# **JavaSE阶段**

## **Day09**

### **类和对象-面向对象**

#### **本课目标**

掌握面向对象思想

掌握类和对象的创建

掌握对象创建内存分析

掌握构造方法

掌握this关键字

#### **面向对象：万事万物皆对象**

对象：用来描述客观事物的一个实体，由一组属性和方法构成

对象的两个要素：属性（有什么），方法（能干什么）

属性——对象具有的各种特征 每个对象的每个属性都拥有特定值

方法——对象执行的操作

对象同时具有属性和方法两项特性 对象的属性和方法通常被封装在一起，共同体现事物的特性， 二者相辅相承，不能分割

#### **什么是类——从对象抽象出类**

类是模子，定义对象将会拥有的特征（属性）和行为（方法）

类是对象的类型 不同于int类型：具有方法

##### **对象和类的关系：**

类是抽象的概念，仅仅是模板 比如说：“人” 对象是一个你能够看得到、摸得着的具体实体

练习：如果要对10个学生的成绩进行排序，然后输出排序结果，应该怎么做？

方案一：

1:创建一个String型的数组，存放姓名

2:创建一个int型数组，存放分数 3:对成绩排序，同步调整成绩和姓名数组

4:同时遍历两个数组，打印姓名和成绩

public class test9 {  
 public static void main(String[] args) {  
 // TODO Auto-generated method stub  
 String[] names = { "一", "三", "二", "四", "五" };  
 int[] scores = { 99, 76, 81, 56, 51 };  
 sort(names, scores);  
 print(names, scores);  
 }  
​  
 public static void sort(String[] names, int[] scores) {  
 int swapscore = 0;  
 String swapName = null;  
 for (int j = 0; j < scores.length - 1; j++) {  
 for (int i = 0; i < scores.length - 1 - j; i++) {  
 if (scores[i] > scores[i + 1]) {  
​  
 // 交换分数  
 swapscore = scores[i];  
 scores[i] = scores[i + 1];  
 scores[i + 1] = swapscore;  
 // 交换姓名  
 swapName = names[i];  
 names[i] = names[i + 1];  
 names[i + 1] = swapName;  
 }  
 }  
 }  
 }  
​  
 public static void print(String[] names, int[] scores) {  
 for (int i = 0; i < scores.length; i++) {  
 System.out.println("name=" + names[i] + " ; score=" + scores[i]);  
 }  
 }  
​  
}

如果还有性别，年龄要保存呢？

定义一个新的类型，由这个新类型创建的变量可以同时存储一个同学的姓名成绩

1.创建一个新类型Student，新类型我们称为Student类

class Student{  
   String name;  
   int score;  
}

2.根据新类型申请一个空间，这个空间可以完整存储一个同学的信息，我们称为一个Student对象

new Student();

申请的空间中有name和score

3.通过一个引用型变量记录这个空间的起始地址

Student student=new Student();

4.通过地址找到对应的空间

student.name="小白";  
student.score=100;

练习：定义一个school类/创建和使用对象

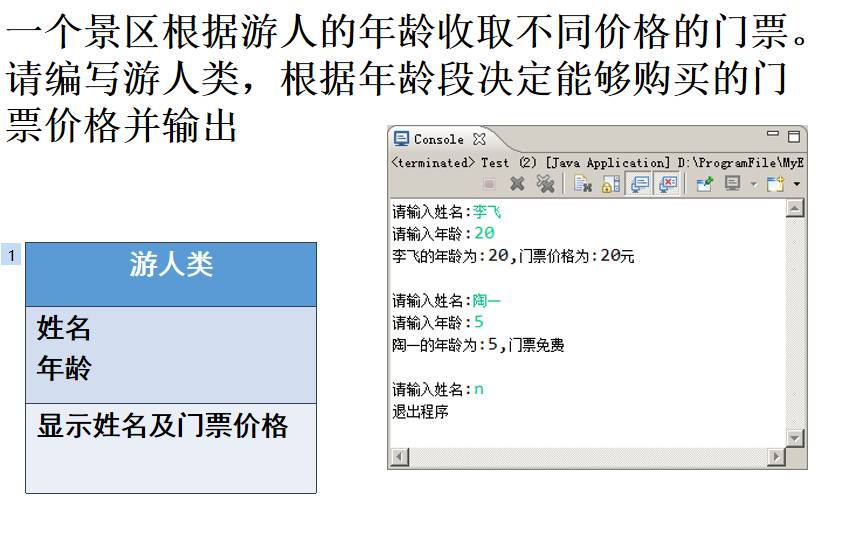
public class Student {  
 String schoolName; //学校名称  
 int classNumber; //教室数目  
 int labNumber; //机房数目  
​  
 public void showCenter(){  
 System.out.println(schoolName + "培训学员\n" + "配备："   
 + classNumber + "教室" + labNumber + "机房");  
 }

