Java语言 面向对象

面向过程：强调步骤；

面向对象：强调对象；

类是对象的抽象，而对象是类的具体实例。

类是抽象的，不占用内存；而对象是具体的，占内存空间。

最小程序没有任何命令，这就是为什么它称为“最小”。

方法的主体由命令组成。

public class Home

{

public stastc void void main(String[] arg)

{

}

}

变量是用来存储数据的特殊实体。

变量的三个重要属性：类型、名称、值。

一个变量的类型决定了可以存储在其中的值/数据的类型。

值是存储在变量中的特定的对象、数据或者信息。

变量也有类型，它只能存储类型与其自身相同的值。

不能再同一个方法中创建两个名称 相同的变量。

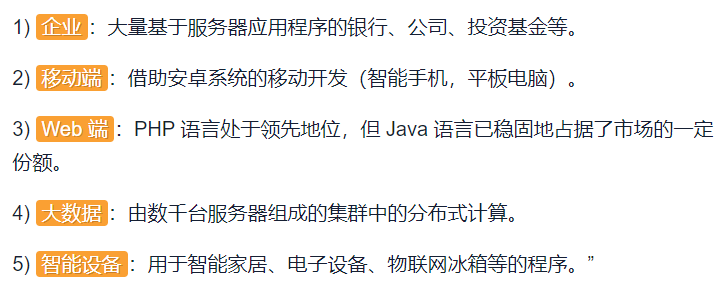
编译器是一个特殊的程序，它可将用编程语言编写的程序转化成一系列的机器代码。

Java 的编译过程：用Java语言编写的程序—Java编译器—与平台无关的特殊代码—Jvm—机器代码

这种方法可以让Java语言编写的程序在任何的设备上运行：计算机、手机、智能手机、自动取款机、烤面包机、甚至信用卡。

所有的安卓系统程序都是用Java语言编写的。

Java 语言在以下变成领域占据了主导地位：



一个程序由一系列带有Java文件扩展名的文件组成，每个文件只包含一个类的代码。

如果一个文件名为MyCat.java ，则它包含MyCat类。

复制是指将另一个变量的值或者由多个变量计算的值放入一个变量中。

变量分为两种：原始类型和引用类型。原始类型直接存储值，而引用类型存储对象的引用。

原始类型包括 int、char、boolean 和许多其他类型。引用类型包括其他所有类型。我们使用类来创建它们。

使用面向对象代码前会出现一行代码

**import** java.util.Arrays; // 使用面向对象时出现的语句；

**int**[] array = {10, 20, 30, 40, 50};  
*//要求打印格式为：[10，20，30，40，50]  
//使用面向过程，每一个步骤细节都要亲力亲为;*System.***out***.print(**"["**);  
**for** (**int** i = 0; i < array.**length**; i++) {  
 **if** (i == array.**length** - 1) { *// 如果是最后一个元素* System.***out***.println(array[i] + **"]"**);  
 } **else** { *// 如果不是最后一个元素* System.***out***.print(array[i] + **","**);  
 }  
}  
System.***out***.println(**"===================="**);  
*//使用面向对象  
//找一个JDK给我们提供好的Arrays类  
//其中一个toString方法，直接就能把数组变成想要的格式的字符串；*System.***out***.println(Arrays.*toString*(array));

注意事项：

1. 成员变量是直接定义在类当中的，在方法外边。‘
2. 成员方法不要写static关键字。

**package** faceObject;  
  
*// 对象的创建和使用*  
*/\*通常情况下，一个类并不能直接使用，需要根据类创建一个对象，才能使用。  
创造对象的三个步骤：  
  
1.导包：也就是指出需要使用的类，在什么位置。  
import 包名称.类名称;  
import faceObject.Student;  
对于和当前类属于同一个包的情况，可以省略导包语句不写。  
  
2.创建，格式：  
类名称 对象名 = new 类名称();  
Student stu = new Student();  
  
3.使用，分为两种情况：  
使用成员变量：对象名.成员变量名  
使用成员方法：对象名.成员方法名(参数)  
 (也就是，想用谁，就用对象名点谁)  
  
注意事项：  
如果成员变量没有进行赋值，那么将会有一个默认值，规则和数组一样。  
 \*/*

**import** org.w3c.dom.ls.LSOutput; // 使用面向对象时自己出现的语句；

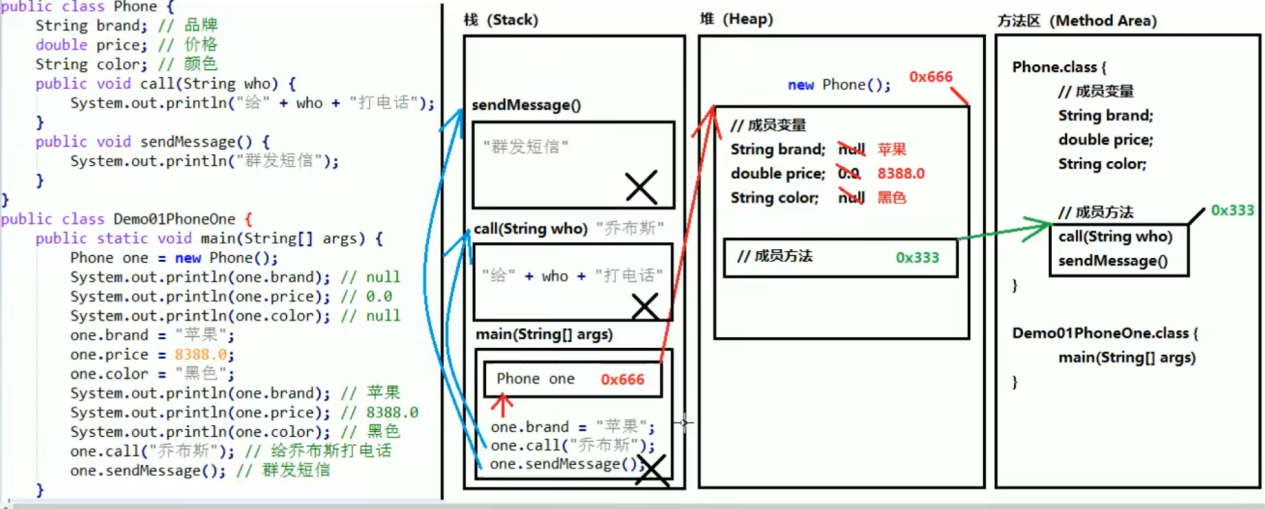
**public class** faceStudent {  
 **public static void** main(String[] args) {  
  
 *// 1.导包  
 //我需要使用Student类，和我自己faceStudent位于同一个包下，所以省略导包语句。  
  
 // 2.创建，格式：  
 // 根据Student类，创建了一个名为stu的对象* Student stu = **new** Student();  
  
 *// 3.使用其中的成员变量，格式：  
 //对象名.成员变量名* System.***out***.println(stu.**name**); *// null;* System.***out***.println(stu.**age**); *// 0;* System.***out***.println(**"================="**);  
  
 *//改变对象当中的成员变量数值内容  
 //将右侧的字符串，赋值交给stu对象当中的name成员变量* stu.**name** = **"阿文"**;  
 stu.**age** = 10;  
 System.***out***.println(stu.**name**); *//阿文* System.***out***.println(stu.**age**); *//10;* System.***out***.println(**"================="**);  
  
 *// 使用对象的成员方法。格式：  
 // 对象名.成员方法名();* stu.eat();  
 stu.sleep();  
 stu.study();  
 }  
}

