

非卖款，0元入学，不1万就业不给1分钱学费，我们已于四年了！

笔记总链接：<http://bbs.itheima.com/thread-200600-1-1.html>

一、Java语言基础组成-Part 3

1.7 程序流程控制

1.7.3 循环结构

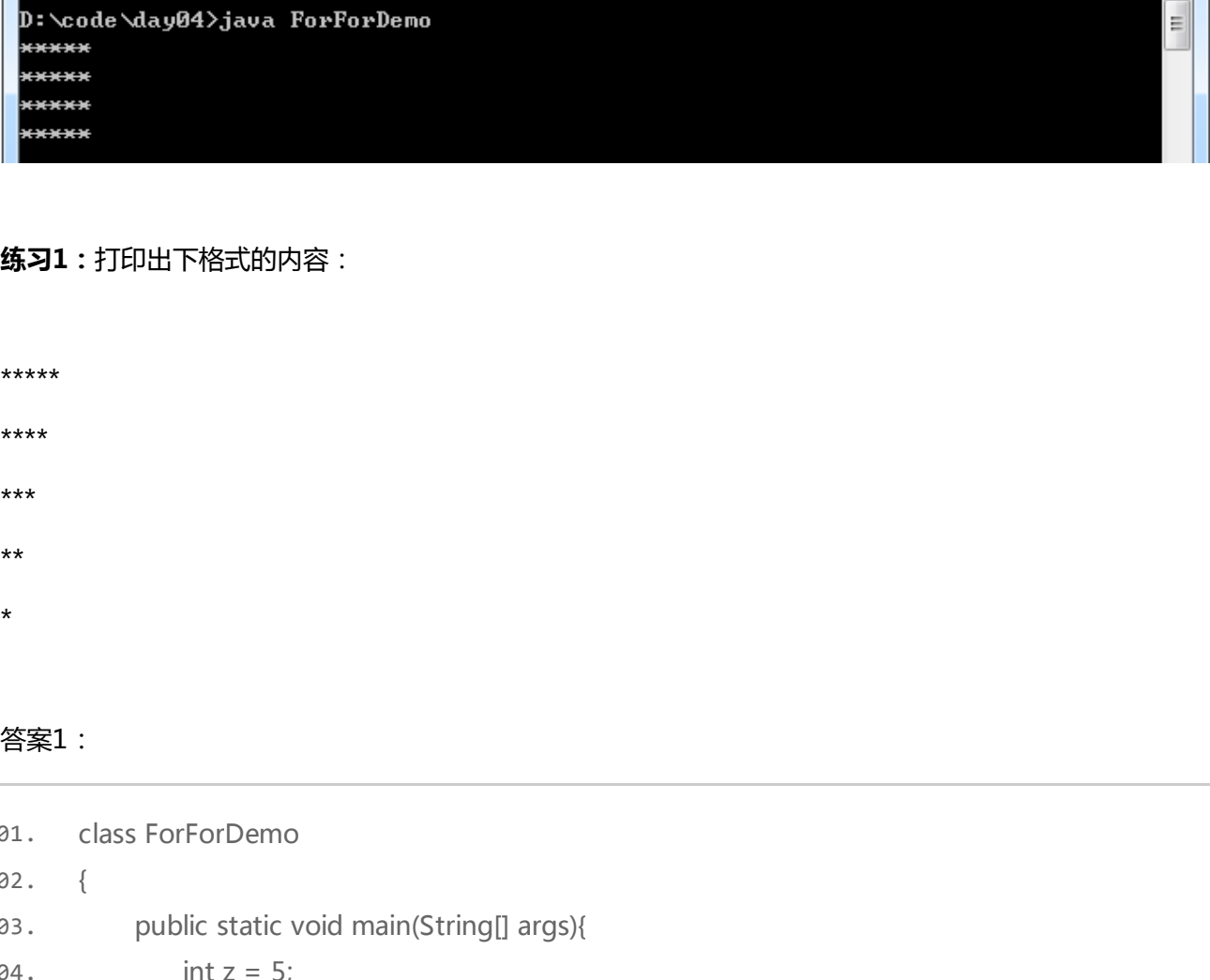
For循环嵌套

示例1：

```
01. class ForForDemo
02. {
03.     public static void main(String[] args){
04.         for(int x = 0; x < 3; x++){
05.             for(int y = 0; y < 4; y++){
06.                 System.out.println("ok" );
07.             }
08.         }
09.     }
10. }
```

复制代码

运行结果：

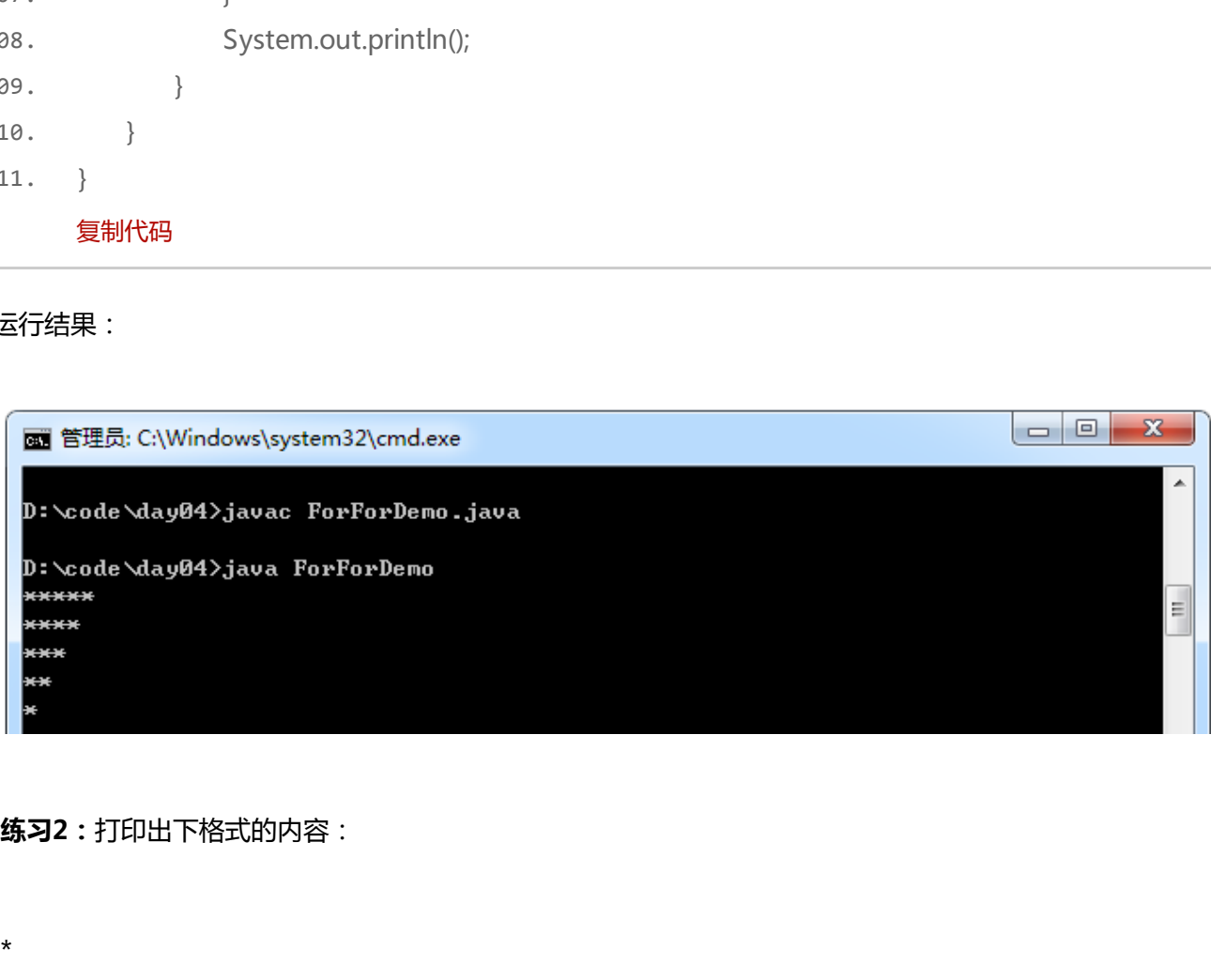


示例2：

```
01. class ForForDemo
02. {
03.     public static void main(String[] args){
04.         for(int x = 0; x < 4; x++){ //外循环控制的是行数
05.             for(int y = 0; y < 5; y++){ //内循环控制的是每一行的个数
06.                 System.out.print("*" );
07.             }
08.             System.out.println();
09.         }
10.     }
11. }
```