public class test9 {  
 public static void main(String[] args) {  
 // TODO Auto-generated method stub  
 Student center = new Student();  
 System.out.println("\*\*\*初始化成员变量前\*\*\*");  
 center.showCenter();  
 center.schoolName = "北京大学"; //给schoolName属性赋值  
 center.classNumber = 10; //给classNumber属性赋值  
 center.labNumber = 10; //给labNumber属性赋值  
 System.out.println("\n\*\*\*初始化成员变量后\*\*\*");  
 center.showCenter();  
 }  
}

方案二：

public class Student {  
 String name;  
   int score;  
​  
 public String toString() {  
 return "Student [name=" + name + ", score=" + score + "]";  
 }  
   
}  
​

public class test10 {  
 public static void main(String[] args) {  
 // TODO Auto-generated method stub  
 Student[] students = new Student[5];  
 students[0] = new Student() ;  
 students[0].name = "一";  
 students[0].score = 99;  
   
 students[1] = new Student();  
 students[1].name = "三";  
 students[1].score = 76;  
   
 students[2] = new Student();  
 students[2].name = "二";  
 students[2].score = 81;  
   
 students[3] = new Student();  
 students[3].name ="四";  
 students[3].score = 56;  
   
 students[4] = new Student();  
 students[4].name ="五";  
 students[4].score = 96;  
   
 sort(students);  
 print(students);  
 }  
 public static void sort(Student[] students) {  
 Student swap = null;  
 for (int j = 0; j < students.length - 1; j++) {  
 for (int i = 0; i < students.length - 1 - j; i++) {  
 if (students[i].score > students[i + 1].score) {  
​  
 // 交换分数  
 swap = students[i];  
 students[i] = students[i + 1];  
 students[i + 1] = swap;  
   
 }  
 }  
 }  
 }  
 public static void print(Student[] students) {  
 for (int i = 0; i < students.length; i++) {  
 System.out.println(students[i].toString());  
 }  
 }

练习：

//游客类  
public class visitor {  
 String name;  
 int age;  
 public void show() {  
 Scanner intput = new Scanner(System.in);  
 while(!"n".equals(name)){  
 if(age>=18 && age<=60){ //判断年龄  
 System.out.println(name + "的年龄为：" + age +  
 "，门票价格为：20元\n" );  
 }else{  
 System.out.println(name + "的年龄为：" + age + "，门票免费\n");  
 }  
 System.out.print("请输入姓名：");  
 name = intput.next(); //给name属性赋值  
 if(!"n".equals(name)){  
 System.out.print("请输入年龄：");  
 age = intput.nextInt(); //给age属性赋值  
 }  
 }  
 System.out.print("退出程序");  
 }  
​  
}  
​

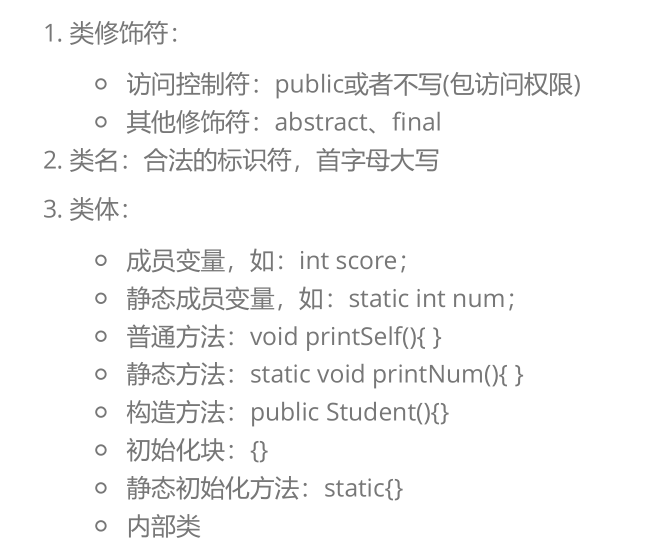
public class InitialVistor {  
 public static void main(String[] args) {  
 Scanner input = new Scanner(System.in);  
 Visitor v = new Visitor(); //创建对象  
 System.out.print("请输入姓名：");  
 v.name = input.next(); //给name属性赋值  
 System.out.print("请输入年龄：");  
 v.age = input.nextInt(); //给age属性赋值  
 v.show();                       //调用显示信息方法  
 }  
}