全局变量全部存在在静态存储区（堆）；

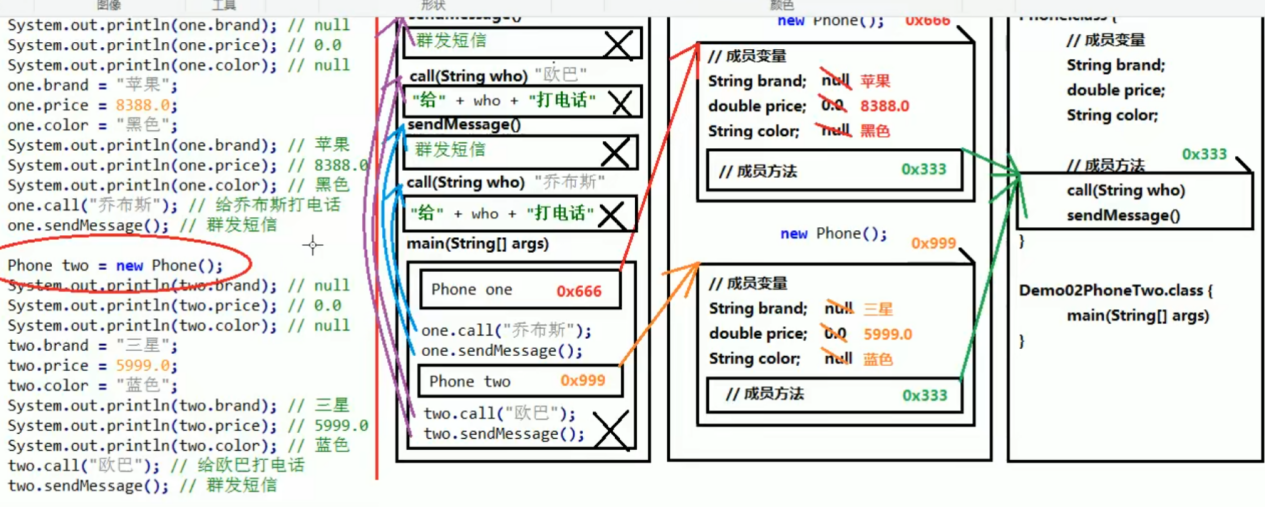
函数方法在动态存储区（栈）；

面对对象程序的存储形式即调用方法

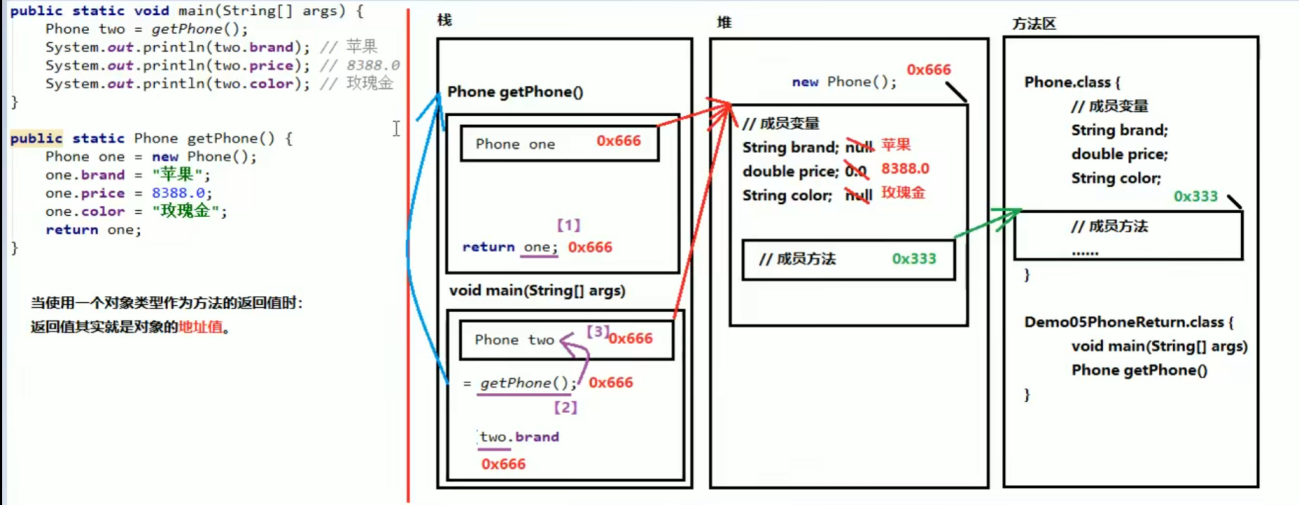
使用一个类面向一个对象的内存图：



使用一个类面向两个对象的内存图：



使用对象类型作为方法的返回值



局部变量和成员变量的区别

1. 定义的位置不一样【重点】

局部变量：在方法内部。

成员变量：在方法的外部，直接写在类当中。

1. 作用范围不一样【重点】

局部变量：只有方法当中才可以使用，除了方法就不能再用。

成员变量：整个类全都可以通用。

1. 默认值不一样【重点】

局部变量：没有默认值，如果在想使用，必须手动进行赋值。

成员变量：如果没有进行赋值，会有默认值，规则和数组一样。

1. 内存的位置不一样（了解）

局部变量：位于栈内存。

成员变量：位于堆内存。

1. 生命周期不一样（了解）

局部变量：随着方法进栈而诞生，随着方法出栈而消失。

成员变量：随着对象的创建而诞生，随着对象被垃圾回收而消失。

**public class** Dom01VariableDifference {  
 String **name**; *// 成员变量* **public void** methodA(){  
 **int** num = 20; *// 局部变量* System.***out***.println(num);  
 System.***out***.println(**name**);  
 }  
 **public void** methodB(**int** param){ *//方法的参数就是局部变量  
 // 参数在方法调用的时候，必然会被赋值。* System.***out***.println(param);  
 **int** age; *// 局部变量  
 //System.out.println(age); age没有进行赋值;  
 //System.out.println(num); 错误写法;* System.***out***.println(**name**);  
 }  
}

**package** demo01;  
*/\*面对对象的三大特征：封装、继承、多态。  
  
封装性在Java当中的体现：  
1.方法就是一种封装。  
2.关键字private也是一种封装。  
  
封装就是将一些细节隐藏起来，对于外界不可见。  
 \*/  
eg：找数组中的最大值，打擂台***public class** Demo01Method {  
 **public static void** main(String[] args) {  
 **int**[] array = {5,15,25,20,100};  
 **int** max = *getMax*(array);  
 System.***out***.println(**"最大值："**+ max);  
 }  
 *// 给我一个数组，我还给你一个最大值*

*//下边的这个方法，在主函数中看不见。可以进行很好的封装。* **public static int** getMax(**int**[] array){  
 **int** max = array[0];  
 **for** (**int** i = 1; i < array.**length**; i++) {  
 **if**(array[i]>max){  
 max = array[i];  
 }  
 }  
 **return** max;  
 }  
}

private关键字的作用及使用

/\*

问题描述：定义Person的年龄时，无法阻止不合理的数值被设置进来。

解决方案：用private关键字将需要保护的成员变量进行修饰。

一旦使用了private进行修饰，那么本类当中仍然可以随意访问。

但是！超出了本类范围之外就不能在直接访问了，可以间接访问。

间接访问private成员变量，就是定义一对 Getter/Setter 方法。

必须叫setXxx 或者是 getXxx 命名规则。

**private int age**; *// 年龄*

对于 Setter 来说，不能有返回值，参数类型和成员变量对应。

**public void** setAge(**int** num){  
 **if**(num<100 && num >= 0){ *// 如果是合理情况* **age** = num;  
 }**else**{  
 System.***out***.println(**"数据不合理！"**);