复制代码

运行结果：



练习1：打印出下格式的内容：

```
*****
****
***
**
*

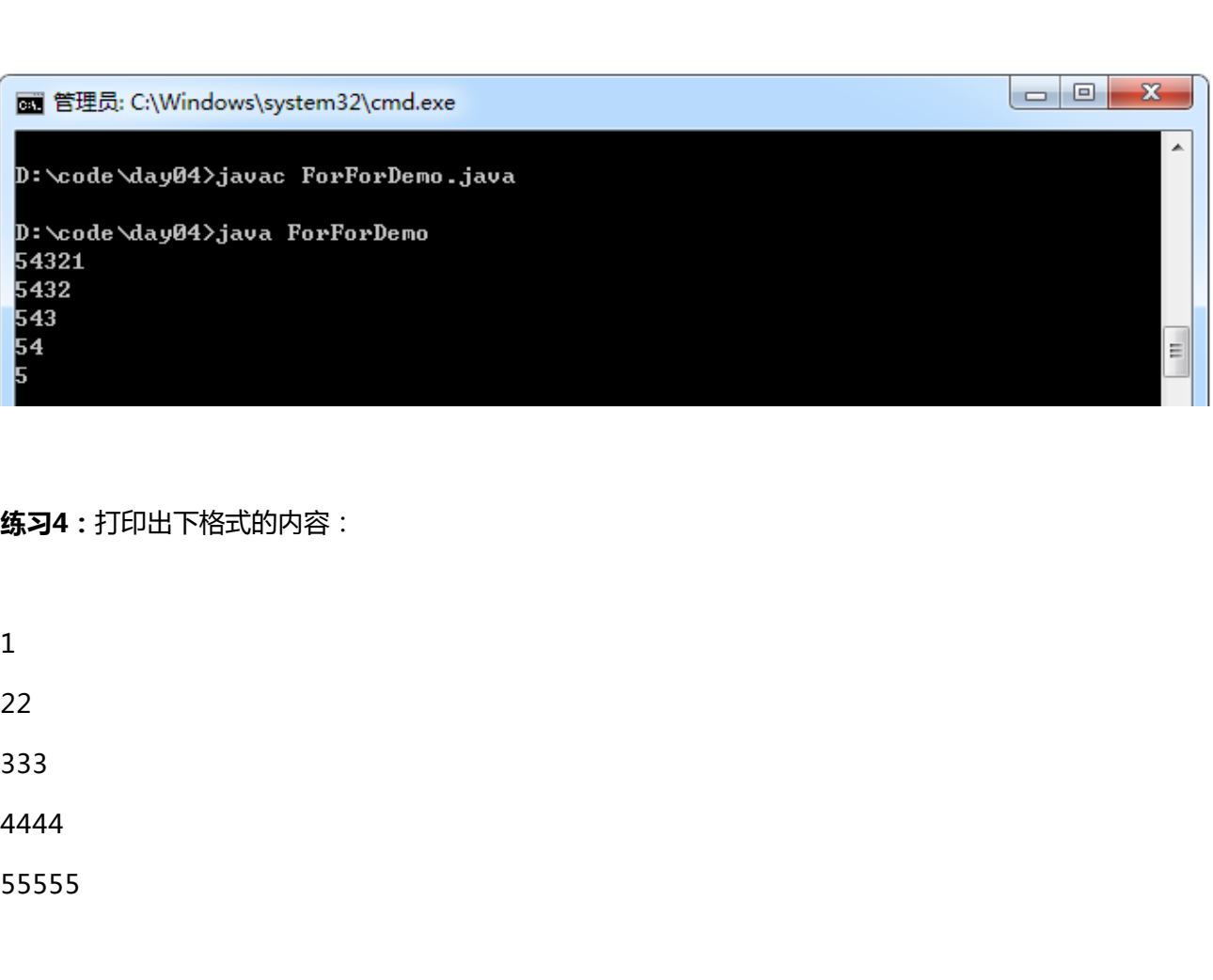
```

答案1：

```
01. class ForForDemo
02. {
03.     public static void main(String[] args){
04.         int z = 5;
05.         for(int x = 0; x <= 4; x++){
06.             for(int y = 1; y <= 5 - x; y++){
07.                 System.out.print("*" );
08.             }
09.             System.out.println();
10.             z--;
11.         }
12.     }
13. }
```

