# Day07

## 对象内存分析

|  |
| --- |
| public class Student{  ……  public static void main(String[] args){  Student stu = new Student(“张三”, 18, ‘男’, “精英班”, 10);  stu.displayInfo();  Student stu2 = new Student(“李四”, 23, ‘女’, “预热班”, 18);  stu2.displayInfo();  }  } |

## this关键字

|  |
| --- |
| 1、每个类的每个非静态方法(没有被static修饰)都会隐含一个this引用名称，它指向调用这个方法的对象。  2、当在方法中使用本类的属性时，都会隐含地使用this名称，当然也可以明确指定。  this可以看作是一个变量，它的值就是当前对象的引用。 |

### this用法一：调用本类成员变量和成员方法

|  |
| --- |
| 当类中某个非静态方法的名跟类的某个成员变量名相同时，为了避免参数的作用范围覆盖了成员变量的作用范围，必须明确地使用this关键字来指定成员变量  public class Employee {  private String name; //姓名  private int age; //年龄  private double salary; //薪水  public Employee(String name, int age, double salary){ //构造方法  this.name = name; //左边的name是成员变量name，右边的name是局部变量  this.age = age;  this.salary = salary;  }  } |

### this用法二：调用本类构造方法

|  |
| --- |
| 如果某个构造方法的第一条语句具有形式this(...)，那么这个构造方法将调用本类中的其他构造方法。  public class Employee {  private String name; //姓名  private int age; //年龄  private double salary; //薪水  public Employee(String name, int age, double salary){ //构造方法1  this.name = name;  this.age = age;  this.salary = salary;  }  public Employee(){ //构造方法2  this(“无名”, 18, 800.0); //调用到了构造方法1  }  }  注意：  **1、调用本类其他构造方法的语句，即this(参数)语句，必须放在的该构造方法的第一行。** |

### this用法三：可以作为参数传递

|  |
| --- |
| this可以作为参数，把当前对象的地址传递。 |

## java方法中的参数传递

|  |
| --- |
| java方法中传递参数，不管任何数据类型，均是**值传递：**即传递的是该变量所存储的值。   1. 基本数据类型传递的就是变量存储的值。 2. 引用数据类型。由于引用数据类型的变量存储的是对象的地址值，那么传递出去的也是对象的地址值，本质上也是值传递。 |

|  |
| --- |
| public class PassValue{  public static void main(String[] args){  PassValue pv = new PassValue();  int x = 5;  System.out.println(“方法调用之前x==” + x);  pv.change(x);  System.out.println(“方法调用之后x==” + x);  }  public void change(int x){  x = 100;  System.out.println(“方法中x==” + x);  }  } |

|  |
| --- |
| public class PassValue2{  private int x ;  public static void main(String[] args){  PassValue2 obj = new Passvalue2();  obj.x = 5;  System.out.println(“方法调用之前obj.x==” + obj.x);  obj.change(obj);  System.out.println(“方法调用之后obj.x==” + obj.x);  }  public void change(PassValue2 obj){  obj.x = 100;  System.out.println(“方法中obj.x==” + obj.x);  }  } |

## 面向对象的特征之一-------封装性(Encapsulation)

|  |
| --- |
| 在java中类的封装性主要体现不能让外部随意修改一个成员变量的值，否则会引起很多逻辑上的错误，以及一些不可预知的错误。  在定义一个类的成员（包括变量和方法），使用private关键字说明这个成员的访问权限，只能被这个类的其他成员方法调用，而不能被其他的类中的方法所调用；  为实现封装性，常将类的成员变量声明为private,再通过public的方法来对这个变量进行访问。对一个变量的操作，一般都有 读取和赋值操作，我们一般定义两个方法来实现这两种操作，即：getXxx()与setXxx();  一个类就是一个模块，我们应该让模块仅仅公开必须要让外界知道的内容，而隐藏其他的一切内容。再进行程序设计时，应尽量避免一个模块直接修改或操作另一个模块的数据，模块设计追求强内聚（许多功能尽量在类的内部独立完成，不让外面干预），弱耦合（提供给外部尽量少的方法调用）。 |

