



DeepL

订阅DeepL Pro以翻译大型文件。

欲了解更多信息，请访问www.DeepL.com/pro。

JC2002 Java 程序设计

第 2 天：Java 基础知识（CS）

JC2002 Java 程序设计

第 2 天，第 1 课时：条件结构和循环

参考文献和学习目标

- 在今天的课程中，我们将继续学习 Java 的基础知识
编程语言
- 大部分材料基于 **Java** 的幻灯片：《**如何编程**》第 2、4、5、7 章
，可通过 MyAberdeen 获取
- 今天的理论课程结束后，您应该能够
 - 实施条件结构和循环
 - 定义并初始化数组和 ArrayLists，并使用它们完成简单的任

务，如计算数组中元素的总和

条件语句

- 在许多情况下，程序需要对以下内容进行比较
何去何从
- 在 Java 中，条件操作通常使用 *if ... else* 来实现。
结构

条件布尔值

表达式（例如
比较）或
布尔变量

如果（条件） {

这样做

**}
否则 {**

那么做


}

如果条件是
为真，这样做

如果条件是
错误，请执行

大括号


- 请注意，Java 允许在写 *if* 时省略大括号。
只包含一个语句的语句
- 不过，建议始终使用它们，以避免出现难以察觉的虫子



```
如果 (x>y) {  
    System.out.println("x>y");  
}
```



```
如果  
(x>y)  
    System.out.println("x>y");
```



```
如果  
(x>y);  
    System.out.println("x>y");
```


带 *if* 结构的比较示例

```
1 // 用 if 进行比较的示例
2
3 import java.util.Scanner; // 需要输入 public
4 class ComparisonIf {
5     public static void main(String[] args) {
6         Scanner input = new Scanner(System.in)
7         ;
8         System.out.print("Enter    x:    ");    int    x    =
9         input.nextInt(); System.out.print("Enter y: "); int y
10        = input.nextInt(); if( x == y ) {
11            System.out.printf("%d == %d\n", x, y);
12        }
13        if( x < y ) {
14            System.out.printf("%d < %d\n", x, y);
15        }
16        if( x > y ) {
17            System.out.printf("%d > %d\n", x, y);
18        }
19    } // 主体部分结束
20 } // end class ComparisonIf
```

输入 x: 5

输入 y: 5

5 == 5

输入 x: 0

输入 y: 1

0 < 1

输入 x: 55

输入 y: 10

55 > 10

布尔运算符

- 您可以使用布尔运算符组合条件
!(NOT), && (AND), and || (OR)

```
if (x > a && y > a) { System.out.println("x >
    a and y > a!");
}
if (x > a || y > a) {
    System.out.println ("x > a 或 y > a! ");
}
if (!(x > y)) {
    System.out.println("x <= y!");
}
```