复制代码

运行结果：

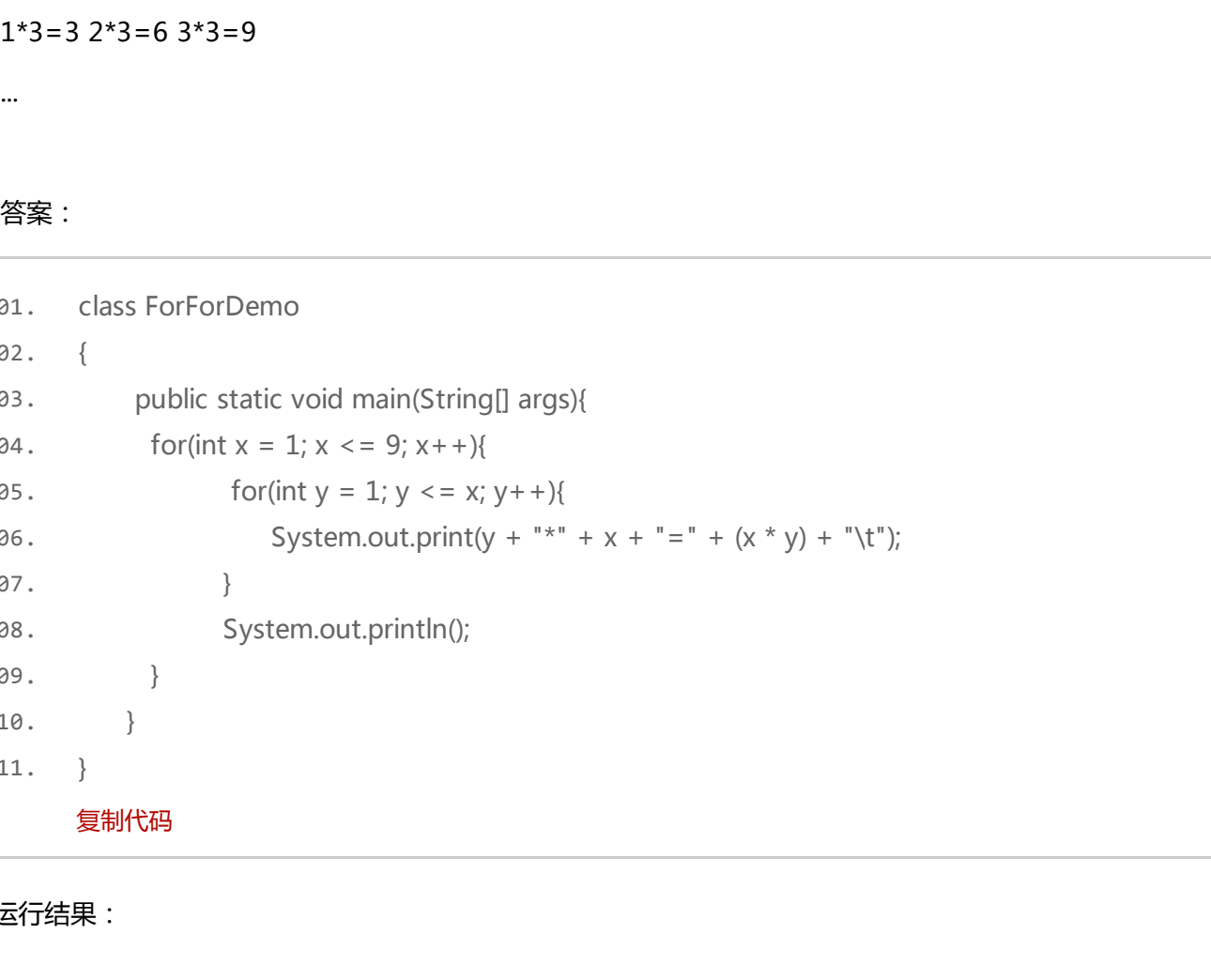


答案2：

```
01. class ForForDemo
02. {
03.     public static void main(String[] args){
04.         for(int x = 1; x <= 5; x++){
05.             for(int y = x; y <= 5; y++){
06.                 System.out.print("*" );
07.             }
08.             System.out.println();
09.         }
10.     }
11. }
```

复制代码

运行结果：



练习2：打印出下格式的内容：

```
*
**
***
****
*****

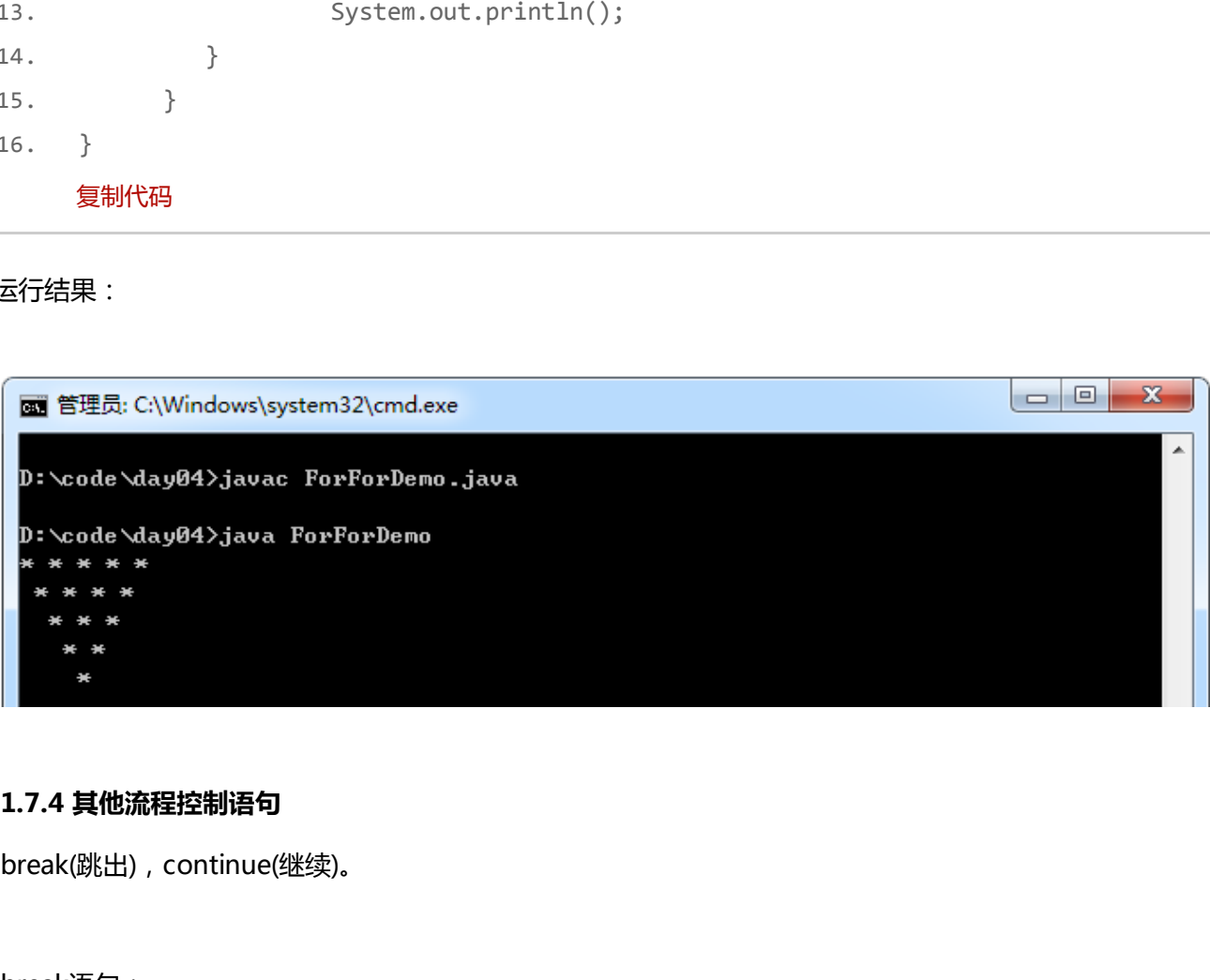
```

答案：

```
01. class ForForDemo
02. {
03.     public static void main(String[] args){
04.         for(int x = 1; x <= 5; x++){
05.             for(int y = 1; y <= x; y++){
06.                 System.out.print(" " );
07.             }
08.             System.out.println();
09.         }
10.     }
11. }
```

复制代码

运行结果：



练习3：打印出下格式的内容：

```
54321
5432
543
54
5

```

答案：

```
01. class ForForDemo
02. {
03.     public static void main(String[] args){
04.         for(int x = 1; x <= 5; x++){
05.             for(int y = 5; y >= x; y--){
06.                 System.out.print(y);
07.             }
08.             System.out.println();
09.         }
10.     }
11. }
```

复制代码

运行结果：



练习5：打印九九乘法表成如下形式：

```
1*1=1
1*2=2 2*2=4
1*3=3 2*3=6 3*3=9
...

```

答案：

```
01. class ForForDemo
02. {
03.     public static void main(String[] args){
04.         for(int x = 1; x <= 9; x++){
05.             for(int y = 1; y <= x; y++){
06.                 System.out.print(y + "*" + x + "=" + (x * y) + "\t");
07.             }
08.             System.out.println();
09.         }
10.     }
11. }
```

复制代码

运行结果：

P.S.

- 1、代码中的“\t”是一个转义字符，也就是制表符。还有其他的一些转义字符：\n回车，\b退格，\r回车符。
- 2、windows系统中回车符其实是由两个转义字符组成的：\r\n，linux中回车符是\n。

例子：

打印"hello world"：

```
System.out.println("\hello word\");
```

打印\hello world\：

```
System.out.println("\\hello word\\");
```

练习6：打印出下格式的内容：

```
*****
****
***
**
*

```

答案：

```
01. class ForForDemo
02. {
03.     public static void main(String[] args){
04.         for(int x = 1; x <= 5; x++){
05.             //首先打印出前面的空格
06.             for(int y = 1; y < x; y++){
07.                 System.out.print(" " );
08.             }
09.             //再打印出*
10.             for(int z = x; z <= 5; z++){
11.                 System.out.print("* " );
12.             }
13.             System.out.println();
14.         }
15.     }
16. }
```

复制代码

运行结果：

1.7.4 其他流程控制语句

break(跳出)，continue(继续)。

break语句：

应用范围：选择结构和循环结构。

示例1：

```
01. class BreakDemo
02. {
03.     public static void main(String[] args){
04.         for(int x = 0; x < 3; x++){
05.             System.out.println("x = " + x);
06.             break;
07.         }
08.     }
09. }
```

复制代码

运行结果：

示例2：

```
01. class BreakDemo
02. {
03.     public static void main(String[] args){
04.         for(int x = 0; x < 3; x++){
05.             if(x == 1)
06.                 break;
07.             System.out.println("x = " + x);
08.         }
09.     }
10. }
```

复制代码

运行结果：

示例3：

```
01. class BreakDemo
02. {
03.     public static void main(String[] args){
04.         for(int x = 0; x < 3; x++){
05.             for(int y = 0; y < 4; y++){
06.                 System.out.println("x = " + x);
07.                 break;
08.             }
09.         }
10. }
```

复制代码

运行结果：

P.S.