## static关键字

|  |
| --- |
| 1、在类中，用static声明的成员变量为静态成员变量，在类第一次使用时被初始化，对于该类的所有对象来说，**static成员变量只有一份**它是该类的公用变量**。** 也叫类变量  2、用static声明的方法为静态方法，该方法独立于类的实例，所以也叫类方法。  3、静态方法中只能调用本类中其他的静态成员(变量和方法)。  4、静态方法中不能使用**this**和**super**关键字。  5、静态成员(类成员)**可以**通过类名直接访问，也可以通过类的对象去访问。**但是强烈建议通过类名访问静态成员。** |

### 静态成员变量

|  |
| --- |
| 用static修饰的成员变量，叫静态成员变量。(静态域、静态属性)。  例如： public static int a; //声明一个静态的整型成员变量。  注意：static只能修饰成员变量，不能修饰局部变量。 |

### 静态方法

|  |
| --- |
| 用static修饰的方法，叫静态方法。  例如：public static int fun(){  }  不用static修饰的方法，在很多地方也称之为非静态方法、实例方法。 |

### 静态代码块和构造代码块(作用同构造方法一样)

|  |
| --- |
| 类中的可以写哪些成员   1. 成员变量 2. 方法 3. **代码块 直接用一对大括号括起来的一段代码。**   **用static修饰的代码块称之为静态代码块**  **例如：**  **public** **class** Test {  **public** **int** a;    { // 构造代码块  System.*out*.println("我构造代码块");  }  **static**{ //用static修饰的代码块称之为静态代码块  System.*out*.println("我是静态代码块");  }    **public** **void** fun(){    }  }  **注意：**   1. **静态变量只有在第一次加载到内存的时候初始化一次。** 2. **静态代码块只有在类第一次加载到内存的时候执行一次。** 3. **构造代码块在每次new对象的时候都会执行一次。** |

## 包(package)

|  |
| --- |
| 1、为了便于管理大型软件系统中数目众多的类，解决类命名冲突的问题，Java引入了包（package）。  2、在使用许多类时，类的名称很难决定。有时需要使用与其他类相同的名称。包基本上隐藏了类并避免了名称上的冲突。  3、用package来声明包，package语句必须是java源文件中的第一条语句。(若无这条语句，则放置在无名包下)  4、在package语句中，用"."来指明包（目录）的层次。包对应着文件系统的目录层次结构。  如：package com.qianfeng.java; 编译后对应的类文件位于com\qianfeng\java目录下。  注意：给包命名的时候，全部用小写字母，每个层次并且不能使用数字开头。建议用公司域名反过来写，一般不会出现包名重复的情况  例如：com.qiangfeng.day07 |

### 在类中声明包

|  |
| --- |
| package com.demo.test; //给本类声明包  public class Employee {  private String name; //姓名  private int age; //年龄  private double salary; //薪水  public Employee(String name, int age, double salary){ //构造方法  this.name = name;  this.age = age;  this.salary = salary;  }  ……  }  注意：   1. **使用package声明包时必须放在类的首行。** |

### 导入包

|  |
| --- |
| import语句  为使用定义在不同包中的JAVA类，需要import语句来引入所需要的类。  语法格式：  import package1[.package2…](classname|\*);  例如：import com.qianfeng.day06.Test; //导入包com.qianfeng.day06下的Test类。  import com.qianfeng.day06.\*; //导入包com.qianfeng.day06下的所有类。  **注意：如果要使用的类与当前类属于同一个包，则无需导入**。 |

### 编译并运行声明包的类

|  |
| --- |
| 在当前目录下生成带包结构的字节码  javac –d . Employee.java  在指定目录下生成带包结构的字节码  javac –d D:\share Employee.java  运行：  java 包名.类名  包名.类名 叫做类的全限定名 |