#### **类的定义**

类就是类型，是数据类型的复合类型，是一个同时带有特征和行为的数据类型，是创建对象的模板。类是Java语言的最基本单位

类修饰符 class 类名{  
//类体  
}

public class Student {  
 String name;  
   int score;  
​  
 public String toString() {  
 return "Student [name=" + name + ", score=" + score + "]";  
 }  
   
}



注意: 1.一个Java文件中可以写多个类，但是编译后还是会生成多个.class文件 2.一个Java文件中最多只能有一个被public修饰的类，这个类的类名必须和Java源文件名保持一致 3.一个Java文件中如果只有一个类，并且这个类不用public修饰的话，类名和文件名可以不一样,但习惯上，我们都会写上public

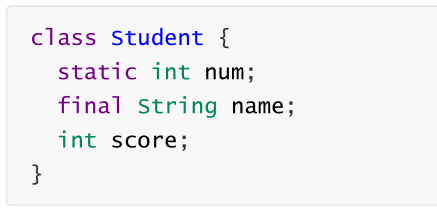
#### **成员变量的定义**

成员变量:作为类的成员存在的变量，也被称为属性。分为静态(static)成员变量和非静态成员变量，在静态方法中只能访问静态成员变量，在非静态方法中可以访问非静态成员变量和静态成员变量

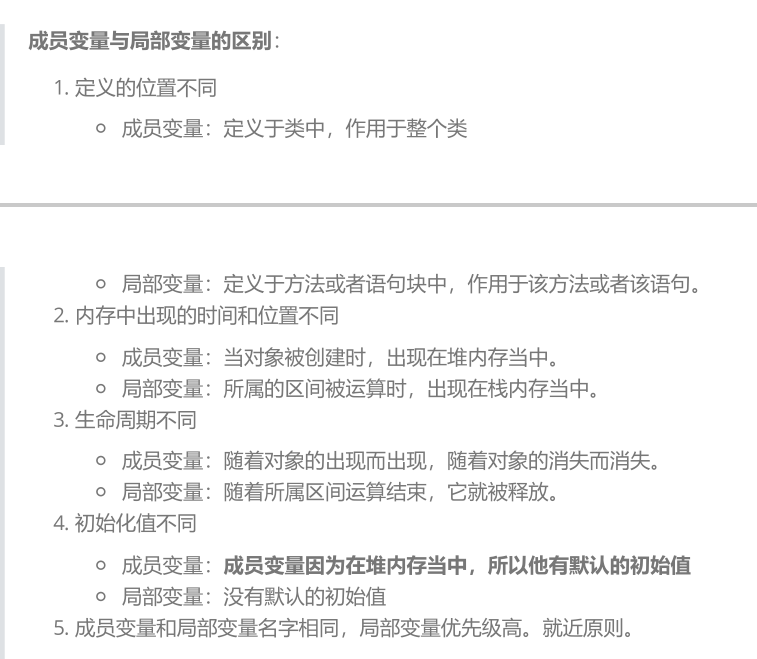
类修饰符 class 类名 {  
成员变量修饰符 变量类型 变量名称;  
}

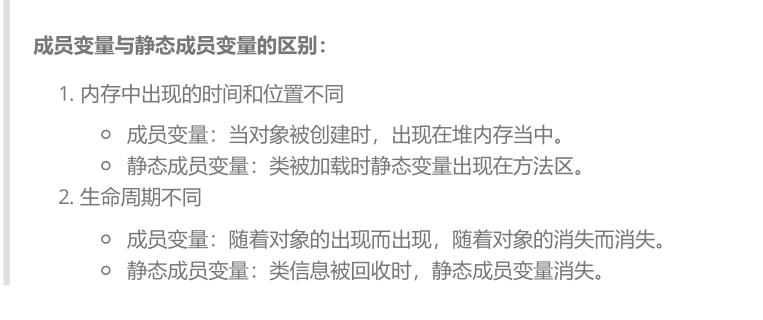
成员变量修饰符: ●访问控制符: public、protected、包(默认)、private

