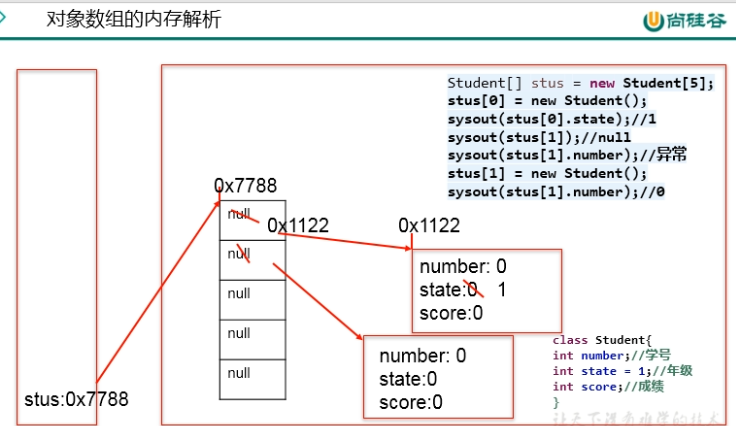
## 对象的内存解析







## 属性(非static)和局部变量的区别

1、定义位置不同

2、关于权限修饰符的不同

3、默认初始化值的不同

属性有初始值

局部变量没有初始值

4、内存中位置的不同

局部变量--栈

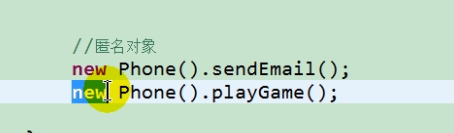
属性--堆空间中

## 理解万事万物皆对象

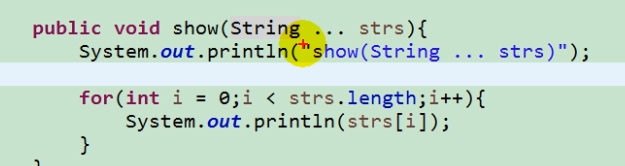
每一个抽象的东西都就可以具体为对象（文件，网络地址，数据库表，数据库连接）

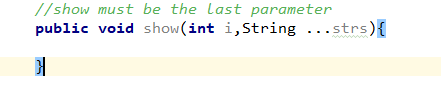
## 匿名对象

创建对象时没有显式的赋值给一个变量名，即为匿名对象

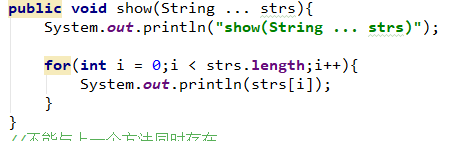


## 可变形参

1 可变个数形参的格式：数据类型 ... 变量名  
2 当调用可变个数形参的方法时，传入的参数个数可以是：0个，1个,2个，。。。  
3 可变个数形参的方法与本类中方法名相同，形参不同的方法之间构成重载  
4 可变个数形参的方法与本类中方法名相同，形参类型也相同的数组之间不构重载。换句话说，**二者不能共存**。  
5 可变个数形参在方法的形参中，**必须声明在末尾**

  
6 可变个数形参在方法的形参中,最多只能声明**一个**可变形参。

使用方法



## 变量的值传递

如果变量是**基本数据类型**，此时赋值的式变量所保存的**数据值**

如果变量是**引用数据类型**，此时复制的式变量所保存的**数据的地址值**

## JavaBean

一种java 语言编写的可重用组件

## 代码块 （初始化块）

1. 代码块的作用：用来初始化类、对象  
    2. 代码块如果有修饰的话，只能使用static.  
    3. 分类：静态代码块 vs 非静态代码块  
      
    4. 静态代码块  
    >内部可以有输出语句  
    >随着类的加载而执行,而且只执行一次  
    >作用：初始化类的信息  
    >如果一个类中定义了多个静态代码块，则按照声明的先后顺序执行  
    >静态代码块的执行要优先于非静态代码块的执行  
    >静态代码块内只能调用静态的属性、静态的方法，不能调用非静态的结构  
      
    5. 非静态代码块  
    >内部可以有输出语句  
    >随着对象的创建而执行  
    >每创建一个对象，就执行一次非静态代码块  
    >作用：可以在创建对象时，对对象的属性等进行初始化  
    >如果一个类中定义了多个非静态代码块，则按照声明的先后顺序执行  
    >非静态代码块内可以调用静态的属性、静态的方法，或非静态的属性、非静态的方法
2. 属性赋值顺序

对属性可以赋值的位置：  
 ①默认初始化  
 ②显式初始化/⑤在代码块中赋值 （谁在下面谁后执行）  
 ③构造器中初始化  
 ④有了对象以后，可以通过"对象.属性"或"对象.方法"的方式，进行赋值  
  