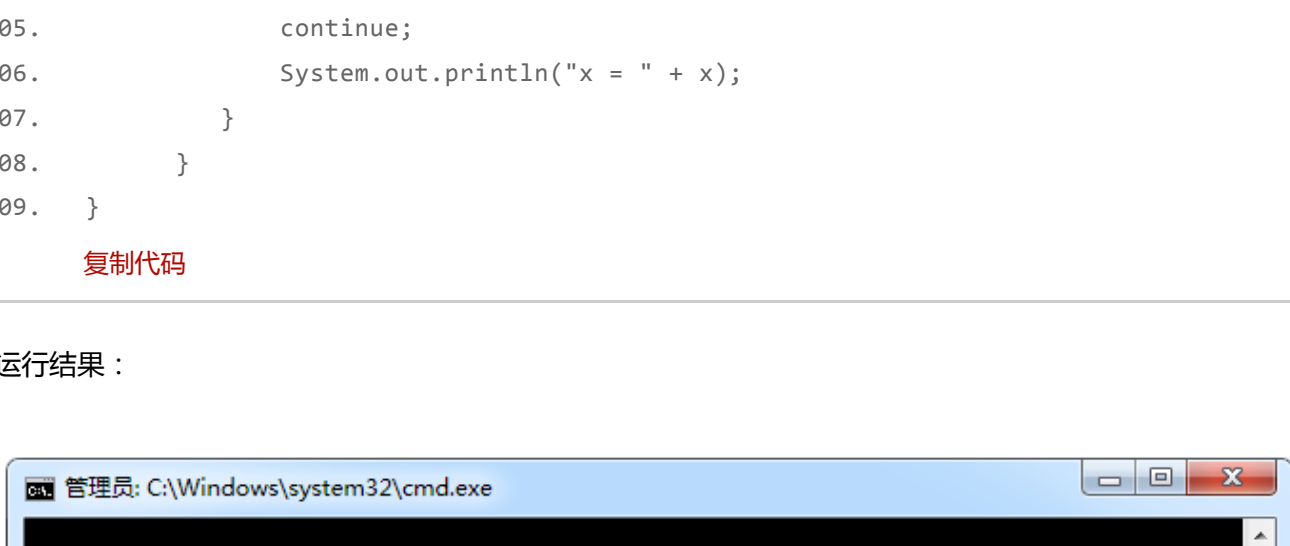
- 1、这两个语句离开应用范围，存在是没有意义的。
- 2、这两个语句单独存在，下面都可以有语句，因为执行不到。

示例1：

```
01. class BreakDemo
02. {
03.     public static void main(String[] args){
04.         for(int x = 0; x < 3; x++){
05.             break;
06.             System.out.println("x = " + x);
07.         }
08.     }
09. }
```

复制代码

运行结果：

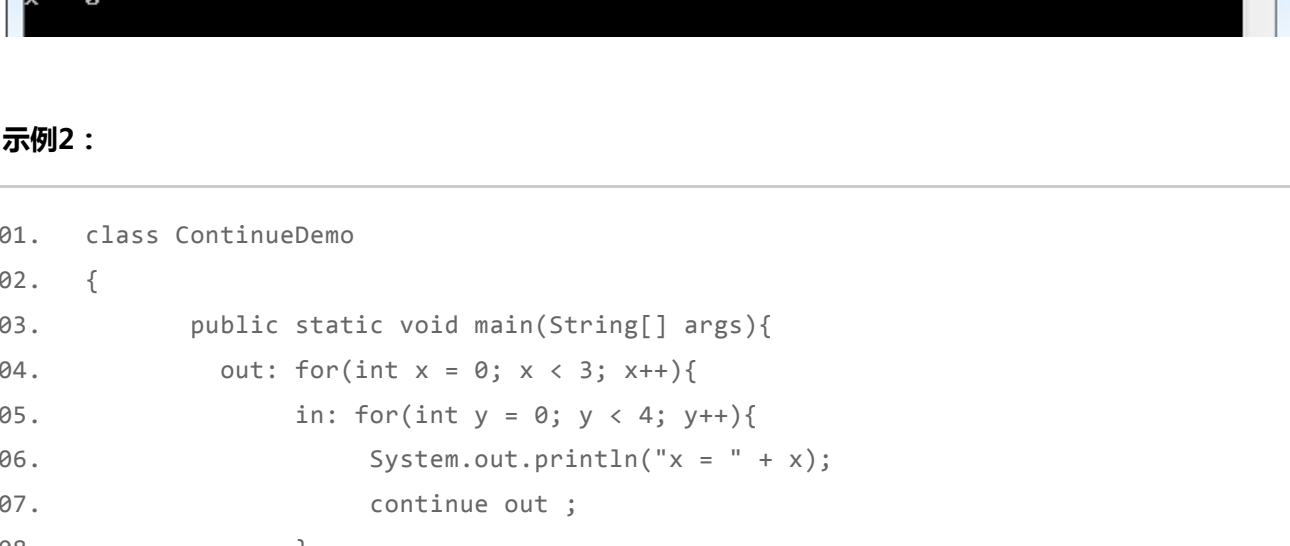


示例2：

```
01. class ContinueDemo
02. {
03.     public static void main(String[] args){
04.         for (int x = 0; x < 3; x++){
05.             continue;
06.             System.out.println("x = " + x);
07.         }
08.     }
09. }
```

复制代码

运行结果：



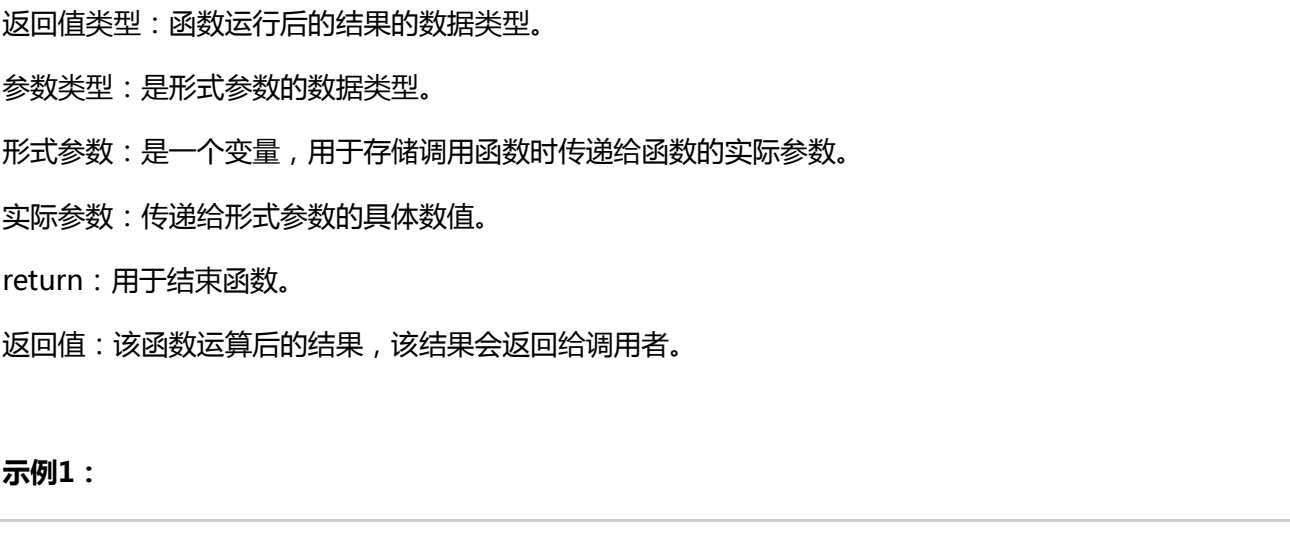
3、标号的出现，可以让这两个语句作用于指定的范围。

示例1：

```
01. class BreakDemo
02. {
03.     public static void main(String[] args){
04.         out:for(int x = 0; x < 3; x++){
05.             in: for(int y = 0; y < 4; y++){
06.                 System.out.println("x = " + x);
07.                 break out ;
08.             }
09.         }
10.     }
11. }
```

