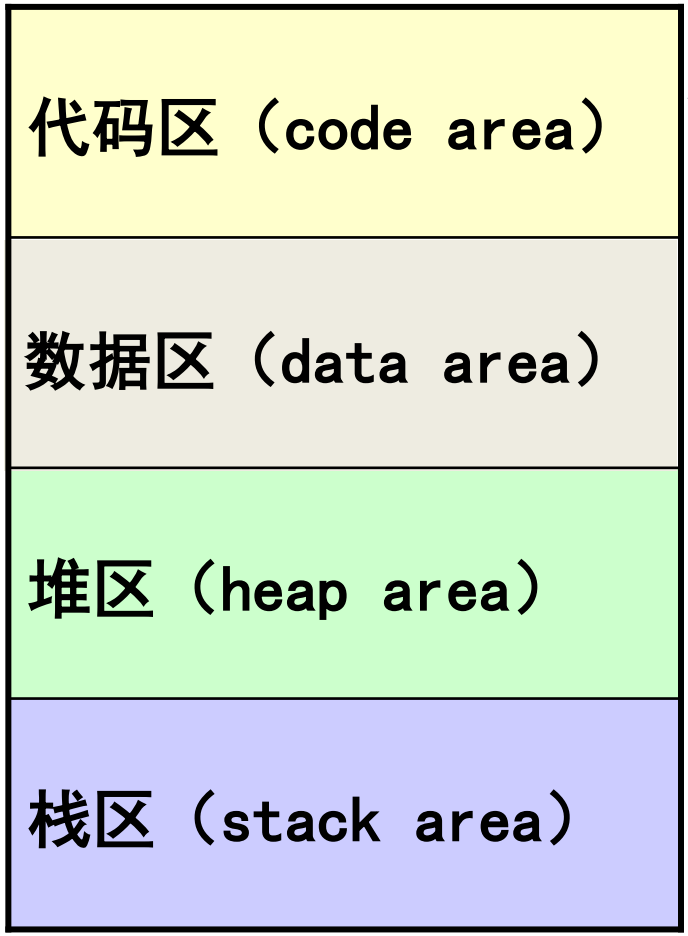
Lesson3
认识对象：封装数据为类

主讲老师：申雪萍

主要内容

- 类的设计原则
- Java程序的基本结构
- 类的定义
  - 成员变量的定义
  - 成员方法的定义
- 对象的生成、使用和清除
  - 默认构造函数（不带参数构造函数）
  - 带参数构造函数
  - 垃圾内存自动回收机制
- 匿名对象
- 进一步理解引用类型变量
- **类的静态属性和静态方法（重点、难点）**

程序运行时的内存占用



存放程序的代码部分

存放程序的全局数据和静态数据

存放程序动态申请的数据

存放程序的局部数据和参数

问题

- 问题一：不论产生多少个对象，或不存在任何对象的情况下，某些特定数据的存储空间都只有一份；
  - 例如：统计出从Person共new出多少个对象？
- 问题二：某些数据或者函数不要和class object绑在一起。

解决方案

- 通过**关键字static**，便可以处理这两种情况。

- Java中把这种*代表类范围信息*的变量用关键字static修饰。

- *类变量和类方法*：当你将某个数据成员或某个成员函数声明为static时，它就不再局限于所属的class object上。

Static的使用场合

一. 用static修饰的属性（变量）称为**静态属性，又叫类变量；**

二. 用static修饰的方法称为**静态方法，又叫类方法（静态方法里,无this）；**

三. 可以用来修饰初始化语句块，这样的语句块常称为静态初始化语句块（要跟非静态初始化语句块区分开来）

案例(统计出从Person共new出多少个对象？)

- 理解并掌握Static关键字的用法
- 类变量概念
- 类方法概念
- 应用场合：当你将某个数据成员或某个成员函数声明为static时，它就不再局限于所属的class object上。

案例(统计出从Person共new出多少个对象？)

```
package com.dal.staticEx;

class Person {
    String name;//instance variable
    String sex;//instance variable
    int age;  //instance variable
    private static int count;  //类变量class variable 在全局分配内存
    public static int getCount() {  //类方法 class method
        return count;
    }
    public Person(String n, String s, int a) {//constructor
        name = n;
        sex = s;
        age = a;
        count++;
    }
    public String toString() {//instance method
        String s = "姓名：" + name + "," + "性别：" + sex + "," + "年龄：" + age;
        return s;
    }
}
```