}  
 }

对于Getter 来说，不能有参数，返回值类型和成员变量对应。

**public int** getAge(){  
 **return age**;  
}

**package** demo01;  
*/\*  
对于基本类型当中的booolean值， Getter方法一定要写成isXxx的形式，而setXxx规则不变。  
 \*/*

*类函数：***public class** Student {  
 **private** String **name**; *// 姓名* **private int age**; *// 年龄* **private boolean male**; *// 是不是爷们儿* **public void** setMale(**boolean** b){  
 **male** = b;  
 }  
 **public boolean** isMale(){  
 **return male**;  
 }  
 **public void** setName(String str){  
 **name** = str;  
 }  
 **public** String getName(){  
 **return name**;  
 }  
 **public void** setAge(**int** num){  
 **age** = num;  
 }  
 **public int** getAge(){  
 **return age**;  
 }  
}

主函数：

**package** demo01;  
  
**public class** Studenbt01 {  
 **public static void** main(String[] args) {  
 Student stu = **new** Student();  
  
 stu.setName(**"张"**);  
 stu.setAge((20));  
 stu.setMale(**true**);  
  
 System.***out***.println(**"姓名："** + stu.getName());  
 System.***out***.println(**"年龄："** + stu.getAge());  
 System.***out***.println(**"是不是爷们儿"** + stu.isMale());  
 }  
}

当方法的局部变量和类的成员变量重名的时候，根据“就近原则”，优先使用局部变量。如果需要访问本类中的成员变量，需要使用格式：

this.成员变量名

“通过谁调用的方法，谁就是this。”

*/\*  
对于基本类型当中的booolean值， Getter方法一定要写成isXxx的形式，而setXxx规则不变。  
 \*/*

*类函数：***public class** Student {  
 String **name** ; *// 我自己的名字  
 //参数name是对方的名字；  
 //成员变量name是自己的名字；* **public void** sayHello(String name){  
 System.***out***.println(name + **",你好。我是"** + **this**.**name**);  
 }  
}

主函数：  
**public class** Studenbt01 {  
 **public static void** main(String[] args) {  
 Student stu = **new** Student();  
 *// 设置我自己的名字* stu.**name** = **"张"**;  
 stu.sayHello(**"王"**);  
 }  
}

构造方法是专门用来创建对象的方法，当我们通过关键字new来创建对象时，其实就是在调用构造方法。

格式：

public 类名称（参数类型 参数名称）{ //这里边可以没有参数也可以有多个参数

方法体

}

注意事项：

1. 构造方法的名称必须和所在的类名称完全一样，就连大小写也要一样。
2. 构造方法不要写返回值类型，连void都不要写。
3. 构造方法不能return一个具体返回值。
4. 如果没有编写任何构造方法，那么编译器将会默认赠送一个构造方法，没有参数、方法体什么事情都不做。 public Student(){}
5. 一旦编写了至少一个构造方法，那么编译器将不再赠送。
6. 构造方法也是可以重载的。

重载：方法名称相同，参数列表不同。

类代码：

**package** demo01;  
  
**public class** Student{  
 *//成员变量* **private** String **name**;  
 **private int age**;  
  
 *//无参数的构造方法* **public** Student() {  
 System.***out***.println(**"无参数构造"**);  
 }  
 *//全参数构造方法* **public** Student(String name, **int** age) {  
 System.***out***.println(**"全参数构造"**);  
 **this**.**name** = name;  
 **this**.**age** = age;  
 }  
  
 **public** String getName() {  
 **return name**;  
 }  
  
 **public void** setName(String name) {  
 **this**.**name** = name;  
 }  
  
 **public int** getAge() {  
 **return age**;  
 }  
  
 **public void** setAge(**int** age) {  
 **this**.**age** = age;  
 }  
}

主函数：

**package** demo01;  
  
**public class** Studenbt01 {  
 **public static void** main(String[] args) {  
 Student stu = **new** Student(); *//无参数构造* System.***out***.println(**"================="**);  
 Student stu2 = **new** Student(name:**"张"**,age:20); *// 全参构造* System.***out***.println(**"姓名："** + stu2.getName() + **"，年龄："** + stu2.getAge());  
 *//如果需要改变对象的成员变量内容，仍然还需要使用ssetXxx方法* stu2.setAge(21);  
 System.***out***.println(**"姓名："** + stu2.getName() + **"，年龄："** + stu2.getAge());  
  
 }  
}

定义一个标准类

一个标准的类通常要有下面四个组成部分：

1. 所有成员变量都要用private关键字修饰
2. 为每一个成员变量编写一对儿Getter / Setter 方法
3. 编写一个无参数的构造方法
4. 编写一个全参数的构造方法

这样标准的类也叫做Java Bean

【重点】：当想要创建Getter / Setter方法时，存在快捷键 Alt+Ins 按住shift可以创建全参数方法，也可以不选择进行创建无参数方法。也可以选择界面中的code，里边的Generate 进行选择。

API和面向对象

* + - 1. API的概述和使用方法

概述：API，应用程序编程接口。Java API就是一本程序员字典，是JDK提供给我们使用的类的说明文档。

* + - 1. Scanner概述及其API文档

Scanner 类的功能：可以实现键盘输入数据到程序当中。

// 除了基本类型以外，其余都为引用函数。

引用类型的一般步骤：  
*// 1.导包***import** java.util.Scanner; *// 自动生成的导入语句，可以自己写，包路径可以在API中查找。  
  
/\*  
1.导包  
import 包路径.类名称； // 包路径可以在API文档中查到  
如果需要使用的目标类，和当前类位于同一个包下，则可以省略导包语句不写。  
只有java.lang包下的内容不需要导包，其他的包都需要import语句。  
当使用API中的引用函数时，会在代码最上边自动生成一个导入语句。  
  
2。创建  
类名称 对象名 = new 类名称();  
  
3.使用  
对象名.成员方法名()  
  
获取键盘输入的一个int数字，int num = sc.nextInt(); // next后边写上数据类型；  
获取键盘输入的一个字符串：String str = sc.next(); // next后边什么也不加；  
 \*/***public class** Student{  
 **public static void** main(String[] args) {  
  
 *// 2.创建  
 //备注：System.in 代表从键盘进行输入* Scanner sc = **new** Scanner(System.***in***); *// 输入语句，格式大概不变。  
  
 //3. 获取键盘输入的int数字* System.***out***.print(**"输入一个int数字："**);  
 **int** num = sc.nextInt();  
 System.***out***.println(**"输入一个int数字："** + num);  
  
 *//4. 获取键盘的字符串* String str = sc.next();  
 System.***out***.println(**"输入的字符串："** + str);  
 }

* + - 1. 练习 键盘输入两个数求和

**public class** Student{  
 **public static void** main(String[] args) {  
 Scanner str = **new** Scanner(System.***in***);  
 System.***out***.print(**"输入第一个数字："**);  
 **int** num1 = str.nextInt();  
 System.***out***.print(**"输入第二个数字："**);  
 **int** num2 = str.nextInt();  
 **int** result = num1 + num2;  
 System.***out***.println(**"两个数的和为："** + result );  
 }  
  
}

-

* + - 1. 练习 键盘输入三个数，并找出最大值

**public class** Student{  
 **public static void** main(String[] args) {  
 Scanner str = **new** Scanner(System.***in***);  
 System.***out***.print(**"输入第一个数字："**);  
 **int** num1 = str.nextInt();  
 System.***out***.print(**"输入第二个数字："**);  
 **int** num2 = str.nextInt();  
 System.***out***.print(**"输入第三个数字："**);  
 **int** num3 = str.nextInt();  
  
 **int** temp = num1 > num2 ? num1 :num2; // 比较三个数的大小，辗转比较；  
 **int** max = temp > num3 ? temp :num3;  
 System.***out***.println(**"最大值："** + max);  
 }  
  
}

* + - 1. 匿名对象的说明

/\*创建对象的标准形式：

类名称 对象名 = new 类名称()；

匿名对象就是只有右边的对象，没有左边的名字和赋值运算符。

new 类名称();