复制代码

运行结果：

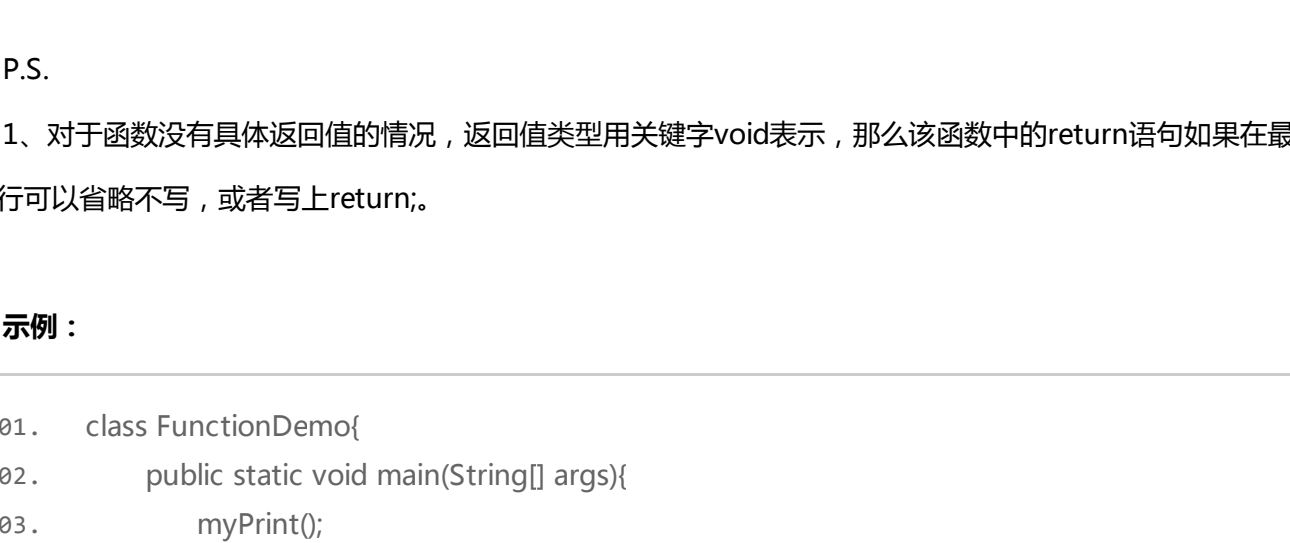


示例2：

```
01. class ContinueDemo
02. {
03.     public static void main(String[] args){
04.         out: for(int x = 0; x < 3; x++){
05.             in: for(int y = 0; y < 4; y++){
06.                 System.out.println("x = " + x);
07.                 continue out ;
08.             }
09.         }
10.     }
11. }
```

复制代码

运行结果：



1.8 函数

1.8.1 函数的定义

函数就是定义在类中的具有特定功能的一段独立小程序，函数也称为方法。

函数的格式：

修饰符 返回值类型 函数名(参数类型 形式参数1，参数类型 形式参数2，...)

```
{
    执行语句;
    return 返回值;
}
```

返回值类型：函数运行后的结果的数据类型。

参数类型：是形式参数的数据类型。

形式参数：是一个变量，用于存储调用函数时传递给函数的实际参数。

实际参数：传递给形式参数的具体数值。

return：用于结束函数。

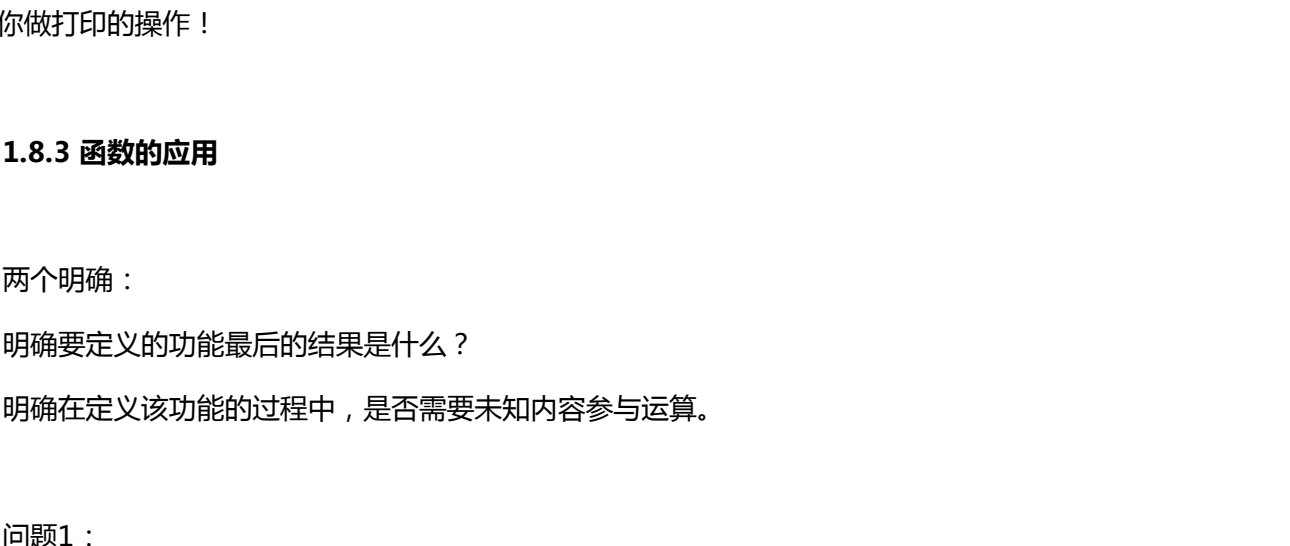
返回值：该函数运算后的结果，该结果会返回给调用者。

示例1：

```
01. class FunctionDemo{
02.     public static void main(String[] args){
03.         int c = add(3,4);
04.         System.out.println("c = " + c);
05.     }
06.
07.     public static int add(int a, int b){
08.         return a + b;
09.     }
10. }
```

复制代码

运行结果：



1.8.2 函数的特点

1、定义函数可以将功能代码进行封装。

2、便于对该功能进行复用。

3、函数只有被调用才会被执行。

4、函数的出现提高了代码的复用性。

P.S.