●其他修饰符: static、final



final:如果成员变量的修饰符中有final，表示该成员变量只可以被赋值一次，可以在变量声明处或者构造函数中初始化。





#### **类中方法的定义**

方法必须作为类的成员出现。分为静态方法和非静态方法，在静态方法中只能调用静态方法，在非静态方法中可以调用非静态方法和静态方法 1.静态能访问静态的，不能直接访问非静态的，如果要访问，必须要创建对象。

2.非静态的能直接访问非静态的和静态的。 3.如果要访问另外一个类中的静态成员变量和静态方法:(类名.静态成员变量 类名.静态方法名）

public static void main(string[] args) {  
student.num=1;  
system.out.print1n(student. getNum());  
​  
student s1=new student();  
s1.name="小王";  
system.out.println(s1.tostring();  
}  
​  
c1ass student {  
static int num;  
string name ;  
int score;  
​  
public static int getNum(){  
return num;  
}  
​  
public string tostring( {  
return "name=" + name + ";score=" + score;}  
}  
​

#### **对象的创建以及内存分析**

类是对象的抽象（模板），对象是类的具体体现（实例）

##### **对象的创建**

对象的创建过程也被称为对象的实例化过程

new 类名()；

示例

new Student();

##### **类引用型变量**

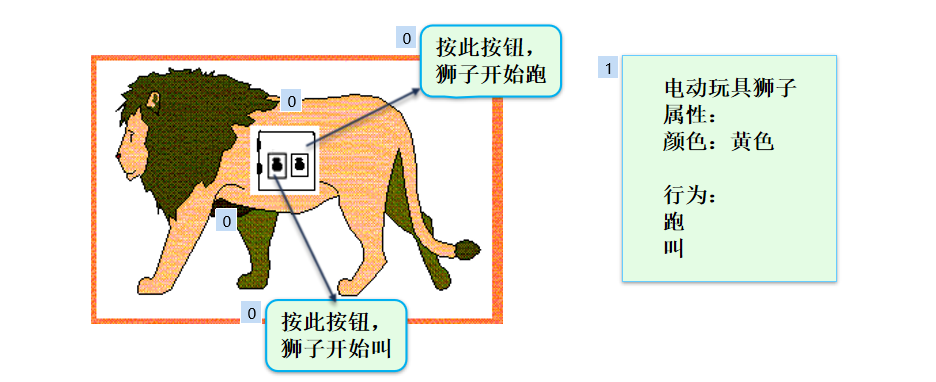
通过一个变量记录对象的地址，从而多次使用这个对象。

类名 变量名;//声明  
类名 变量名 = new 类名();//声明、初始化

通过引用型变量使用对象: ●调用属性: 对象名.属性名 ●调用方法: 对象名.方法名(实参); 注意:引用型变量作为方法参数和返回值，规则同数组作为方法参数和返回值

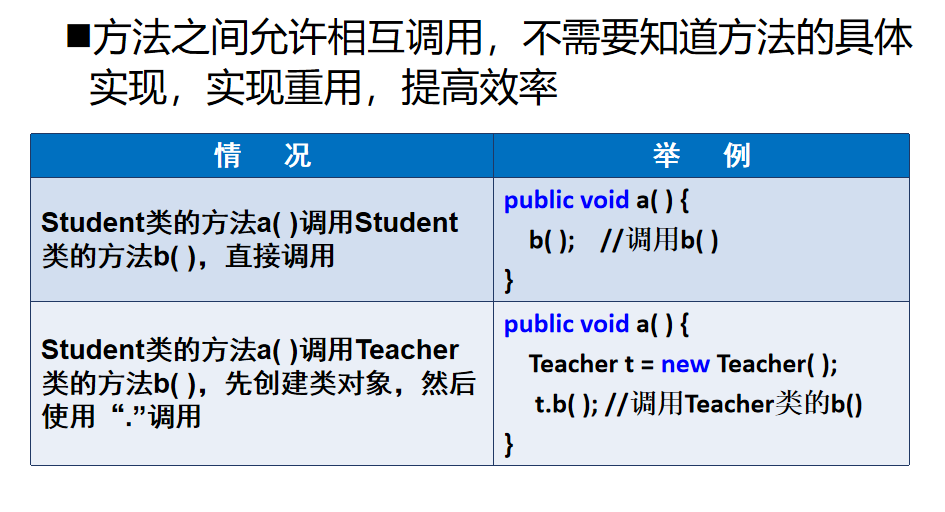
student student1= new student();  
student1.name = "一";  
student1.score = 99;  
string str=student1.tostring();  
​