注意事项：匿名对象只能使用唯一的一次，下次在用不得不在创建一个新的对象。

使用建议：如果确定有一个对象只需要使用唯一的一次，就可以使用匿名对象。

\*/

主函数：

*// 左边的one就是对象的名字*Studenbt01 one = **new** Studenbt01();  
one.**name** = **"张"**;  
one.showName(); *// 我叫张  
  
// 匿名对象***new** Studenbt01().**name** = **"王"**;  
**new** Studenbt01().showName(); *// 我叫：null*

类函数：

String **name** ;  
**public void** showName(){  
 System.***out***.println(**"我叫："** + **name** );  
}

* + - 1. Random的使用（产生随机变量）

**import** java.util.Random; //使用Random方法时API自己产生的导包代码；  
  
*/\*  
Random类用来生成随机数字。使用的三个步骤：  
  
1.导包  
import java.util.Random; // 这个地址也是API所提供的，出现在代码最上边。  
  
2.创建  
Random r = new Random(); //小括号当中留空即可。  
  
3.使用  
获取一个随机的int数字（范围是int的所有范围，有正负两种）： int num = r.nextInt()  
获取一个随机的int数字（参数代表了范围，左闭右开区间）：int num = r.nextInt(3)  
实际上代表的含义是；[0,3)，也就是 0~2。  
 \*/***public class** Student{  
 **public static void** main(String[] args) {  
 *// 没有范围的随机产生数字* Random r = **new** Random();  
 **int** num = r.nextInt();  
 System.***out***.println(**"产生的随机数是："** + num);  
  
 *// 产生有范围的随机数字* Random s = **new** Random();  
 **int** num1 = s.nextInt(10);  
 System.***out***.println(num1);  
 }  
  
}

* + - 1. 使用random生成1~n之间的随机数字

注意：如果在Ran写10，那么就是0~9，然而想要的是1~10，可以发现，整体+1即可。

、

**int** n = 5;  
Random r = **new** Random();  
**int** result = r.nextInt(n) + 1; *// 重点 +·1. 将[0,5) 变为了 [1,6) 及 [1,5];*System.***out***.println(result);

* + - 1. 猜数字小游戏

代码：用Scanner和Random方法，和循环处理。

Random r = **new** Random();  
**int** randdomNum = r.nextInt(100) + 1; *// [1,100]*Scanner sc = **new** Scanner(System.***in***);  
  
**while** (**true**){  
 System.***out***.print(**"请输入你猜的数字："**);  
 **int** guseeNum = sc.nextInt(); *//键盘输入猜测的数字* **if**(guseeNum>randdomNum){  
 System.***out***.println(**"输入的数值太大"**);  
 }**else if**(guseeNum<randdomNum){  
 System.***out***.println(**"输入的数字太小"**);  
 }**else**{  
 System.***out***.println(**"恭喜，猜对啦"**);  
 **break**;  
 }

* + - 1. 对象数组

要求：定义一个数组，用来存放三个Person对象。

数组有一个缺点：一旦创建，程序运行期间长度不可以在发生变化。

主函数： 类函数为Person省略

**public class** Person01 {  
 **public static void** main(String[] args) {  
 *// 首先定义一个长度为2的数组，用来存放两个Person类型的对象；* Preson[] array = **new** Preson[3];  
 *//定义两个对象* Preson one = **new** Preson(**"张"**,10);  
 Preson two = **new** Preson(**"王"**,20);  
 *// 将one的地址值赋值给0号元素；* array[0] = one;  
 array[1] = two;  
  
 System.***out***.println(array[0]); *// 地址值* System.***out***.println(array[1]); *// 地址值* System.***out***.println(array[0].getName()); *// 张* }  
}

* + - 1. ArrayList集合概述和基本使用

**package** demo01;  
**import** java.util.ArrayList; // 使用ArrayList时自己产生的导包代码；  
  
*/\*  
数组的长度不可以在发生变化。  
但是ArrayList集合的长度是可以随意变化的。  
  
对于ArrayList来说，有一个尖括号 <E> 代表泛型。  
泛型：也就是装在集合当中的所有元素，全都是统一的什么类型。  
注意：泛型只能是引用类型，不能是基本类型。  
  
注意事项：  
对于ArrayList集合来说，直接打印得到的不是地址值，而是内容。  
如果内容是空，得到的是空的中括号：[];  
 \*/***public class** Person01 {  
 **public static void** main(String[] args) {  
 *// 创建一个ArrayList 集合，集合的名称是List，里面装的全是String字符串类型的数组；  
 // 备注：从JDK 1.7+开始，右侧的尖括号内部可以不写内容，但是 <>本身还是要写的;* ArrayList<String> list = **new** ArrayList<>();  
 System.***out***.println(list); *// []  
  
 // 向集合当中添加一些数据，需要用到add方法。* list.add(**"张"**);  
 list.add(**"王"**);  
 list.add(**"程"**);  
 System.***out***.println(list); *// [张,王,程]  
 // list.add(100); 错误写法，因为创建的数组为String类型，不能添加int类型数据。* }  
}

* + - 1. ArrayList 集合的常用方法和使用   
         */\*  
         ArrayList当中常用的方法：  
         public boolean add(E e):向集合当中添加元素，参数的类型和泛型一致。  
         备注：对于ArrayList集合来说，add添加动作一定会成功的，所以返回值可用可不用。  
         但是对于其他集合（今后学习）来说，add添加动作不一定成功。  
           
         public E get(int index):从集合中获取元素，参数是索引编号，返回值就是对应位置的元素。  
         public E remove(int index):从集合当中删除元素，参数是索引编号，返回值就是被删掉的元素。  
         public int size():获取集合的尺寸长度，返回值是集合中包含的元素个数。  
          \*/***public class** Person01 {  
          **public static void** main(String[] args) {  
          *// 用ArrayList方法定义一个数组* ArrayList<String> list = **new** ArrayList<>();  
          *// 向集合中添加元素* **boolean** success = list.add(**"张"**);  
          System.***out***.println(list); *// [张]* System.***out***.println(success); *// true,代表插入成功* list.add(**"王"**);  
          list.add(**"程"**);  
          System.***out***.println(list); *// [张,王,程]  
          // 获取数组当中的第三个* System.***out***.println(list.get(2)); *// 程 ,获取并不是删除  
          // 删除数组中第二个元素，及王* list.remove(1);  
          *//String name = list.remove(1);  
          //System.out.println("被删除的人是：" + name);* System.***out***.println(list); *// [张,程]  
           
          // 获取集合的长度，也就是数组当中元素的个数* **int** size = list.size();  
          System.***out***.println(**"集合中的长度："** + size);  
          }  
         }

遍历数组：用list.size() 即可。

**int** size = list.size();  
System.***out***.println(**"集合中的长度："** + size);  
System.***out***.print(**"["**);  
**for** (**int** i = 0; i < list.size(); i++) {  
 System.***out***.print(list.get(i));  
 **if**(i< list.size()-1){  
 System.***out***.print(**","**);  
 }  
}  
System.***out***.println(**"]"**);

结果为：[张,程]

* + - * 1. ArrayList集合存储的基本数据

如果希望向集合ArrayList当中存储基本类型数据，必须使用基本类型对应的“包装类”;

基本类型 包装类（引用类型，包装类都位于Java.long包下）

byte Byte

short Short

int Integer 【特殊】

long Long

float Float

double Double

char Character 【特殊】

boolean Boolean

从JDK 1.5+ 开始，支持自动装箱、自动拆箱。

自动装箱：基本类型 --> 包装类型

自动拆箱：包装类型 --> 基本类型

主函数：

**public class** Person01{  
 **public static void** main(String[] args) {  
 ArrayList<String> listA = **new** ArrayList<>();  
 *// 错误写法！泛型只能是引用类型，不能是基本类型  
 // ArrayList<int> listB = new ArrayList<>();* ArrayList<Integer> listC = **new** ArrayList<>();  
 listC.add(10);  
 listC.add(20);  
 System.***out***.println(listC); *// [10,20]  
 // 其实数组里边的数据类型为int类型，直接赋值给int的num即可。* **int** num = listC.get(1);  
 System.***out***.println(**"数组中的第二个元素为:"** + num);  
 }  
}