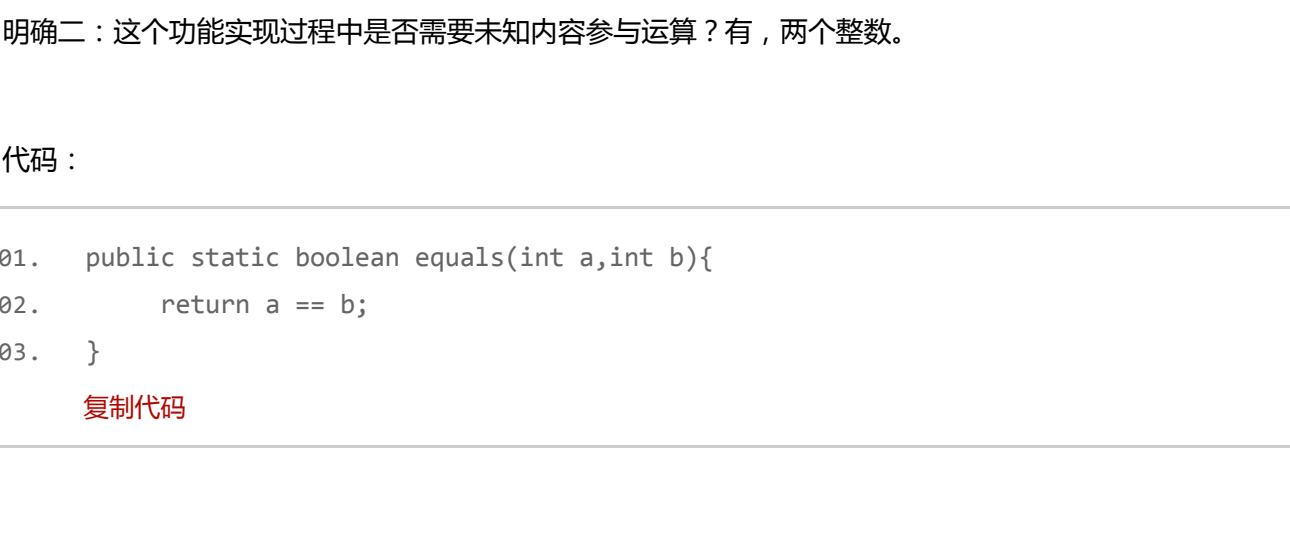
1、对于函数没有具体返回值的情况，返回值类型用关键字void表示，那么该函数中的return语句如果在最后一行可以省略不写，或者写上return。

示例：

```
01. class FunctionDemo{
02.     public static void main(String[] args){
03.         myPrint();
04.         myPrint();
05.     }
06.
07.     public static void myPrint(){
08.         System.out.println("hello world");
09.         return;
10.     }
11. }
```

复制代码

运行结果：



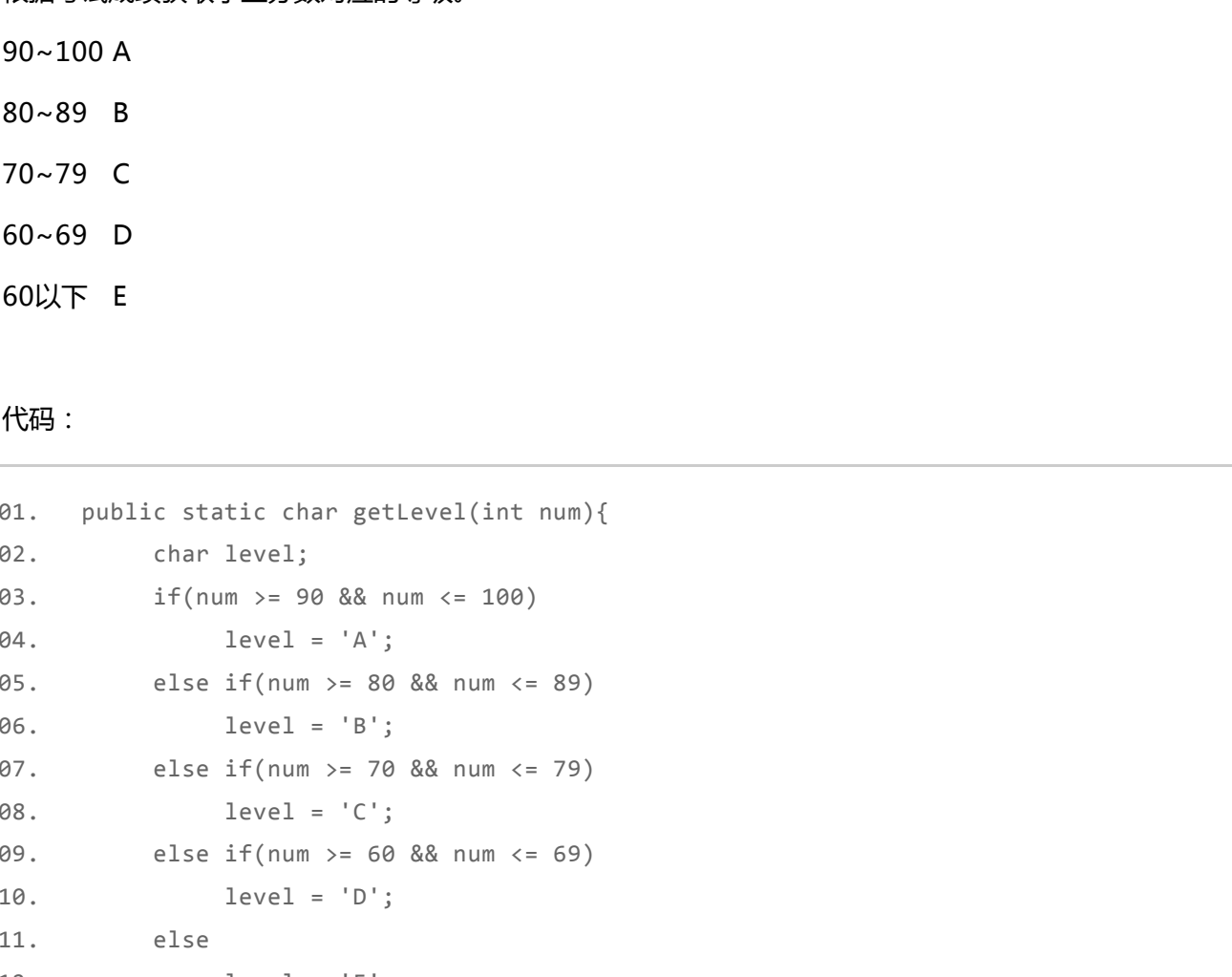
2、函数中只能调用函数，不可以在函数内部定义函数。否则，编译时期就会报错。

示例：

```
01. class FunctionDemo{
02.     public static void main(String[] args){
03.         public static void myPrint() {
04.             System.out.println(" hello world");
05.             return;
06.         }
07.     }
08. }
```

复制代码

运行结果：



3、定义函数时，函数的结果应该返回给调用者，交由调用者处理。

提问：

如果需要写一个两数相加的例子，那么使用下面的代码一好呢？还是代码二好呢？

代码一：

```
01.     public static int add(int a,int b){
02.         return a + b;
03.     }
```

复制代码

代码二：

```
01.     public static void add(int a,int b){
02.         System.out.println(a + b);
03.     }
```

复制代码

答案：虽说两种都可以执行出正确的数据，代码一比较好，因为调用者只需要获取两数相加的结果，而不需要你做打印的操作！

1.8.3 函数的应用

两个明确：

明确要定义的功能最后的结果是什么？

明确在定义该功能的过程中，是否需要未知内容参与运算。

问题1：

需求：定义一个功能，完成两个整数的和的获取。

思路：既然定义功能，就可以用函数来体现。

通过两个明确来完成：

明确一：这个功能的结果是什么？是和。是功能的结果，所以该功能的返回值类型是int。其实，就是在明确函数的返回值类型。

明确二：这个功能实现过程中是否需要未知内容参与运算？有，加数和被加数。这就是函数的参数列表（参数的个数，参数的类型）。其实，就是在明确参数列表。

P.S.