#### **类的无参方法**



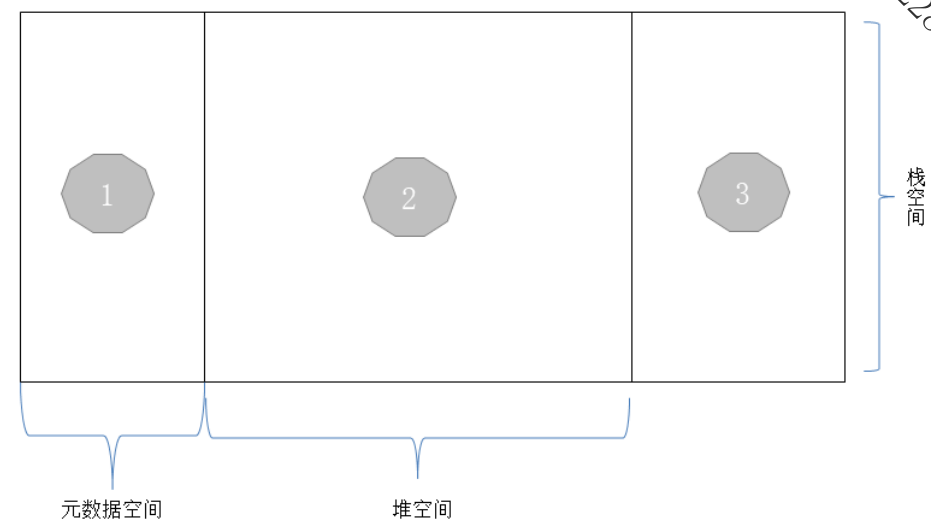
public class AutoLion {  
 String color = "黄色"; // 颜色  
​  
 public void run() {  
 System.out.println("正在以0.1米/秒的速度向前奔跑。");  
 }  
​  
 public String bark() {  
 String sound = "大声吼叫";  
 return sound;  
 }  
 public String getColor() {  
 return color;  
 }  
​  
 public String showLion() {   
 return "这是一个" + getColor() + "的玩具狮子!";   
 // 另一种方式   
 //return "这是一个" + color + "的玩具狮子!";  
 }  
}  
​

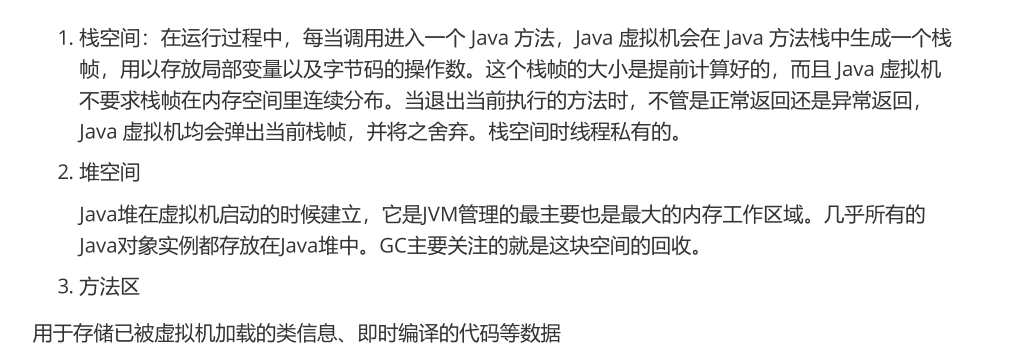
public class TestLion {  
 public static void main(String[] args) {  
 AutoLion lion = new AutoLion();         //创建AutoLion对象  
         System.out.println(lion.showLion());   //调用方法显示类信息  
         lion.run();                             //调用跑方法  
         System.out.println(lion.bark());       //调用叫方法  
 }  
}