* + - * 1. ArrayList练习一 存储随机数字

产生 1~33 之间的随机数字，组成一个数组。

ArrayList<Integer> list = **new** ArrayList<>();  
Random r = **new** Random();  
**for** (**int** i = 0; i < 6; i++) {  
 **int** num = r.nextInt(33) + 1;  
 list.add(num);

* + - * 1. ArrayList 练习 存储自定义形式

题目：自定义4个学生对象，添加到集合，并遍历。

主函数： 类函数省略

*// 1.自定义Student学生类，四个部分；（无参、全参、Getter / Setter）  
// 2.创建一个集合，用来存储学生对象。泛型：<Student>  
// 可以这个定义： ArrayList<自定义类名称> list = new ArrayList<>();*ArrayList<Student> list = **new** ArrayList<>();  
*// 3.根据类，创建4个学生对象。*Student one = **new** Student(**"张"**,10);  
Student two = **new** Student(**"王"**,20);  
*// 4.将4个学生添加到数组*list.add(one);  
list.add(two);  
*// 5.遍历数组，for循环***for** (**int** i = 0; i < list.size(); i++) {  
 Student stu = list.get(i);  
 System.***out***.println(**"姓名："** + stu.getName()+ **"，年龄："** + stu.getAge());  
}

* + - * 1. ArrayList 练习三 按指定形式输出数组

要求：输出的格式为 {元素@元素@元素}

System.out.println(list) // [10.20.30]

printArrayList(list) // {10@20@30}

主函数：

ArrayList<String> list = new ArrayList<>();

printArrayList(); // 调用方法函数

方法函数：

**public static void** printArrayList(ArrayList<String> list){ *//任何类型的数据都可以作为参数，用数组作为参数  
 // {10@20@30}* System.***out***.print(**"{"**);  
 **for** (**int** i = 0; i < list.size(); i++) {  
 String name = list.get(i);  
 **if**(i == list.size() - 1){  
 System.***out***.println(name + **"}"**);  
 }**else**{  
 System.***out***.print(name + **"@"**);  
 }  
 }  
}

* + - * 1. ArrayList 练习四 筛选集合中的数字

题目：用一个大集合存放20个随机数字，然后筛选出其中的偶数元素，放到小集合当中

要求使用自定义的方法实现筛选。

**package** demo01;  
  
**import** java.lang.reflect.Array;  
**import** java.util.ArrayList;  
**import** java.util.Random;  
  
**public class** Person01 {  
 **public static void** main(String[] args) {  
 *// 定义一个数组，存放大数组中的数据* ArrayList<Integer> bigList = **new** ArrayList<>();  
 *// 产生随机数字* Random r = **new** Random();  
 *// 产生20个随机数字* **for** (**int** i = 0; i < 20; i++) {  
 *// 给定产生随机数的范围 1~100* **int** num = r.nextInt(100) + 1; *// 1~100* bigList.add(num);  
 }  
 *// 定义一个小数组，把大数组中的偶数给smallList数组* ArrayList<Integer> smallList = *getSmallList*(bigList);  
 System.***out***.println(**"偶数共有："** + smallList.size());  
 **for** (**int** i = 0; i < smallList.size(); i++) {  
 System.***out***.println(smallList.get(i));  
 }  
 }  
  
 *// 这个方法，接收大集合参数，返回小集合结果* **public static** ArrayList<Integer> getSmallList(ArrayList<Integer> bigList) {  
 *// 创建一个小集合，用来装结果* ArrayList<Integer> smallList = **new** ArrayList<>();  
 **for** (**int** i = 0; i < bigList.size(); i++) {  
 **int** num = bigList.get(i);  
 *// 判断是不是为偶数，如果是就将他放在小数组中，及smallList数组中；* **if** (num % 2 == 0) {  
 smallList.add(num);  
 }  
 }  
  
 }  
}

* + - * 1. 字符串的概述和特点

java.long.String类代表字符串。

PAI当中说，java程序中的所有字符串字面值（如:”abc”）都作为此类的实例实现。

其实就是说：程序当中所有的双引号字符串，都是String类的对象。（就算没有new，也照样是）

字符串的特点：

字符串的内容永远不变。【重点】

正是因为字符串不可改变，所以字符串是可以共享使用的。

字符串效果上相当于char[]字符数组，但是底层原理是byte[]字节数组。

* + - * 1. 字符串的构造方法和直接创造

*创建字符串的常见3+1种方法:  
三种构造方法：  
public String():创建一个空白字符串，不含有任何内容。  
public String(char[] array):根据字符数组的内容，来创建对应的字符串。  
public String(byte[] array):根据字节数组的内容，来创建对应的字符串。  
一种直接创建：  
String str = "Hello"; //右边直接双引号  
  
 注意：直接写上双引号，就是字符串对象。  
 \*/***public class** Person01{  
 **public static void** main(String[] args) {  
 *// 使用空参数构造* String str1 = **new** String(); *// 小括号留空，说明字符串什么内容都没有。* System.***out***.println(**"第一个字符串为："** + str1); *// 空白  
  
 // 根据字符数组创建字符串* **char**[] charArray = {**'A'**,**'B'**,**'C'**};  
 String str2 = **new** String(charArray);  
 System.***out***.println(**"第二个字符串为："** + str2); *// ABC  
  
 // 根据字节数组创建字符串* **byte**[] byteArray = {97,98,99};  
 String str3 = **new** String(byteArray);  
 System.***out***.println(**"第三个字符串为： "** + str3); *// abc  
  
 // 直接创建* String str4 = **"Hello"**;  
 System.***out***.println(**"第四个字符窜为："** + str4);  
  
 }  
}

* + - * 1. 字符串的常量池

字符串常量池：程序当中直接写上的双引号字符串，就在字符串常量池中。

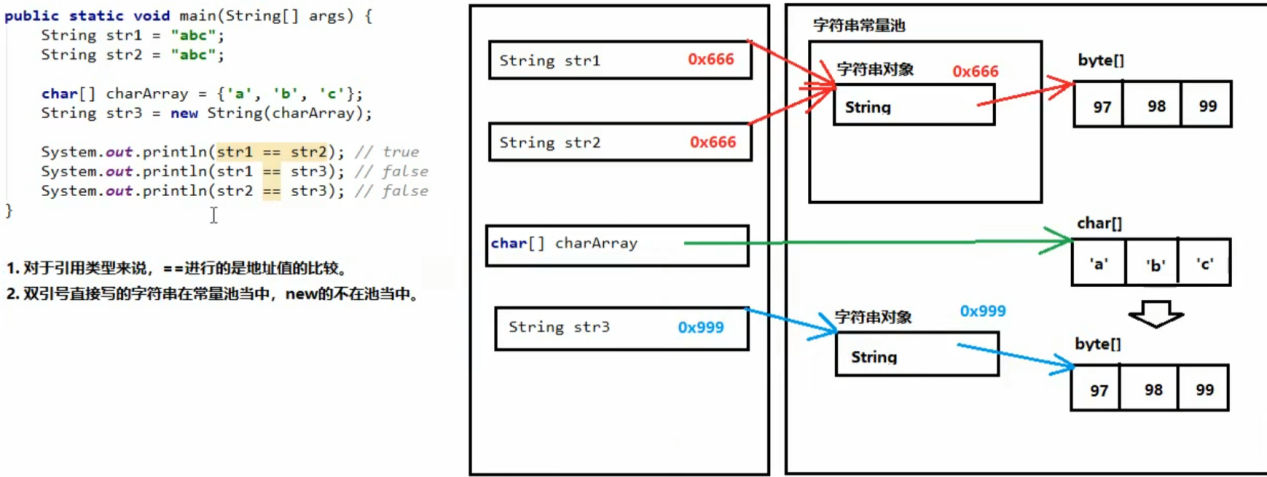
对于基本类型来说， ==是进行数值的比较。

对于引用类型来说， ==是进行【地址值】的比较。

主函数：

**public static void** main(String[] args) {  
 String str1 = **"abc"**;  
 String str2 = **"abc"**;  
  
 **char**[] charArray = {**'a'**,**'b'**,**'c'**};  
 String str3 = **new** String(charArray);  
  
 System.***out***.println(str1==str2); *// true 此处的 == 比较的为地址值* System.***out***.println(str2==str3); *// flase* System.***out***.println(str1==str3); *// flase*}