返回值类型和参数类型没有直接关系。

代码：

```
01.     public static int add(int a,int b){
02.         return a + b;
03.     }
```

复制代码

问题2：

需求：定义一个功能，画一个矩形在控制台。

思路：既然定义功能，就可以用函数来体现。

明确一：这个功能结果是什么？没有结果，因为直接打印到了控制台，并未返回给调用者。用void表示。

明确二：这个功能实现过程中是否需要未知内容参与运算？有，行和列不确定，两个，整数int类型。

代码：

```
01.     public static void draw(int row,int col){
02.         for(int x = 1; x <= row; x++){
03.             for(int y = 1; y <= col; y++){
04.                 System.out.print("**");
05.             }
06.             System.out.println();
07.         }
08.         return;//可以省略不写
09.     }
```

复制代码

问题3：

需求：两个数是否相等。

明确一：这个功能的结果是什么？有，boolean类型。

明确二：这个功能实现过程中是否需要未知内容参与运算？有，两个整数。

代码：

```
01.     public static boolean equals(int a,int b){
02.         return a == b;
03.     }
```

复制代码

问题4：

需求：定义一个功能，获取两个整数中较大的那个数。

明确一：这个功能的结果是什么？有，是int类型。

明确二：这个功能实现过程中是否需要未知内容参与运算？有，参与比较的两个整数，int类型。

代码：

```
01.     public static int getMax(int a,int b){
02.         return a > b?a:b;
03.     }
```

复制代码

问题5：

定义功能，打印99乘法表。

代码：

```
01.     public static void print99(){
02.         for(int x = 1; x <= 9; x++){
03.             for(int y = 1; y <= x; y++){
04.                 System.out.println(y + "*" + x + "=" + y*x + "\t");
05.             }
06.         }
07.     }
```

复制代码

问题6：

根据考试成绩获取学生分数对应的等级。

90~100 A

80~89 B

70~79 C

60~69 D

60以下 E

代码：

```
01.     public static char getLevel(int num){
02.         char level;
03.         if(num >= 90 && num <= 100)
04.             level = 'A';
05.         else if(num >= 80 && num <= 89)
06.             level = 'B';
07.         else if(num >= 70 && num <= 79)
08.             level = 'C';
09.         else if(num >= 60 && num <= 69)
10.             level = 'D';
11.         else
12.             level = 'E';
13.         return level;
14.     }
```

复制代码

1.8.4 函数的重载

重载的概念：

在同一个类中，允许存在一个以上的同名函数，只要它们的参数个数或者参数类型不同即可。

重载的好处：

方便于阅读，优化了程序设计。

示例：

```
01. class FunctionDemo{
02.     public static void main(String[] args){
03.         System.out.println(add(3,4));
04.         System.out.println(add(3,0,4,0));
05.         System.out.println(add(3,4,5));
06.     }
07.
08.     //加法运算，两个整数的和
09.     public static int add(int a, int b){
10.         return a + b;
11.     }
12.
13.     //加法运算，两个小数的和
14.     public static double add(double a, double b){
15.         return a + b;
16.     }
17.
18.     //加法运算，三个整数的和
19.     public static int add(int a, int b,int c){
20.         return add(a,b) + c;
21.     }
22. }
```

复制代码

运行结果：



总结：

函数的功能一样，仅仅是参与运算的未知内容不同时，可以定义多函数，却使用统一函数名称，这样方便阅读。在调用时，虚拟机通过参数列表的不同来区分同名函数。

P.S.

1、重载与返回值类型无关，只看参数列表。

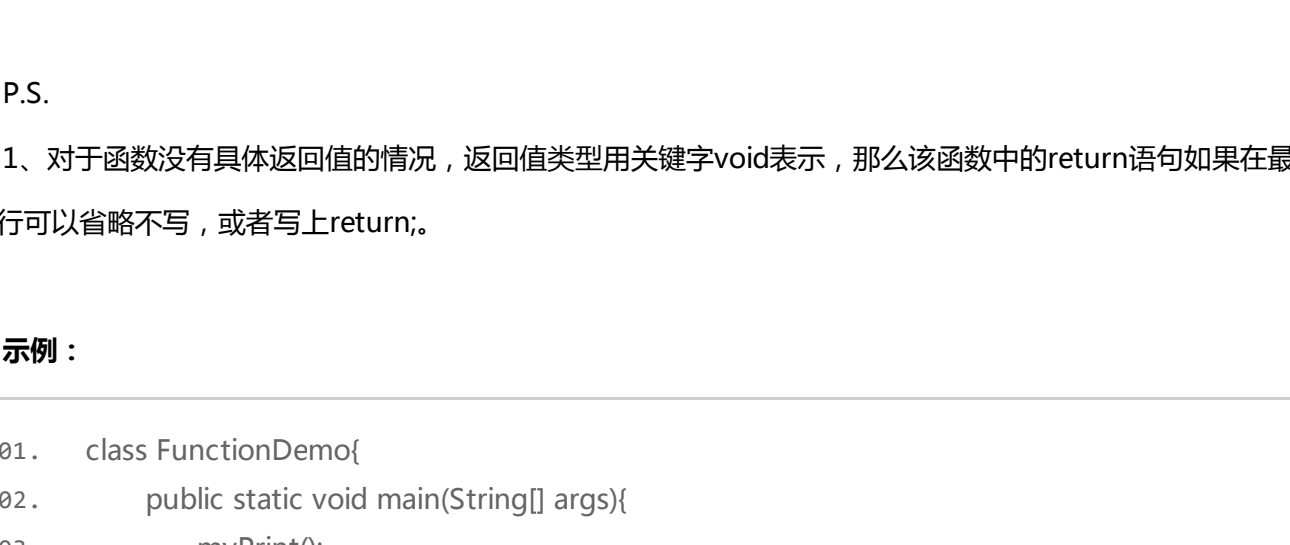
2、java是严谨性语言，如果函数调用出现不确定性，会编译失败。

练习：打印乘法表。

```
01. class FunctionDemo{
02.     public static void main(String[] args){
03.         printCFB(5);
04.         System.out.println("-----");
05.         printCFB();
06.     }
07.
08.     public static void printCFB(int num){
09.         for(int x = 1; x <= num; x++){
10.             for(int y = 1; y <= x; y++){
11.                 System.out.print(y + "*" + x + "=" + y*x + "\t");
12.             }
13.             System.out.println();
14.         }
15.     }
16.
17.     public static void printCFB(){
18.         for(int x = 1; x <= 9; x++){
19.             for(int y = 1; y <= x; y++){
20.                 System.out.print(y + "*" + x + "=" + y*x + "\t");
21.             }
22.             System.out.println();
23.         }
24.     }
25. }
```

复制代码

运行结果：



总结：

函数的功能一样，仅仅是参与运算的未知内容不同时，可以定义多函数，却使用统一函数名称，这样方便阅读。在调用时，虚拟机通过参数列表的不同来区分同名函数。

P.S.