Java *if ... else* 语句

- 大多数 Java 程序员都喜欢编写前面嵌套的
如果..... 否则语句为

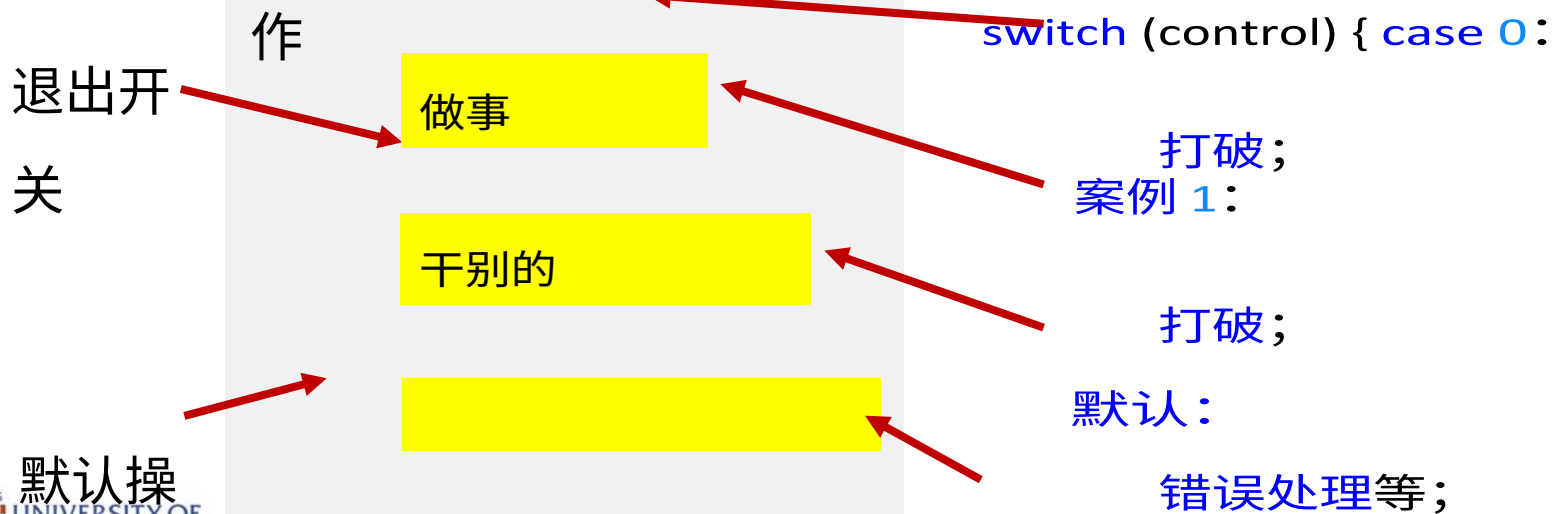
```
if (studentGrade >= 90) {  
    System.out.println("A");  
}  
else if (studentGrade >= 80) {  
    System.out.println("B");  
}  
else if (studentGrade >= 70) {  
    System.out.println("C");  
}  
else if (studentGrade >= 60) {  
    System.out.println("D");  
}  
否则 {
```

注：按照惯例，Java 中的变量名标识符使用首字母小写的驼峰字母命名法（如 **firstNumber**、**studentGrade**）。

```
    System.out.println("F");  
}
```

Java *switch ... case* 语句

- 有时，使用 *switch ... case* 案例结构也是合理的多重比较：



}

变量控制将
在不同条件下进行测试

如果变量值为 0，
将执行以下操作

如果变量值为 1、
这将做到

如果定义的情况都不
符合，则默认情况下
会这样做

与 *switch...case* 结构的比较

```
1 // 使用 case 的条件语句示例 import java.util.Scanner;
2
3 // 需要输入 public class TestCase {
4     public static void main(String[] args) {
5         Scanner input = new Scanner(System.in);
6         System.out.print("Choose 1 or 2: "); int value = input.nextInt();
7         switch(value) {
8             案例 1:
9             System.out.println("You chose 1!");
10            break;
11
12            case 2:
13            System.out.println("You chose 2!");
14            break;
15
16            默认:
17            System.out.println("You did not choose 1 or 2!");
18            break;
19        }
20    } // 主体部分结束
```



```
} // 结束类 TestCase
```

选择 1 或 2: 1
你选择了 1!

选择 1 或 2: 2
你选择了 2!

选择 1 或 2: 3
您没有选择 1 或 2!

运算符的优先性和关联性

操作员	关联性	类型
* / %	从左到右	乘法
+ -	从左到右	添加剂
< <= >	从左到右	关系
== !=	从左到右	平等

=

从右到左

任务

条件运算符 (?:)

- 条件运算符(?:) 是 *if...else* 的简写。
 - 三元运算符 (取三个操作数)
- 操作数和 ?: 构成条件表达式
 - ?左边的操作数是布尔表达式--求值为布尔值 (真或假)
 - 第二个操作数 (在 ? 和 : 之间) 是布尔表达式为真时的值
 - 第三个操作数 (:右侧) 是布尔表达式求值为 false 时的值

```
System.out.println(studentGrade >= 60 ? "Passed" : "Failed");
```

<-- 布尔表达式 -->

<-- 如果为真
-->

<-- 如果为假 --
>

条件运算符示例

```
1 // 条件运算符示例
2
3 import java.util.Scanner; // 需要输入 public
4 class ClassCond {
5     public static void main(String[] args) {
6         Scanner input = new Scanner(System.in);
7         System.out.print("Write your age: ");
8
9         int age = input.nextInt();
10
11         字符串 str = age < 18 ? "minor" : "adult"; System.out.printf("You
        are %s!\n", str);
    } // 主体部分结束
} // end class ClassCond
```