比较图示：



字符串的比较相关方法

*/\*  
==是进行对象的地址值比较，如果确实需要字符串的内容比较，可以用两个方法：  
  
public boolean equals(Object obj):参数可以是任何对象，只是参数是一个字符串并且内容相同的才会给true，否则返回flase。  
注意事项：  
1. 任何对象都能用Object进行接收。  
2. equals方法具有对称性，也就是a.equals(b)和b.equals(a)效果一样。  
3. 如果比较双方一个常量一个变量，推荐把常量字符串写在前面。  
推荐： "abc".equals(str) 不推荐： str.equals("abc")  
  
public boolean equalsIgnoreCase(String str): 忽略大小写，进行内容比较。  
 \*/***public class** Person01{  
 **public static void** main(String[] args) {  
 *// 定义字符串* String str1 = **"Hello"**;  
 String str2 = **"Hello"**;  
 **char**[] charArray = {**'H'**,**'e'**,**'l'**,**'l'**,**'o'**};  
 String str3 = **new** String(charArray);  
  
 *// 比较字符串* System.***out***.println(str1.equals(str2)); *// true* System.***out***.println(str2.equals(str3)); *// true* System.***out***.println(str3.equals(**"Hello"**)); *// true* System.***out***.println(**"Hello"**.equals(str3)); *// true* String str4 = **"hello"**;  
 System.***out***.println(str1.equals(str4)); *// false equals方法区分字母大小写* String str5 = **"hello"**;  
 System.***out***.println(str5.equalsIgnoreCase(str1)); *// true equalsIgnoreCase方法不区分字母大小写；  
 // 注意：只有英文字母区分大小写，其他都不区分大小写* System.***out***.println(**"abc一123"**.equalsIgnoreCase(**"abc壹123"**)); *// 错误写法；* }  
}

* + - * 1. 字符串的获取相关方法

*\*  
String当中与获取相关的常用方法：  
  
public int Length(): 获取字符串当中含有的字符个数，拿到字符串长度。  
public String concat(String str): 将当前字符串和参数字符串拼接成为返回值的新字符串。  
public char charAt(int index): 获取指定索引位置的单个字符。（索引从0开始）  
public int indexOf(String str): 查找参数字符串在本字符串当中首次出现的索引位置，如果没有返回-1值。  
 \*/***public class** Person01{  
 **public static void** main(String[] args) {  
 *// 获取字符串的长度* **int** length = **"hdiahsidhaisoh"**.length();  
 System.***out***.println(**"字符串的长度为："** + length);  
  
 *// 拼接字符串* String str1 = **"Hello"**;  
 String str2 = **"Word"**;  
 String str3 = str1.concat(str2);  
 System.***out***.println(str1); *// Hello 原封不动* System.***out***.println(str2); *// Word 原封不动* System.***out***.println(str3); *// HelloWord, 新的字符串  
  
 // 获取指定索引位置的单个字符* **char** ch = **"Hello"**.charAt(1);  
 System.***out***.println(**"在1号索引位置的字符是："** + ch);  
  
 *// 查找参数字符串在本来字符串当中出现的第一次索引位置  
 // 如果根本没有，返回-1值* String original = **"HelloWordHelloWord"**;  
 **int** index = original.indexOf(**"llo"**);  
 System.***out***.println(**"第一次索引值是："** + index); *// 2* System.***out***.println(**"HelloWord"**.indexOf(**"abc"**)); *// -1* }  
}

* + - * 1. 字符串的截取方法

*/\*  
字符串的截取方法：  
  
public String substring(int index): 截取从参数位置一直到字符串末尾，返回新字符串。  
public String substring(int begin,int end): 截取从begin开始，一直到end结束，中间的字符串。  
备注：[begin,end) ，包含左边，不包含右边。  
\*/***public class** Person01{  
 **public static void** main(String[] args) {  
 *// 截取从开始到最后* String str1 = **"HelloWord"**;  
 String str2 = str1.substring(5);  
 System.***out***.println(str1); *// HelloWord 截取时原字符串不改变* System.***out***.println(str2); *// Word 截取的新字符串  
 // 截取指定范围的字符串* String str3 = str1.substring(4,7);  
 System.***out***.println(str3); *// owo  
 // 下面这种写法，字符串的内容仍然是没有改变的  
 // 下面有两个字符串："HEllo", "Java"  
 // StrA当中保存的是地址值  
 // 本来地址值是Hello的0x666  
 // 后来地址值变成了Java的0x999* String strA = **"Hello"**;  
 System.***out***.println(strA); *// Hello* strA = **"Java"**;  
 System.***out***.println(strA); *// Java* }  
 }

* + - * 1. 字符串的转换相关方法

*/\*  
String 当中的与转换相关的常用方法：  
  
public char[] toCharArray()： 将当前字符串拆分成为字符数组作为返回值。  
public byte[] getBytes(): 获取当前字符串底层的字节数组。  
public String replace(CharSequence oldString , CharSequence newString);  
将所有出现的老字符串替换为新的字符串，返回替换之后的结果新字符串。  
备注：CharSequence意思就是说可以接受字符串类型。  
\*/***public class** Person01{  
 **public static void** main(String[] args) {  
 *// 转换成为字符数组* **char**[] chars = **"Hello"**.toCharArray();  
 System.***out***.println(chars[0]); *// H* System.***out***.println(chars.**length**); *// 5  
  
 // 转换成为字节数组* **byte**[] bytes = **"abc"**.getBytes();  
 **for** (**int** i = 0; i < bytes.**length**; i++) {  
 System.***out***.println(bytes[i]);  
 }  
 *// 字符串的内容替换* String str1 = **"How do you do ?"**;  
 String str2 = str1.replace(**"o"**,**"\*"**);  
 System.***out***.println(str1); *// How do you do ？* System.***out***.println(str2); *// H\*w d\* y\*u d\* ?* }  
 }

* + - * 1. 字符串的分割方法

**package** demo01;  
*/\*  
分割字符串：  
public String[] splita(String regex): 按照参数的规则，将字符串切分成若干部分。  
注意事项：  
split方法的参数其实是一个"正则表达式"，今后学习。  
今天要注意：如果按照英文句点“ . “ 进行切分，必须要写 ” \\. “（两个反斜杠）  
\*/***public class** Person01{  
 **public static void** main(String[] args) {  
 String str1 = **"aaa,bbb,ccc"**;  
 String[] array1 = str1.split(**","**); *// 代表以“ ，”作为分割点，分为三段* **for** (**int** i = 0; i < array1.**length**; i++) {  
 System.***out***.println(array1[i]); *// aaa bbb ccc* }  
 String str2 = **"aaa.bbb.ccc"**;  
 String[] array2 = str2.split(**"."**);  
 **for** (**int** i = 0; i < array2.**length**; i++) {  
 System.***out***.println(array2[i]); *// 没有结果* }  
 *// 注意：以“ . ” 为分割点，要替换为这个形式 “ \\. ”;* String str3 = **"aaa.bbb.ccc"**;  
 String[] array3 = str3.split(**"\\."**);  
 **for** (**int** i = 0; i < array3.**length**; i++) {  
 System.***out***.println(array3[i]); *// aaa bbb ccc* }  
 }  
 }

* + - * 1. 字符串练习一 按指定格式拼接字符

题目：

定义一个方法，把数组[1,2,3]按照指定格式拼接成一个字符串，格式参考如下：[word1word2word3].