案例(统计出从Person共new出多少个对象？)

```
package com.dal.staticEx;

public class TestPerson3 {
    public static void main(String[] args) {
        Person p1 = new Person("张三", "男", 20);
        System.out.print("count=" + p1.getCount() + "\t");//1
        System.out.print("count=" + Person.getCount() + "\n");//1
        Person p2 = new Person("Tom", "M", 50);
        System.out.print("count=" + p2.getCount() + "\t");//2
        System.out.print("count=" + Person.getCount() + "\n");//2
        Person p3 = new Person("Mary", "F", 10);
        System.out.print("count=" + p3.getCount() + "\n");//3
        System.out.println("通过类名和不同对象名访问静态变量count:");
        System.out.print("count=" + Person.getCount() + "\n");//3
        System.out.print("count=" + p1.getCount() + "\t");//3
        System.out.print("count=" + p2.getCount() + "\t");//3
        System.out.print("count=" + p3.getCount() + "\n");//3
    }
}
```

案例：TestPerson.java

- 重点：
  - 类变量和类方法
  - 实例变量和方法

TestPerson. java

```
class Student {
    static int num=10;   //类变量，保存在全局数据区，所有的对象共享
    int number;          //实例变量，保存在heap，每个对象有一份独立的拷贝
    int add1() {//add1()方法退出时，内存中的值可以保留
        num = num + 1;
        return num;
    }
    int add2() {  //add2()方法退出时，内存中的值不能保留
        number = number + 1;
        return number;
    }
}
```

TestPerson. java

即使没有创建该类的具体对象，
类中的static类成员也会存在

```
public class StuDemo
{
    public static void main(Strin args[ ])
    {
        System.out.println(Student.num);
        Student zhang=new Student( );//
        Student wang=new Student( );
        Student jiang=new Student( );
        System.out.println("zhang.num="+zhang.add1( ));//1
        System.out.println("zhang.number="+zhang.add2( ));//1
        System.out.println("wang.num="+wang.add1( ));//2
        System.out.println("wang.number="+wang.add2( ));//1
        System.out.println("jiang.num="+jiang.add1( ));//3
        System.out.println("jiang.number="+jiang.add2( ));//
    }
}
```

```
10
zhang.num=11
zhang.number=1
wang.num=12
wang.number=1
jiang.num=13
jiang.number=1
```

类的静态属性和静态方法

- 即使没有创建该类的具体对象，类中的static类成员也会存在，这时可以通过：
  1. 类名.静态变量
  2. 类名.静态方法

静态方法与非静态方法的区别

- 两者本质上的区别是：静态方法是在类中使用staitc修饰的方法，**在类定义的时候已经被装载和分配。**而非静态方法是不加static关键字的方法，在类定义时没有占用内存，**只有在类被实例化成对象时，对象调用该方法才被分配内存。**
- **静态方法中只能直接调用静态成员或者方法，不能调用非静态方法或者非静态成员，**而非静态方法既可以调用静态成员或者方法又可以调用其他的非静态成员或者方法。

静态方法不能直接引用非静态方法

```java
class Test{
    public int sum(int a,int b){//非静态方法
      return a+b;
    }
    public static void main(String[] args){
        int  result=sum(1,2);//静态方法调用非静态方法
        System.out.println("result="+result);
    }
}
```

```
C:\Users\Zu>cd desktop

C:\Users\Zu\Desktop>javac test.java
test.java:15: 错误: 无法从静态上下文中引用非静态 方法 sum(int,int)
              int  result=sum(1,2);
                          ^
1 个错误

C:\Users\Zu\Desktop>
```

解决方案

- 静态方法只能访问静态方法和静态成员
- 非静态方法要被实例化才能被静态方法调用

静态方法只能直接访问静态方法和静态成员

```
1  class Test{
2      public static int sum(int a,int b){//加入static关键字，变成静态方法
3          return a+b;
4      }
5      public static void main(String[] args){
6          int  result=sum(1,2);//静态方法调用静态方法
7          System.out.println("result="+result);
8      }
9  }
```