写下你的年龄： 10 你
是未成年人！

写下您的年龄： 50 您
已成年！

当心：它会使代码难以阅读并容易出现错误！

Java 遍历语句 *while*

- 在某些情况下，程序需要多次重复某个操作。次，如果条件仍然成立
- 在 Java 中，*while* iteration 语句可用于此目的

条件：布尔表达式（如比较）或布尔变量

则继续此处

```
while (condition)
{ 执行此操作
}
在此继续
```

如果条件为假，

如果条件为真
，执行此操作
，然后再次测
试条件，如果
仍为真，重复
执行此操作

使用 *while* 迭代语句的示例

```
产品 = 3;  
while (product <= 100) product  
    = 3 * product;
```

- 找出第一个大于 100 的 3 的幂：
- 每次迭代都会将乘积乘以 3，因此乘积会以数值依次为 9、27、81 和 243

• 当乘积变为 243 时，乘积 ≤ 100 变为 false

- 迭代结束，乘积的最终值为 243
- 后的下一条语句继续执行程序。
while 语句

while 循环示例

```
1  // while 循环示例
2  import java.util.Scanner; // 需要输入
3  公共类平均值 {
4      public static void main(String[] args) {
5          Scanner input = new Scanner(System.in);
6          int total = 0; // 初始化成绩总和
7          int gradeCounter = 0; // 初始化年级数 8
8          while (gradeCounter <= 10) { // loop ten times
9              System.out.print ("Enter grade: ");
10             int grade = input.nextInt();
11             total = total + grade;
12             gradeCounter = gradeCounter + 1;
13         }
14     }
```

```
15         System.out.printf("Average: %d%n", total/gradeCounter);
16     } // 主体部分结束
17 } // end classAverage
```

Java *do ... while* 迭代语句

- 迭代语句 *do...while* 类似于 *while* 语句
- 在 *while* 语句中，在执行循环正文**之前**，在循环开始时测试循环终止条件；如果条件为假，则正文永不执行
- *do...while* 语句**在执行循环主体后**测试循环终止条件；因此，主体总是至少执行一次
- 当 *do...while* 循环终止时，将继续执行下一条语句

do...while 循环示例

```
1 // do...while 循环示例 public class
2 DoWhileTest {
3     public static void main(String[] args) {
4         int counter = 1; // 初始化计数器
5
6         做 {
7             System.out.printf("%d ", counter);
8             ++counter;
9         } while (counter <= 10);
10
11 // 主体部分结束
12
13 }
```

1 2 3 4 5 6 7 8 9 10

Java *for* 循环

- *For* 循环在许多编程语言中都很常见
- 在 Java 中，*for* 循环有以下头部组件：

初始化表达式

测试表达式

更新表达式

```
for( int counter = 1; counter <= 10; counter++ )
```

启动循环时，将控制

变量的值初始化为 1

如果

检验表达为真

更新表达式在每次
迭代中更新控制变
量

for 循环示例

```
1 // for 循环示例 公共类
2
3 ForTest {
4     public static void main(String[] args) {
5         for (int counter = 1; counter <= 10; counter++) {
6             System.out.printf("%d ", counter);
7         }
8         System.out.println()
9         ;
10    } // 主体部分结束
11
```

1 2 3 4 5 6 7 8 9 10

有问题或意见？

JC2002 Java 程序设计

第 2 天，第 4 课：数组和数组列表

什么是数组？

- 数组是一组变量（称为元素或组件），包含类型相同的值
- 数组是对象（*引用类型*），而数组的元素可以是*原始类型*或*引用类型*（包括数组）。
 - 记住：布尔型、字节型、字符型、短型、int 型、长型、浮点型和 double 型是基本类型；所有其他类型都是 *引用类型*

声明和创建数组

- 数组对象
 - 您可以在 *数组创建表达式* 中指定元素类型和元素个数，该表达式会返回一个可存储在数组变量中的引用
- 12 数组的声明和数组创建表达式
int elements:

```
int[] c = new int[12];
```

```
int[] c; // 声明数组变量 c = new  
int[12]; // 创建数组
```

- 也可以分以下两个步骤进行：

多维数组

- Java 不直接支持多维数组
 - 为了达到同样的效果，我们可以指定元素也是一维数组的一维数组，等等。
- 二维数组通常用于表示数值表，其数值为按行列排列的数据
 - 有 m 行 n 列的数组称为 m -by- n 数组
- 每个表元素都有两个索引
 - 按照惯例，第一个索引是行，第二个索引是列

	第 0 栏	第 1 栏	第 2 栏
第 0 行	a[0][0]	a[0][1]	a[0][2]
第 1 行	a[1][0]	a[1][1]	a[1][2]

数组名称 行索引 列索引

多维数组可以有两个
以上的维度!