分析：

1. 首先准备一个int[] 数组，内容是：1、2、3；
2. 定义一个方法，用来将数组变成字符串。

三要素：

返回值类型：String；

方法名称：fromArrayToString；

参数列表：int[]；

1. 格式：[word1word2wprd3]

用到：for循环、字符串拼接、每个数组元素之前都有一个word字样、分隔使用的是#、区分下一个是不是最后一个。

1. 调用方法：得到返回值，并打印结果字符串。

**public class** Person01{  
 **public static void** main(String[] args) {  
 **int**[] array = {1,2,3};  
 String result = *fromArrayToString*(array);  
 System.***out***.println(result);  
  
 }  
 **public static** String fromArrayToString(**int**[] array){  
 String str = **"["**;  
 **for** (**int** i = 0; i < array.**length**; i++) {  
 **if**( i ==array.**length** - 1 ) {  
 str += **"word"** + array[i] + **"]"**; // 可以进行字符之间的相加  
 }**else**{  
 str += **"word"** + array[i] + **"#"**;  
 }  
 }  
 **return** str;  
 }  
}

* + - * 1. 字符串的练习二 统计一个字符串中各样字符出现的次数

**package** demo01;  
  
**import** java.util.Scanner;  
  
*/\*  
题目：  
键盘输入一个字符串，并且统计其中各种字符串出现的次数。  
种类有：大写字母、小写字母、数字、其他  
  
思路：  
1. 既然用到键盘输入，肯定是Scanner  
2. 键盘输入的字符串，那么：String str = sc.next();  
3. 定义四个变量，分别代表四种字符串各自的出现次数。  
4. 需要对字符串一个字、一个字检查，String --> char[]，方法就是toCharArray();  
5. 遍历char[]字符数组，对于当前字符的种类进行判断，并且用四个变量进行++动作。  
6. 打印输出四个变量，分别代表四种字符出现的次数。  
\*/***public class** Person01{  
 **public static void** main(String[] args){  
 *// 1.用户输入字符串* Scanner sc = **new** Scanner(System.***in***); *// 使用 Scaner 的三步;* System.***out***.print(**"请输入一个字符串："** );  
 String input = sc.next();  
  
 *// 2.定义几个变量，用于存储各样的字符* **int** countUpper = 0; *// 大写字母* **int** countLower = 0; *// 小写字母* **int** countNumber = 0; *// 数字* **int** countOther = 0; *// 其他  
  
 // 3.将字符串拆分为单个字符 使用toCharArray() 方法* **char**[] charArray = input.toCharArray();  
  
 *// 4. 使用循环，将字符串拆分为单个字符* **for** (**int** i = 0; i < charArray.**length**; i++) {  
 **char** ch = charArray[i]; *// ch代表单个字符* **if**(**'A'**<= ch && ch <= **'Z'**){ *// 只要 char 类型，只要进行运算就转换为数字之间进行比较。* countUpper++;  
 }**else if**(**'a'**<= ch && ch <= **'z'**){  
 countLower++;  
 }**else if**(**'0'** <= ch && ch <= **'9'**){  
 countNumber++;  
 }**else**{  
 countOther++;  
 }  
 }  
 *// 5. 打印结果* System.***out***.println(**"大写字母有："** + countUpper);  
 System.***out***.println(**"小写字母有："** + countLower);  
 System.***out***.println(**"数字有："** + countNumber);  
 System.***out***.println(**"其他字符有："** + countOther);  
  
 }  
}

* + - * 1. 静态static关键字概述



* + - * 1. 静态Static关键字修饰成员

如果一个成员变量使用了Static关键字，那么这个变量不再属于对象自己，而是属于所在的类，多对象共享同一份数据。

类函数;

**private** String **name**; *// 姓名***private int age**; *// 年龄***static** String *room*; *// 所在教室***private int id**;  
**private static int** *idCounter* = 0; *// 学号计数器，每当new了一个新的对象的时候，计数器++；***public** Person() {  
 **this**.**id** = ++*idCounter*; *// 先++，就可以保证学号从1开始。*}  
  
**public** Person(String name, **int** age) {  
 **this**.**name** = name;  
 **this**.**age** = age;  
 **this**.**id** = ++*idCounter*; *// 先++，就可以保证学号从1开始。*}  
  
**public int** getId() {  
 **return id**;  
}  
  
**public void** setId(**int** id) {  
 **this**.**id** = id;  
}

主函数：

**public class** Person01 {  
 **public static void** main(String[] args) {  
 Person one = **new** Person(**"王"**,10);  
 one.*room* = **"101教室"**;  
 System.***out***.println(**"姓名："** + one.getName()  
 + **"，年龄："** + one.getAge() + **"，教室："** + one.*room* + **"，学号："** + one.getId());  
  
 Person two = **new** Person(**"程"**,20);  
 System.***out***.println(**"姓名："** + two.getName()  
 + **"，年龄："** + two.getAge() + **"，教室："** + two.*room* + **"，学号："** + two.getId());  
 *// 姓名：王，年龄：10，教室：101教室，学号：1  
 //姓名：程，年龄：20，教室：101教室，学号：2  
 // 备注：此处 room 在类函数中声明为 Static 类型，所以整个类函数都可以使用。* }  
}

* + - * 1. 静态static关键字修饰成员

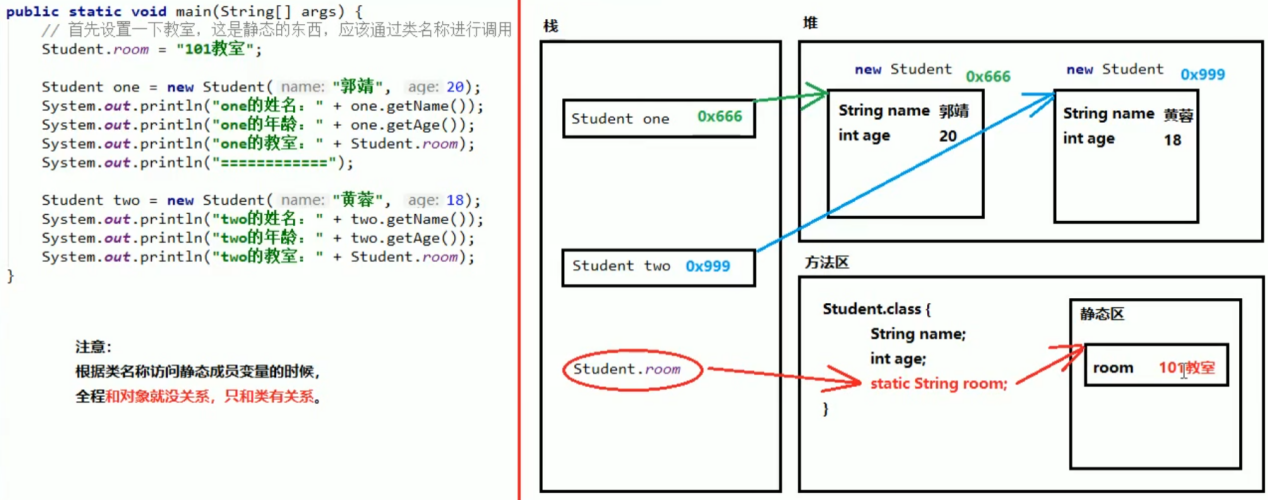
**package** demo01;  
*/\*  
一旦使用static修饰成员的方法，那么这就是成为了静态方法。静态方法不属于对象，而是属于类的。  
  
