Java语言程序设计（第2版）

第 1 章
Java 语言基础知识

清华大学　郑 莉

导学

1. <标题>（静音）
2. <人像>（欢迎）

本章主要内容

- Java与面向对象程序设计简介
- 基本数据类型与表达式
- 数组
- 算法的流程控制

学习建议

- 有C语言基础：可以快速浏览本章内容
- 没有C语言基础：除了学习本章课程内容还要多加练习

Java与面向对象程序设计简介

<1.1>

1. <标题>（静音）
2. <人像>（欢迎）

- 计算机程序设计
  - 对问题进行抽象
  - 用计算机语言表述，利用机器求解

- 程序设计语言发展的历程
  - 机器语言
  - 汇编语言
  - 高级语言
  - 面向对象的语言

- 面向对象的思想
  - 将客观事物看作具有状态和行为的对象，通过抽象找出同一类对象的共同状态和行为，构成类。

- 面向对象技术给软件发展带来的益处
  - 可重用性
  - 可靠性

- 面向对象语言的基本特征
  - 抽象和封装
  - 继承
  - 多态

Java 的发展历程

Java 的发展历程

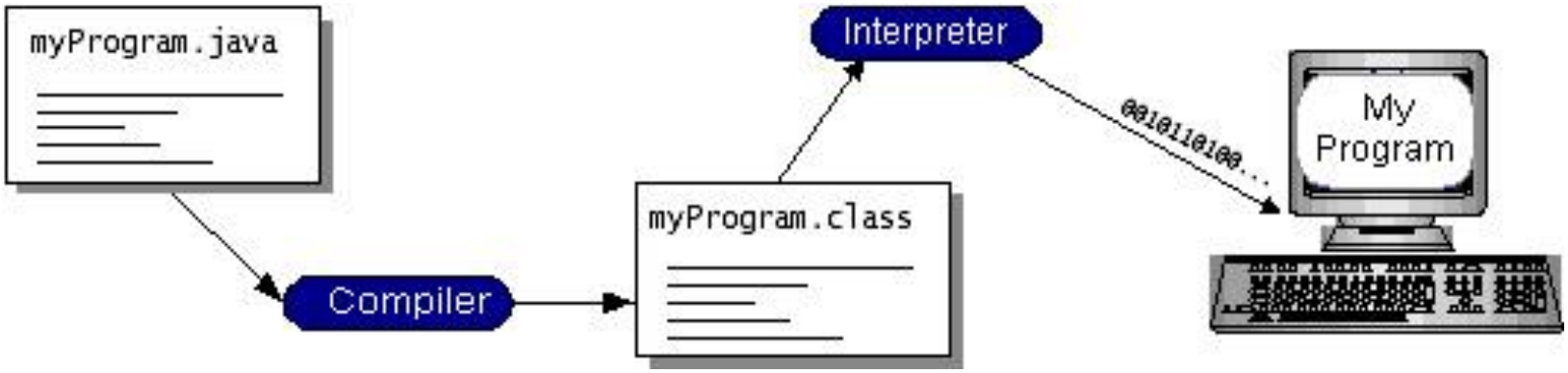
- 1995年年初，Sun公司发布了Java语言。
- 1996年，Sun公司发布了JDK1.0。
- 1998年，Sun公司发布了Java历史上的一个重要版本：JDK 1.2。该版本将Java分成了J2SE、J2EE和J2ME三个版本。
- 2004年，Sun公司发布了万众期待的JDK 1.5，同时将JDK 1.5改名为Java SE5.0，J2EE改名为Java EE，J2ME改名为Java ME。
- 2006年，Sun公司发布了JDK 1.6，也就是Java SE 6.0。
- 2009年4月，Oracle公司收购了Sun公司。
- Google公司在2007年推出的基于Linux的开源移动操作系统Android极大地推动了Java语言的发展。
- 2011年，Oracle公司发布了Java SE 7，这是Oracle发布的第一个Java版本。
- 2014年3月19日Oracle发布JDK8。