数组示例 1

- 一维数组，将数组元素初始化为默认值为零

```
1  公共类 InitArray1 {  
2      public static void main(String[] args) {  
3          // 声明变量数组并用数组对象进行初始化  
4          int[] array = new int[10]; // 创建数组对象  
5  
6          System.out.printf("%s%8s%n", "Index", "Value"); // 栏目标题  
7  
8          // 输出每个数组元素的值  
9          for (int counter = 0; counter < array.length; counter++) {  
10             System.out.printf("%5d%8d%n", counter, array[counter]);  
11         }  
12     }  
13 }
```

数组示例 2

- 一维数组，用数组初始化器

```
1  公共类 InitArray2 {  
2      public static void main(String[] args) {  
3          // 初始化器列表指定了每个元素的初始值  
4          int[] array = {32, 27, 64, 18, 95, 14, 90, 70, 60, 37};  
5  
6          System.out.printf("%s%s\n", "Index", "Value"); // 栏目标题  
7  
8          // 输出每个数组元素的值  
9          for (int counter = 0; counter < array.length; counter++) {  
10             System.out.printf("%5d%8d\n", counter, array[counter]);  
11         }  
12     }  
13 }
```

数组示例 3

- 的元素之和。
数组

```
1  公共类 SumArray{
2      public static void main(String[] args) {
3          int[] array = {87, 68, 94, 100, 83, 78, 85, 91, 76, 87};
4          int total = 0;
5
6          // 将每个元素的值加到总数中
7          for (int counter = 0; counter < array.length; counter++) {
8              total += array[counter];
9          }
10
11              System.out.printf ("数组元素总数: %d\n", 总数);
12      }
13 }
```

数组示例 4 (1)

- 一维数组、传递数组和单个数组元素到方法

```
1 public class PassArray {  
2     // main 创建数组并调用 modifyArray 和 modifyElement  
3     public static void main(String[] args) {  
4         int[] array = {1, 2, 3, 4, 5};  
5  
6         System.out.printf(  
7             "传递对整个数组的引用的影响: %n" +  
8             "原始数组的值为: %n") ;  
9  
10        // 输出原始数组元素  
11        for (int value : array) {  
12            System.out.printf(" %d", value);  
13        }
```

数组示例 4 (2)

```
14      modifyArray(array) { // 传递数组引用
15          System.out.printf("%n%n修改后的数组值为: %n");
16
17          // 输出修改后的数组元素
18          for (int value : array) {
19              System.out.printf(" %d", value);
20          }
21
22          System.out.printf(
23              "%n%n传递数组元素值的效果: %n" +
24              "array[3] before modifyElement: %d%n", array[3]);
25
26          modifyElement(array[3]); // 尝试修改 array[3]
27          System.out.printf(
28
29
```

```
        "array[3] after modifyElement: %d%n", array[3]);  
    }
```

数组示例 4 (3)

```
30      // 将数组中的每个元素乘以 2
31
32      public static void modifyArray(int[] array2) {
33          for (int counter = 0; counter < array2.length; counter++) {
34              array2[counter] *= 2;
35          }
36      }
37
38      // 将参数乘以 2
39
40      public static void modifyElement(int element) {
41          元素 *= 2;
42          System.out.printf(
43              "modifyElement 中元素的值: %d\n", 元素);
44      }
45  }
```


数组示例 4 (4)

- 程序输出

传递对整个数组的引用的效果：原始数组的值是

1 2 3 4 5

修改后的数组值为

2 4 6 8 10

传递数组元素值的效果：

`modifyElement` 之前的 `array[3]`: 8

`modifyElement` 中元素的值: 16 `array[3]`

after `modifyElement`: 8

数组示例 5 (1)

- 初始化二维数组

```
1  公共类 Init2DArray {  
2      // 创建并输出二维数组  
3      public static void main(String[] args) {  
4          int[][] array1 = {{1, 2, 3}, {4, 5, 6}};  
5          int[][] array2 = {{1, 2}, {3}, {4, 5, 6}};  
6  
7          System.out.println ("数组 1 各行的值为") ;  
8          outputArray(array1); // 按行显示 array1  
9  
10         System.out.printf ("%nValues in array2 by row are%n");  
11         outputArray(array2); // 按行显示 array2  
12     }
```