如果没有static关键字，那么必须首先创建对象，然后通过对象才能使用它。  
如果有了static关键字，那么不需要创建对象，直接就能通过类名称来使用它。  
  
无论是成员变量，还是成员方法。如果有了static，都推荐使用类名称进行调用。  
静态变量： 类名称.静态变量;  
静态方法： 类名称.静态方法();  
  
注意事项：  
1. 静态不能直接访问非静态。  
原因：因为在内存dangzh是【先】有的静态内容，【后】有的非静态内容。  
2. 静态方法当中不能用this。  
原因：this代表当前对象，通过谁调用的方法，谁就是当前对象。  
 \*/*

***主函数：*****public class** Person01 {  
 **public static void** main(String[] args) {  
 *// 首先创建对象* Person obj = **new** Person();  
 *// 然后才能使用没有static关键字的内容* obj.method();  
  
 *// 对于静态方法来说，可以通过对象名进行调用，也可以直接通过类名称来调用。* obj.*methodStatic*(); *// 正确，不推荐，这种写法在编辑之后也会被javac翻译成为”类名称.静态方法名”* Person.*methodStatic*(); *// 正确，推荐  
  
 // 对于本来当中的静态方法，可以省略类名称  
 myMethod*();  
 Person01.*myMethod*(); *// 完全等效* }  
 **public static void** myMethod(){  
 System.***out***.println(**"自己的方法！"**);  
 }  
}

类函数：

**public class** Person {  
 **int num**; *// 成员变量* **static int** *numStatic*; *// 静态变量  
  
 // 成员方法* **public void** method(){  
 System.***out***.println(**"这是一个成员方法。"**);  
 *// 成员方法可以访问成员变量* System.***out***.println(**num**);  
 *// 成员方法可以访问静态变量* System.***out***.println(*numStatic*);  
 }  
 *// 静态方法* **public static void** methodStatic(){  
 System.***out***.println(**"这是一个静态方法。"**);  
 *// 静态方法可以访问静态变量* System.***out***.println(*numStatic*);  
 *// 静态不能直接访问非静态【重点】  
 // System.out.println(num); // 错误写法！* }  
}

* + - * 1. 静态static的内存图



* + - * 1. 静态代码块 *静态代码块的格式是：  
           public class 类名称{  
            static {  
            // 静态代码块的内容  
            }  
           }  
           特点：当第一次用到本类时，静态代码块执行唯一的一次。  
           静态内容总是优先于非静态，所以静态代码块比方法构造方法先执行。  
             
           静态代码块的典型用途：  
           用来一次性地对静态成员变量进行赋值。  
            \*/*

类函数：

**public class** Person {  
 **static** {  
 System.***out***.println(**"静态代码块执行！"**);  
 }  
 **public** Person(){  
 System.***out***.println(**"构造方法执行！"**);  
 }  
}

主函数：**public class** Person01 {  
 **public static void** main(String[] args) {  
 Person one = **new** Person();  
 Person two = **new** Person();  
 *// 结果为：static只执行一次，但是方法执行new的次数。  
 // 静态代码块执行！  
 // 构造方法执行！  
 // 构造方法执行！*

}

}

* + - * 1. 数组工具类Arrays

**package** demo01;  
  
**import** java.util.Arrays;  
  
*/\*  
java.util.Arrays是一个与数组相关的工具类，里面提供了大量的静态方法，用来实现数组常见的操作。  
  
public static String toString(数组): 将参数数组变成字符串（按照默认格式：[元素1,元素2,元素3]  
public static void sort(数组): 按照默认升序（从小到大）,对数组的元素进行排序。  
备注：  
1. 如果是数值，sort默认按照升序从小到大  
2. 如果是字符串，sort默认按照字母升序  
3. 如果是自定义的类型，那么这个自定义的类需要有Comparable或者Comparator接口的支持。（今后学习）  
 \*/*

主函数：**public class** Person01 {  
 **public static void** main(String[] args) {  
 *// 定义一个int类型的数组* **int**[] intArray = {10,20,30};  
 *// 将int[]数组按照默认格式变成字符串* String inStr = Arrays.*toString*(intArray);  
 System.***out***.println(inStr); *// [10, 20, 30]  
  
 // 定义一个数字类型的数组，进行排序* **int**[] array1 = {1,3,2,4};  
 Arrays.*sort*(array1);  
 System.***out***.println(Arrays.*toString*(array1)); *// [1, 2, 3, 4]  
  
 // 定义一个字母的数组，进行排序* String[] array2 = {**"bbb"**,**"aaa"**,**"ccc"**};  
 Arrays.*sort*(array2);  
 System.***out***.println(Arrays.*toString*(array2)); *// [aaa, bbb, ccc]* }  
}

* + - * 1. Arrays练习：字符串倒序

题目：

请使用Arrays相关的API，将一个随机字符串中的所有字符串升序排列，并倒序打印。

【重点】：5.fori 可以快速建立 ++ 循环 ， 5.forr 可以快速建立 -- 循环。

**package** demo01;  
**import** java.util.Arrays;

**public class** Person01{  
 **public static void** main(String[] args) {  
 *// 1. 定义一个字符串* String str = **"akfd8s7f8sdjfds97f7d8ufd9"**;  
 *// 2.使用sort方法，先建立一个字符数字，Arrays.sort方法只能在数组当中使用  
 // 3.String --> 单个字符数组，用toCharArray* **char**[] chars = str.toCharArray();  
 *// 4.对数组进行升序排序* Arrays.*sort*(chars);  
  
 *// 5. 需要倒序遍历，用--即可 使用 chars.forr 可以直接生成。* **for** (**int** i = chars.**length** - 1; i >= 0; i--) {  
 System.***out***.println(i);  
 }   
 }  
}

* + - * 1. 数学工具类Math

**package** demo01;  
*/\*  
java.util.Math类是数学相关的工具类，里面提供了大量的静态方法，完成与数学运算相关的操作。  
  
public static double abs(double num): 获取绝对值。  
public static double ceil(double num): 向上取整。  
public static double floor(double num): 向下取整。  
public static long round(double num): 四舍五入。  
  
Math.PI代表近似的圆周率。  
 \*/***public class** Person01{  
 **public static void** main(String[] args) {  
 *// 获取绝对值* System.***out***.println(Math.*abs*(3.14)); *// 3.14* System.***out***.println(Math.*abs*(-2.5)); *// 2.5  
  
 // 向上取整* System.***out***.println(Math.*ceil*(3.9)); *// 4.0* System.***out***.println(Math.*ceil*(3.0)); *// 3.0  
  
 // 向下取整,抹零* System.***out***.println(Math.*floor*(30.1)); *// 30.0* System.***out***.println(Math.*floor*(30.9)); *// 30.0  
  
 // 四舍五入* System.***out***.println(Math.*round*(20.4)); *// 20* System.***out***.println(Math.*round*(20.5)); *// 21* }  
}

* + - * 1. Math练习：小学数学真题

题目：

计算在 -10.5到5.9之间，绝对值大于6或者小于2.1的整数有多少个？

**package** demo01;  
  
**public class** Person01 {  
 **public static void** main(String[] args) {  
 **double** min = -10.8;  
 **double** max = 5.9;  
 **int** CountNum = 0;  
 *// 这样处理，变量i就是区间内所有的整数* **for** (**int** i = (**int**) min; i < max; i++) {  
 **int** abs = Math.*abs*(i); *// 绝对值* **if** (abs > 6 || abs < 2.1) {  
 System.***out***.println(i);  
 CountNum++;  
 }  
 }  
 System.***out***.println(**"总共有"** + CountNum + **"个整数！"**);  
 }  
}