   
 执行的先后顺序：① - （② / ⑤） - ③ - ④

## 九、代码执行顺序

父类静态代码块

子类静态代码块

父类普通代码块

父类构造方法

子类普通代码块

子类构造方法

父类 >子类

静态块>非静态块

块>构造器

## 十、抽象类

### 1、abstract修饰类：抽象类

1、此类**不能实例化，相当于是一个模板**

2、抽象类中一定有构造器，便于子类实例化时调用（涉及：子类对象实例化的全过程）

3、开发中，都会提供抽象类的子类，让子类对象实例化，完成相关的操作

### 2、abstract修饰方法：抽象方法

1、抽象方法只有方法的声明，没有方法体

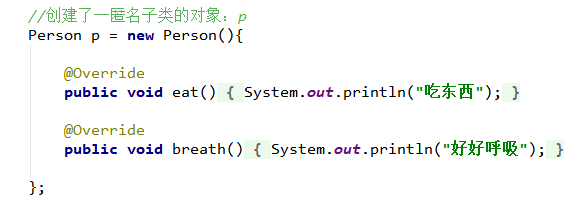
2、包含**抽象方法的类，一定是一个抽象类**。反之，抽象类中可以没有抽象方法的。

3、若子类重写了父类中的所有的抽象方法后，此子类方可实例化

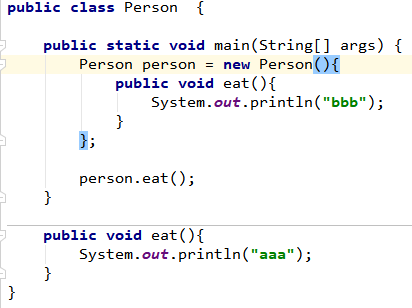
若子类没有重写父类中的所有的抽象方法，则此子类也是一个抽象类，需要使用abstract修饰

### 3、abstract 使用的注意点1、abstract不能用来修饰：属性、构造器等结构 2. abstract不能用来修饰私有方法、静态方法、final的方法、final的类

### 4、抽象类的匿名子类

如图Person为抽象类，创建了Person的匿名子类

非抽象类的匿名子类对象 输出bbb



## 十一、接口

### 1、接口的概念

子父类继承关系是、 **你是什么**、 你是父类的子类 、 人是生物

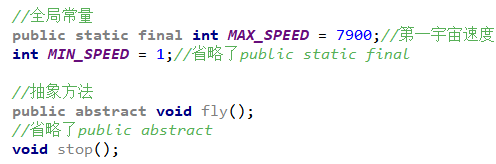
接口实现的关系是 、 **你能干什么**、你能使用接口所代表的能力、 人能走

### 2、如何定义接口：定义接口中的成员

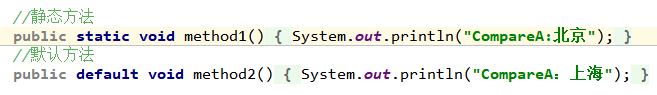
3.1 JDK7及以前：只能定义全局常量和抽象方法

>全局常量：public static final的.但是书写时，可以省略不写

>抽象方法：public abstract的



3.2 JDK8：除了定义全局常量和抽象方法之外，还可以定义静态方法、 默认方法



接口中不能定义构造器的！意味着接口不可以实例化

### 3、注意事项

1、. Java开发中，接口通过让类去实现(implements)的方式来使用.

如果实现类覆盖了接口中的所有抽象方法，则此实现类就可以实例化

如果实现类**没有**覆盖接口中所有的抽象方法，则此实现类**仍为一个抽象类**

2、 Java类可以实现多个接口 --->**弥补了Java单继承性的局限性**

3、 接口与接口之间可以继承，而且可以**多继承**

4、接口的具体使用，体现多态性

5、 接口，实际上可以看做是一种**规范**

## 内部类

### 1、什么是内部类

Java中允许将一个类A声明在另一个类B中，则类A就是内部类，类B称为外部类

### 2、内部类的分类

**成员内部类（静态、非静态）** vs **局部内部类(方法内、代码块内、构造器内)**

### 3、成员内部类说明

一方面，**作为外部类的成员**：

>调用外部类的结构

>可以被static修饰

>可以被4种不同的权限修饰

另一方面，**作为一个类**：

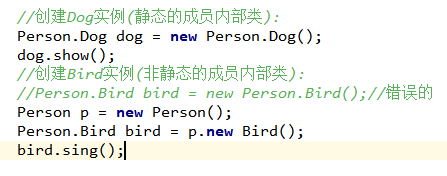
> 类内可以定义属性、方法、构造器等

> 可以被final修饰，表示此类不能被继承。言外之意，不使用final，就可以被继承

> 可以被abstract修饰

### 4、关注如下的3个问题

4.1 如何实例化成员内部类的对象



4.2 如何在成员内部类中区分调用外部类的结构

4.3 开发中局部内部类的使用 见《InnerClassTest1.java》