数组示例 5 (2)

```
13      // 输出二维数组的行和列
14      public static void outputArray(int[][] array) {
15          // 循环数组的行
16          for (int row = 0; row < array.length; row++) {
17              // 循环浏览当前行的列
18              for (int column = 0; column < array[row].length; column++) {
19                  System.out.printf("%d ", array[row][column]);
20              }
21              System.out.println();
22          }
23      }
24  }
```

数组示例 5 (3)

- 程序输出

价值观 于 数组1 由 行数 是

1 2 3

4 5 6

价值观 于 数组2 由 行数 是

1 2

3

4 5 6

类 ArrayList

- 数组在执行时不会改变其大小，以便容纳更多元素
 - 如果需要在执行过程中增加数组大小，则需要重新声明和初始化数组
- 对于具有动态大小的数组，可以使用类包 `java.util` 中的 `ArrayList<T>`
 - 请注意，`T` 是存储在 `ArrayList` 对象中的元素类型的占位符。具有这种占位符、可用于任何类型的类称为 *泛型类*；我们将在本课程的后面部分详细讨论泛型类。

数组列表方法

方法	说明
增加	在 ArrayList 的末尾或特定索引处添加一个元素。
清除	删除 ArrayList 中的所有元素。
载有	如果 ArrayList 包含指定元素，则返回 true，否则返回假的
获取	返回指定索引处的元素。
indexOf	返回指定元素在 ArrayList 中首次出现的索引。
去除	删除第一个出现的指定值或指定索引处的元素。
尺寸	返回 ArrayList 中存储的元素个数。

`trimToSize`

将 ArrayList 的容量减小到当前大小。

数组列表示例 (1)

- 通用 ArrayList<E> 示例

```
1  import java.util.ArrayList;
2
3  公共类 ArrayListCollection {
4      public static void main(String[] args) {
5          // 创建一个新的字符串数组列表, 初始容量为 10
6          ArrayList<String> items = new ArrayList<String>();
7
8          items.add("red"); // 在列表中添加一个项目
9          items.add(0, "yellow"); // 在索引 0 处插入 "yellow" (黄色)
10
11         // 标头
12         System.out.print(
13             "使用计数器控制的循环显示列表内容: " );
```


数组列表示例 (2)

```
14      // 显示列表中的颜色
15      for (int i = 0; i < items.size(); i++) {
16          System.out.printf(" %s", items.get(i));
17      }
18
19
20      // 在显示方法中使用增强的 for 显示颜色
21      display(items、
22          " (nDisplay list contents with enhanced for statement:)");
23
24      items.add("green"); // 在列表末尾添加 "green" items.add("yellow"); // 在
25      列表末尾添加 "yellow" display(items, "List with two new elements:");
26
27      items.remove("yellow"); // 删除第一个 "yellow"。
```

```
display(items, "移除第一个黄色实例：");
```

```
items.remove(1); // 删除索引 1 上的项目
```

```
display(items, "移除第二个列表元素（绿色）：");
```

数组列表示例 (3)

```
32      // check if a value is in the List
33      System.out.printf("/"red\" is %sin the list\\n",
34      items.contains("red") ?"":"not");
35
36
37      // 显示列表中元素的数量
38
39      System.out.printf ("大小: %s\\n", items.size());
40  }
41
42      // 在控制台上显示 ArrayList 的元素
43      public static void display(ArrayList<String> items, String header) {
44          System.out.printf(header); // 显示标题
45
46
47          // 显示项目中的每个元素
48          for (String item : items) {
49              System.out.printf(" %s", item);
50          }
```

```
    }  
    System.out.println();  
}  
}
```

数组列表示例 (4)

- 程序输出

显示带有计数器控制循环的列表内容： 黄 红色 显示带有增强 `for` 语句的列表内容：
黄 红色 带有两个新元素的列表： 黄 红色 绿色 黄色
移除黄色的第一个实例：红色 绿色 黄色 移除第二个列表元素（绿色
）：红色 允许 "红色 "出现在列表中
大小： 2

摘要

- 介绍 Java 程序的基本结构
 - 实现条件语句：if...else、switch...case、三元运算符？
 - 实现循环：while、do.....while、for
 - 定义和使用数组和 ArrayLists

有问题或意见？