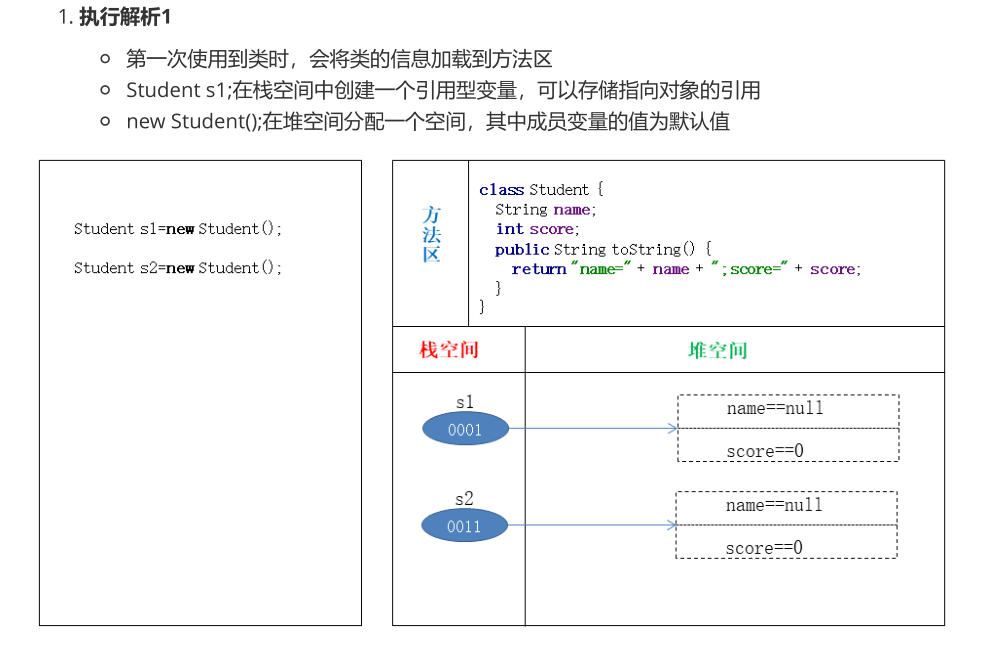
#### **jvm内存布局**

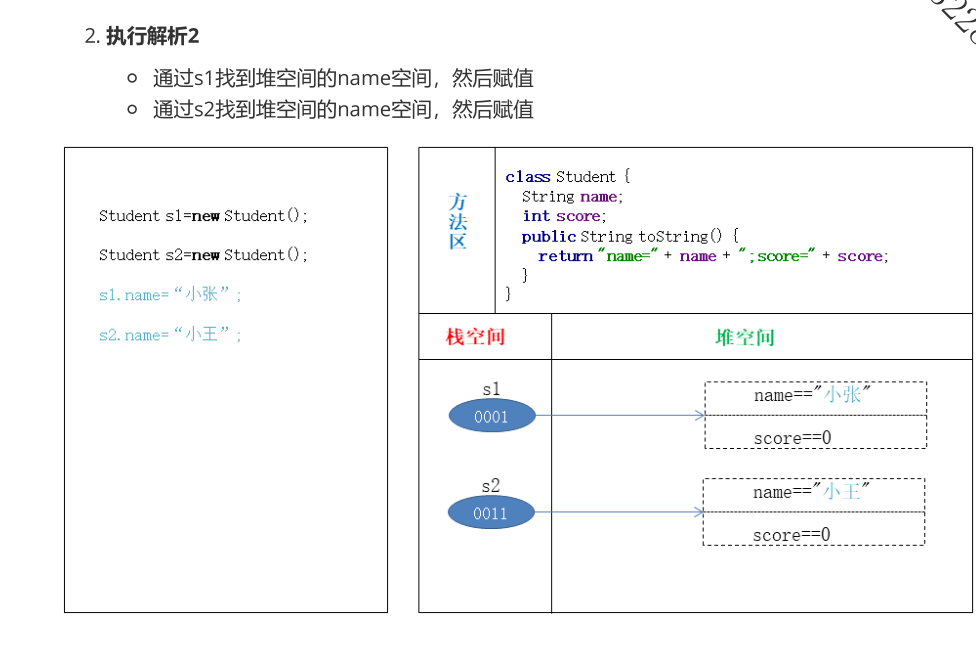
程序运行时，jvm会按区域来使用内存，主要的内存空间：

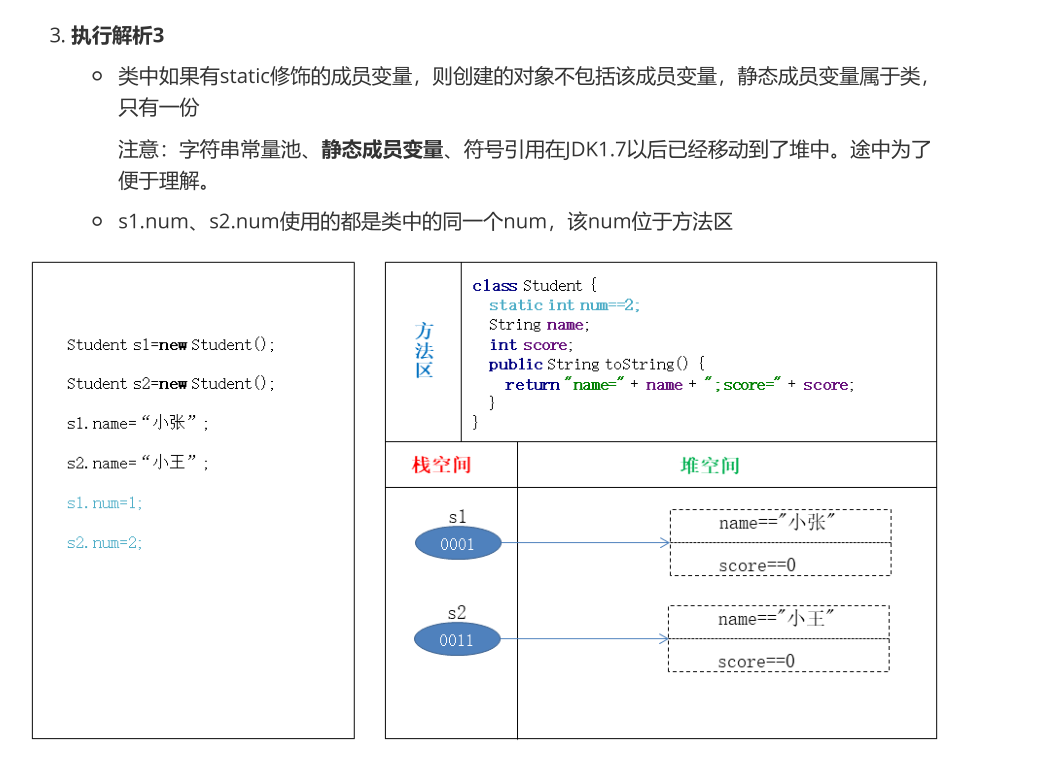