非静态方法要被实例化才能被静态方法调用

```
1  class Test{
2      public int sum(int a,int b){
3        return a+b;
4      }
5      public static void main(String[] args){
6        Test test=new Test();//实例化类
7        int  result=test.sum(1,2);//调用非静态方法
8        System.out.println("result="+result);
9      }
10 }
```

问题

一．你知道main()为什么要用static修饰的原因了吗？

二．main()方法中的static可以去掉吗？

静态代码块与非静态代码块的异同点

- 相同点：
  1. 都是在JVM加载类时且在构造方法执行之前执行，在类中都可以定义多个。
  2. 一般在代码块中对一些static变量进行赋值。

静态代码块与非静态代码块的异同点

- 不同点：

1. 静态代码块在非静态代码块之前执行：

   ① 静态代码块—>非静态代码块—>构造方法

2. 静态代码块只在第一次new执行一次，之后不再执行，而非静态代码块在每new一次就执行一次

PuTong.java

```java
public class PuTong {
    public PuTong() {
        System.out.print("默认构造方法! -->");
    }
    // 非静态代码块
    {
        System.out.print("非静态代码块! -->");
    }
    // 静态代码块
    static {
        System.out.print("静态代码块! -->");
    }
   //静态成员方法
    public static void test() {
            System.out.println("普通方法中的代码块! ");
    }
    public static void main(String[] args) {
        PuTong c1 = new PuTong();
        c1.test();  //or PuTong.test();
        PuTong c2 = new PuTong();
        c2.test(); //or PuTong.test();
    }
}
```

```
静态代码块! -->非静态代码块! -->默认构造方法! -->普通方法中的代码块!
非静态代码块! -->默认构造方法! -->普通方法中的代码块!
```

staticBlock.java

```
This is static block.
This is main method.
counter=5
```

```java
public class staticBlock {

    private static int counter;
    public static void main(String args[]) {
        System.out.println("This is main method.");
        System.out.println("counter="+counter);
    }

    // 定义一个静态代码块,静态代码块的执行早于main()的执行
    static {
        System.out.println("This is static block.");
        counter = 5; //初始化counter
    }
}
```

静态代码块

一. 静态代码块只能定义在类里面，它独立于任何方法，不能定义在方法里面。

二. 静态代码块里面声明的变量都是局部变量，只在本块内有效。

三. 静态代码块会在类被加载时自动执行，而无论加载者是JVM还是其他的类。

四. 一个类中允许定义多个静态代码块，执行的顺序根据定义的顺序进行。

五. 静态代码块只能访问类的静态成员，而不允许访问实例成员。

类的静态属性和静态方法(小结)

- 类变量和类方法
  1. 类的变量和方法是由关键字*static*修饰的的。
  2. 类变量和类方法由该类产生的所有实例共享。
  3. 调用方式：
     - 类名\对象名.静态变量
     - 类名\对象名.静态方法

- 实例变量和方法
  1. 每一个实例拥有一份独立拷贝，因此每个实例对象的数据是独立且唯一的。
  2. 调用方式：
     1. 对象名.事例变量
     2. 对象名.事例方法

类的静态属性和静态方法（小结）

- 即使没有创建该类的具体对象，类中的static类成员也会存在，这时可以通过：

1. 类名.静态变量

2. 类名.静态方法

注意

- static方法中不能直接调用非static的域或方法（必须通过对象名引用）。
  1. static函数并不需要先产生任何对象，就可以通过类名来调用。
  2. non-static数据/函数和对象绑定（原因）。
  3. 在static函数中“直接”取用non-static数据/函数，会产生语法错误。

Java中static class使用方式（以后学习）

- 在java中，不能用static修饰顶级类（top level class）。只有内部类可以为static。
- 静态内部类和非静态内部类之间区别：
  - 1．内部静态类不需要有指向外部类的引用。但非静态内部类需要。
  - 2．静态类只能访问外部类的静态成员，非静态内部类能够访问外部类的静态和非静态成员。
  - 3．非静态内部类不能脱离外部类实体被创建，非静态内部类可以访问外部类的数据和方法，因为他就在外部类里面。

本节主要内容

- 类的定义
  - 成员变量的定义
  - 成员方法的定义
- 对象的生成、使用和清除
  - 默认构造函数（不带参数构造函数）
  - 带参数构造函数
  - 垃圾内存自动回收机制
- 引用类型变量内存分配、this关键字
- 类变量和类方法（static），实例变量和实例方法