1、重载与返回值类型无关，只看参数列表。

2、java是严谨性语言，如果函数调用出现不确定性，会编译失败。

练习：打印乘法表。

```
01. class FunctionDemo{
02.     public static void main(String[] args){
03.         printCFB(5);
04.         System.out.println("-----");
05.         printCFB();
06.     }
07.
08.     public static void printCFB(int num){
09.         for(int x = 1; x <= num; x++){
10.             for(int y = 1; y <= x; y++){
11.                 System.out.print(y + "*" + x + "=" + y*x + "\t");
12.             }
13.             System.out.println();
14.         }
15.     }
16.
17.     public static void printCFB(){
18.         for(int x = 1; x <= 9; x++){
19.             for(int y = 1; y <= x; y++){
20.                 System.out.print(y + "*" + x + "=" + y*x + "\t");
21.             }
22.             System.out.println();
23.         }
24.     }
25. }
```

复制代码

运行结果：





1.9 数组

1.9.1 数组的定义

概念：

同一种类型数据的集合。其实，数组就是一个容器。

数组的好处：

可以自动给数组中的元素从0开始编号，方便操作这些数据。

格式1：

元素类型[] 数组名 = new 元素类型[元素个数或数组长度];

示例1：需要一个容器，但是不明确容器的具体数据。

```
01. int[] arr = new int[5];
```

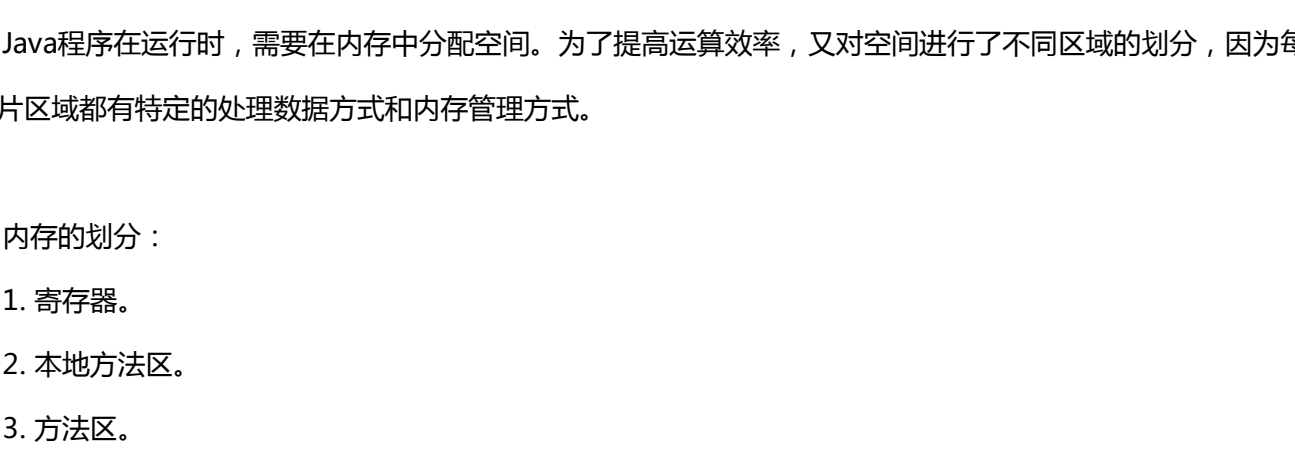
[复制代码](#)

示例2：

```
01. class ArrayDemo{
02.     public static void main(String[] args){
03.         int[] arr = new int[3];
04.         System.out.println(arr[0]);
05.
06.         arr[0] = 89;
07.         System.out.println(arr[0]);
08.     }
09. }
```

[复制代码](#)

运行结果：



格式2：需要一个容器，存储已知的具体数据。

元素类型[] 数组名 = new 元素类型[] (元素，元素，.....);

示例：

```
01. int[] arr = new int[]{3,5,1,7};
```

[复制代码](#)

或

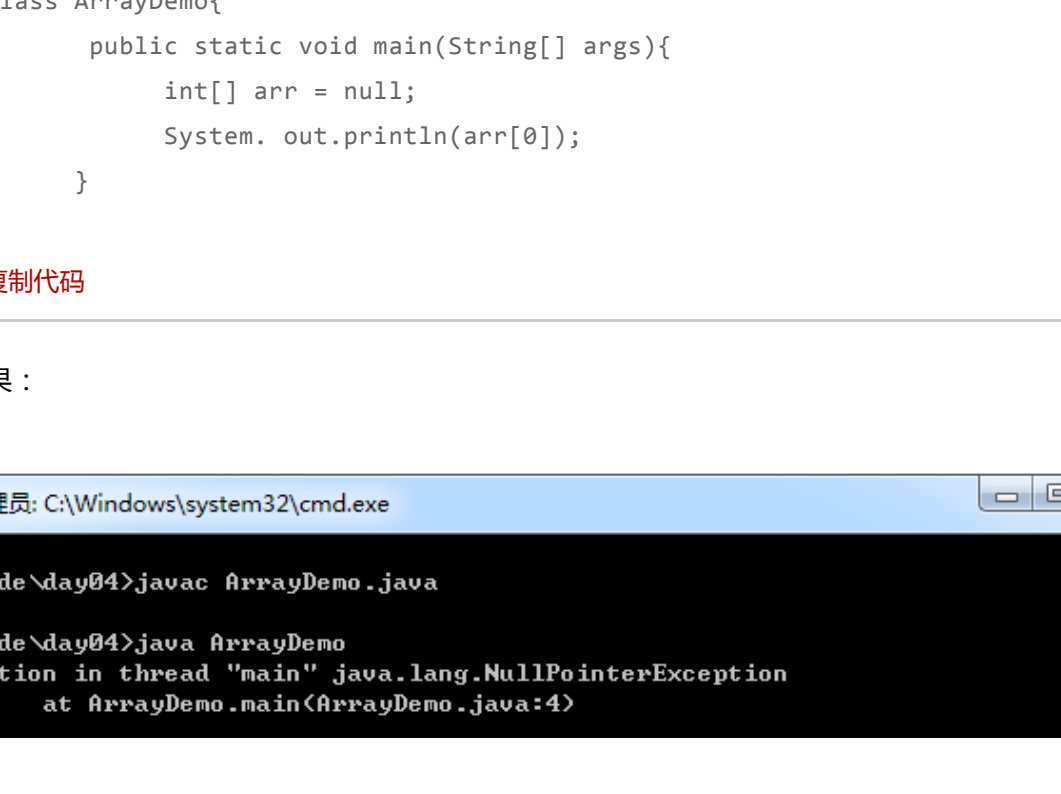
```
01. int[] arr = {3,5,1,7};
```

[复制代码](#)

1.9.2 数组的内存分配及特点

int[] arr = new int[4];

内存分配图：



Java程序在运行时，需要在内存中分配空间。为了提高运算效率，又对空间进行了不同区域的划分，因为每一片区域都有特定的处理数据方式和内存管理方式。

内存的划分：

1. 寄存器。
2. 本地方法区。
3. 方法区。
4. 栈内存。
5. 堆内存。

栈内存：

用于存储局部变量，当变量所属的作用域一旦结束，所占空间会自动释放。

堆内存：

数组和对象，通过new建立的实例都存放在堆内存中。

每一个实体都有内存地址值。

实体中的变量都有默认初始化值，根据类型的不同而不同。整数类型是0，小数类型是0.0或0.0f，boolean类型是false，char类型是'\u0000'。

如果将数组的引用实体设置为null，也就是实体不再被使用，那么会在不确定的时间内被垃圾回收器回收。

1.9.3 数组操作常见问题

数组越界异常(ArrayIndexOutOfBoundsException)：访问到了数组中不存在的索引时发生。

示例1：

```
01. class ArrayDemo{
02.     public static void main(String[] args){
03.         int[] arr = new int[3];
04.         System.out.println(arr[3]);
05.     }
06. }
```

[复制代码](#)

运行结果：



空指针异常(NullPointerException)：当引用型变量没有指向任何实体时，用其操作实体，就会发生该异常。

示例2：

```
01. class ArrayDemo{
02.     public static void main(String[] args){
03.         int[] arr = null;
04.         System.out.println(arr[0]);
05.     }
06. }
```

[复制代码](#)

运行结果：



P.S.

直接打印数组的引用变量，打印出来的结果是数组初始地址的哈希值。

示例3：

```
01. 示例3:
02. ArrayDemo.java
03. class ArrayDemo{
04.     public static void main(String[] args){
05.         int[] arr = new int[3];
06.         System.out.println(arr);
07.     }
08. }
```

[复制代码](#)

运行结果：



P.S.

"[]"表示的是int类型数组，"@@"后面的内容表示数组初始地址的哈希值。

~END~



~爱上海，爱黑马~