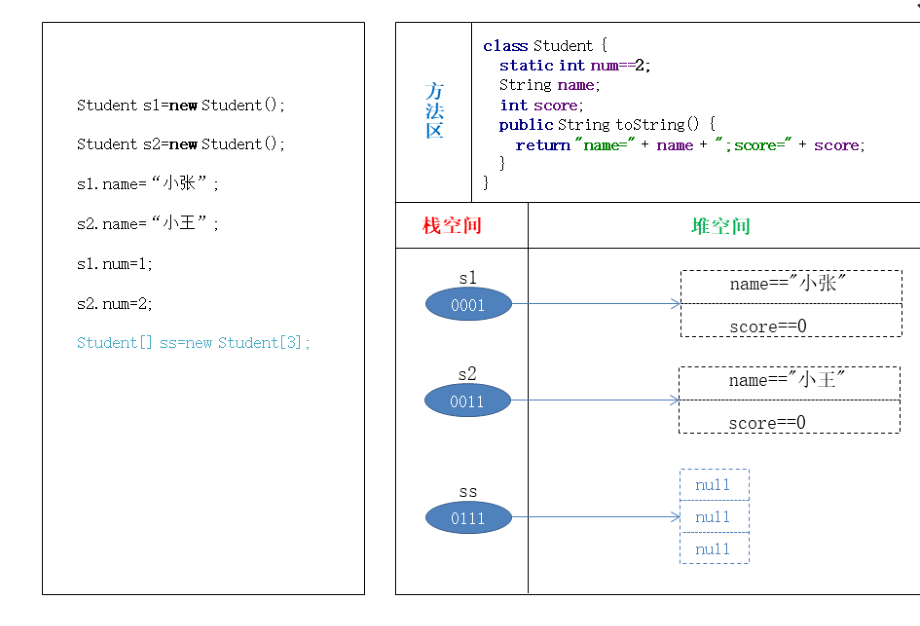
#### **jvm内存分配**

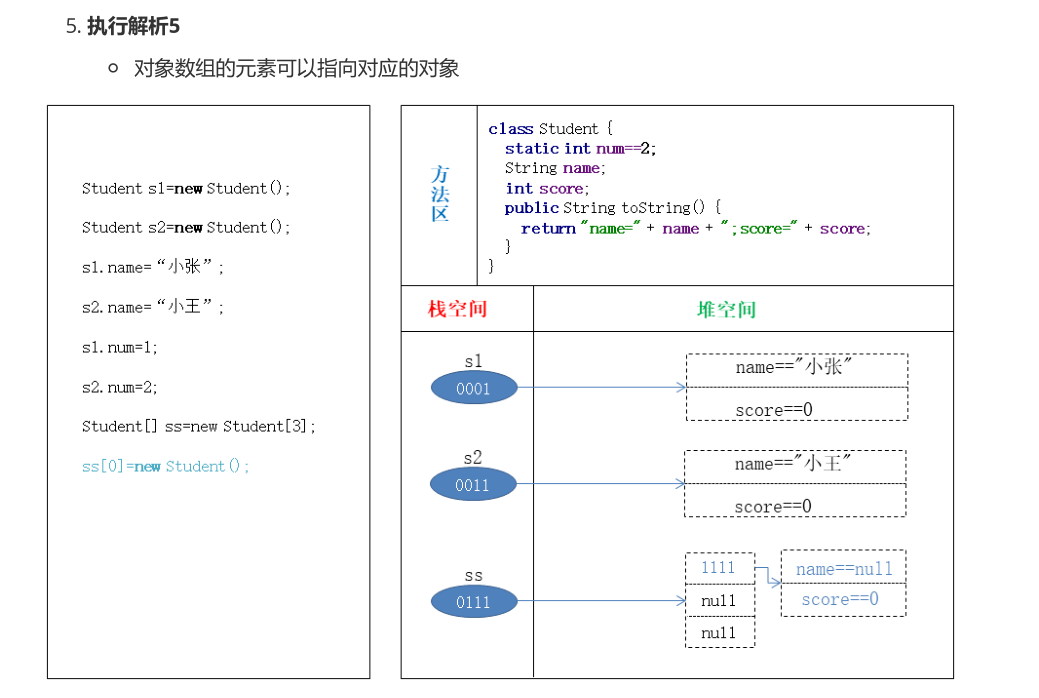




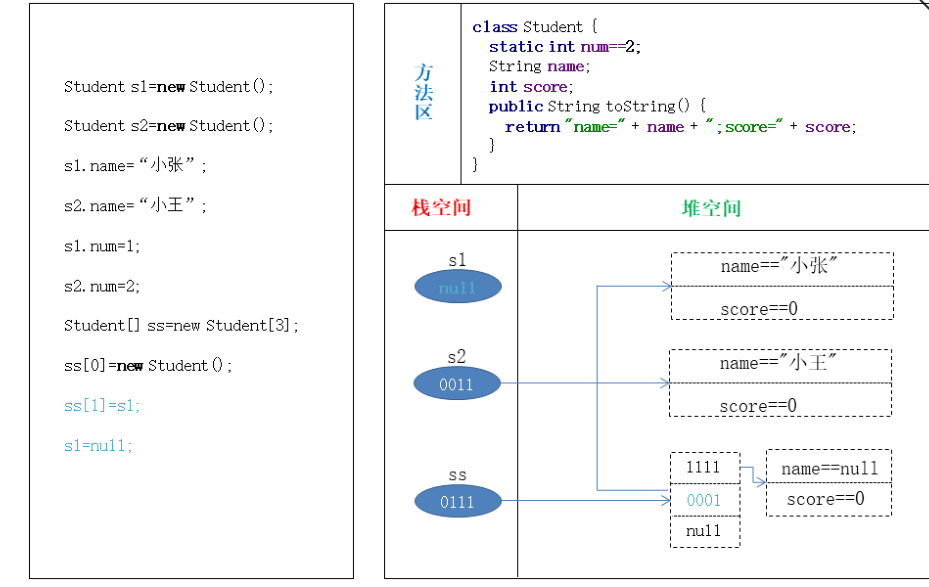


4.执行解析





6.执行解析6 多个引用型变量可以指向一个对象。当不需要某个对象时可以将所有执行该对象的引用型变 量赋值为null。当一个对象没有引用型变量指向的时候，GC会在适当的时候回收这个对象。



总结: 1.静态成员变量的使用静态成员变量和方法都隶属于类，可以通过引用型变量访问，但是最好使用类名.属性的语法来使用。 2.静态方法的使用 只要类被加载到方法区，静态方法就可以被调用(类名.方法名())，但是在静态方法中不能直接使用非静态的东西。