Java 的发展历程

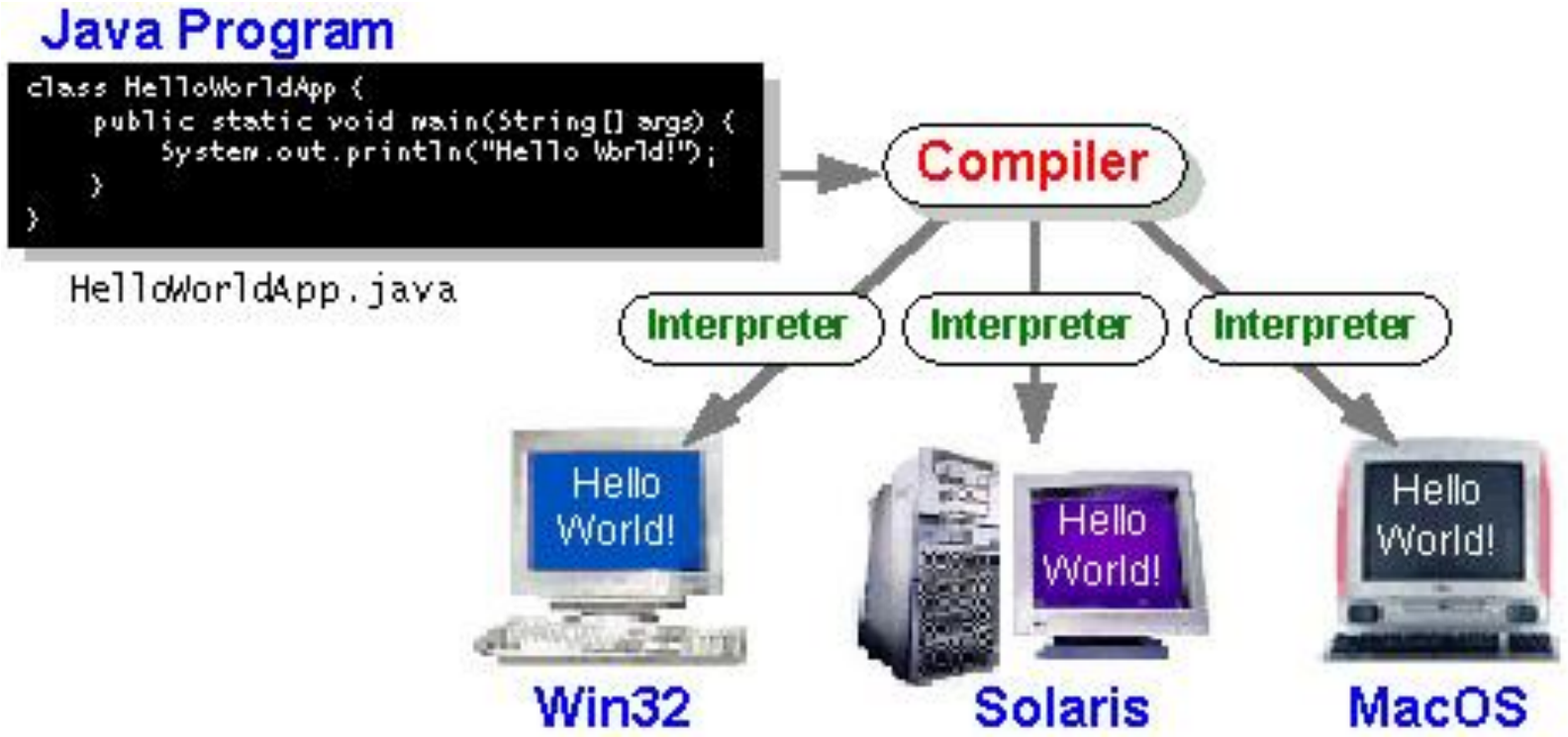
- 2006年，Sun公司发布了JDK 1.6，也就是Java SE 6.0。
- 2009年4月，Oracle公司收购了Sun公司。
- Google公司在2007年推出的基于Linux的开源移动操作系统Android极大地推动了Java语言的发展。
- 2011年，Oracle公司发布了Java SE 7，这是Oracle发布的第一个Java版本。
- 2014年3月19日Oracle发布JDK8。

Java程序的跨平台特性

- Java程序编译执行的过程



- 一次编写，各处运行


Java Program
class HelloWorldApp {
public static void main(String[] args) {
System.out.println("Hello World!");
}
}
HelloWorldApp.java
Compiler
Interpreter
Interpreter
Interpreter
Hello World!
Hello World!
Hello World!
Win32
Solaris
MacOS

结束语

- 这一节对面向对象的程序设计思想和Java语言做了一个概要介绍。下一节我们将学习具体的Java语法。

基本数据类型与表达式

<1.2>

1. <标题>（静音）
2. <人像>

- 变量与常量
- 基本数据类型
- 表达式与运算符
- 类型转换

文字量

- 文字量直接出现在程序中并被编译器直接使用，比如30、3.141592654。文字量也称为文字常量，所谓常量，就是在其生存期内值不可改变的量。

- 标识符
  - 标识符是一个名称，与内存中的某个位置（地址）相对应
  - 标识符的第一个字符必须是下列字符之一：
    - **大写字母 (A-Z)**
    - **小写字母 (a-z)**
    - 下划线(_)
    - 美元符号 ($)
  - 标识符的第二个字符及后继字符必须是：
    - 上述列表中的任意字符
    - 数字字符 (0-9)

- 变量
  - 一个由标识符命名的项；
  - 每个变量都有类型；
  - 变量的值可以被改变。
- 常量
  - 常量一旦被初始化以后就不可改变。

基本数据类型

数值型

类型	说明	长度	最小值	最大值
byte	带符号微整数	8位	-128	127
short	带符号短整数	16位	-2^{15}	$2^{15}-1$
int	带符号整数	32位	-2^{31}	$2^{31}-1$
long	带符号长整数	64位	-2^{63}	$2^{63}-1$
float	单精度浮点数	32位	-2^{-149}	$(2-2^{-23})\cdot 2^{127}$
double	双精度浮点数	64位	2^{-1074}	$(2-2^{-52})\cdot 2^{1023}$

数值型文字量

数据类型	文字量
byte,short,int	十进制数，开头不为0；0X跟十六进制数，如0XF1C4；0跟八进制数，如0726
long	同上，但后面跟l或L，如：84l，0X1F39L
float	数字后跟f或F，如1.23456F，1.893E2F
double	后面可选d或D做后缀，如：1.23D
boolean	true或false

字符型

- 字符类型的文字量是单引号括起来的字符或者转义序列，如：‘Z’，‘k’，‘\t’。
- 用16位的Unicode字符作为编码方式，如下面代码所示：
- char var_char = ‘a’;
- char char_tab = ‘\t’;

- 某些特殊的字符型常量需要使用转义的形式来表示

转义字符	表示含义
\’	单引号字符
\”	双引号字符
\\	反斜杠字符
\r	回车
\n	回车并换行
\t	水平制表符
\b	退格

布尔类型

- 布尔类型表示一个逻辑量， 有两个取值： true和false
- 例如:

```
boolean is_salaried;
  boolean is_hourly;
  is_salaried = true; //将 is_salaried设置为true
  is_hourly = false; //将is_hourly设置为false
```

字符串

- String 是一个类
- String类JDK标准类集合中的一部分

```
String animal = "walrus";
```

- 字符串文字量
  - 由零个或多个字符组成，以双引号括起
  - 每一个字符都可以用转义序列来表示
  - 例如：

```
""                          // 空字符串
"\""                        // 只包含 " 的字符串
"This is a string"          // 有16个字符的字符串
"This is a " + "string"
    //字符串常量表达式,由两个字符串常量组成
```

运算符与表达式

- 算术运算符
  - 运算符 ++ 和 --
    例如： i++; --j;
  - 一元运算符 + 和 –
  - 加法运算符 + 和 -
  - 乘法运算符 *, /, 和 %

- 赋值运算符
  - 简单赋值运算符 =
  - 复合赋值运算符

    *= /= %= += -= <<= >>= >>>=

    &= ^= |=

    E1 op= E2 等效于 *E1* = (*T*)((*E1*) *op* (*E2*)), 其中*T* 是 *E1* 的类型

- 关系运算符
  - 关系表达式的类型永远是布尔类型（bool）.
  - 算术比较运算符 <, <=, >, and >=
  - 类型比较运算符 instanceof
    - 例如： e instanceof Point //Point 是一个类
  - 相等关系运算符
    - 数字相等运算符 == ， !=
    - 布尔相等运算符 == ， !=
    - 引用相等运算符 == ， !=

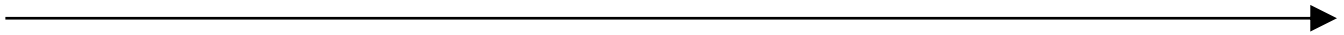
- 逻辑运算符
  - “与”运算 &&
    - 如果两个操作数的值都为true运算结果为true; 否则，结果为false.
  - “或”运算 ||
    - 如果两个操作数的值都为false运算结果为false;否则，结果true
  - “非”运算符！
    - 操作数的类型必须是布尔类型
    - 如果操作数的结果为 false，则表达式的结果为 true，如果操作数的结果为 true则表达式的结果为 false

- 条件运算符 (表达式1？表达式2：表达式3)
  - 首先计算表达式1
  - 如果表达式1的值为 true, 则选择表达式2的值
  - 如果表达式1的值为 false,则选择表达式3的值

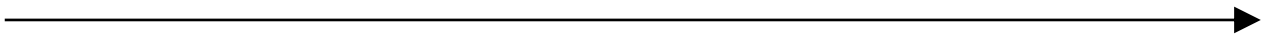
类型转换

- 每个表达式都有类型
- 如果表达式的类型对于上下文不合适
  - 有时可能会导致编译错误
  - 有时语言会进行隐含类型转换

- 扩展转换
  - byte, char, short, int, long, float, double



  - 从一种整数类型到另一种整数类型，或者从float到double的转换不损失任何信息
  - 从整数类形向float或double转换，会损失精度
- 窄化转换
  - double, float, long, int, short, byte,char



  - 窄化转换可能会丢失信息

- 隐含转换
  - 赋值转换
    - 将表达式类型转换为制定变量的类型
  - 方法调用转换
    - 适用于方法或构造方法调用中的每一个参数
  - 字符串转换
    - 任何类型（包括null类型）都可以转换为字符串类型
    - 只当一个操作数是String类型时，适用于+运算符的操作数
- 显式转换（强制转换）
  - 将一个表达式转换为指定的类型
  - 例如 (float)5.0

结束语

- 这一节介绍了Java的数据类型和表达式，其中大部分与C语言的语法是类似的，也有一些不同点他家要特别关注。下一节我们将学习数组，Java的数组与C语言还是有很大不同的。

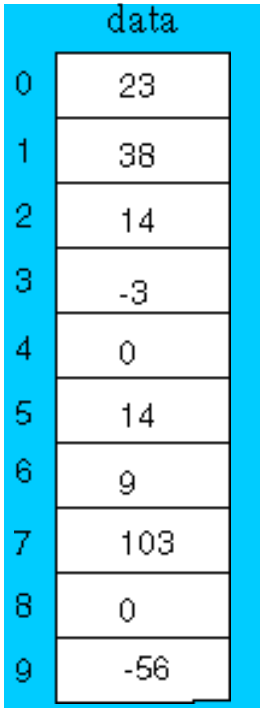
数组

<1.3>

1. <标题>（静音）
2. <人像>（欢迎）

数组的概念

- 数组由同一类型的对象或者基本数据组成，并封装在同一个标识符（数组名称）下。



- 数组是对象
  - 动态初始化
  - 可以赋值给Object类型的变量
  - 在数组中可以调用类Object 的所有方法
  - 每个数组都有一个由 public final 修饰的成员变量：*length*，即数组含有元素的个数（*length*可以是正数或零）

- 数组元素
  - 数组中的变量被称作数组的元素
  - 元素没有名字，通过数组名字和非负整数下标值引用数组元素。

数组的创建和使用

- 数组的声明
- 数组的创建
- 数组元素的初始化
- 使用数组

数组引用的声明

数组引用的声明

- 声明数组时无需指明数组元素的个数，也不为数组元素分配内存空间
- 不能直接使用，必须经过初始化分配内存后才能使用

数组声明举例

Type[ ] arrayName;

例如：

int[] intArray;

String[] stringArray;

Type arrayName[ ];

例如：

int intArray[];

String stringArray[];

数组的创建

数组的创建

- 用关键字new构成数组的创建表达式，可以指定数组的类型和数组元素的个数。元素个数可以是常量也可以是变量
- 基本类型数组的每个元素都是一个基本类型的变量；引用类型数组的每个元素都是对象的的引用

数组的创建举例

arryName=new Type[componets number];

- 例如：

  int[] ai;　ai=new int[10];

  String[] s;　s=new String[3];

- 或者可以将数组的声明和创建一并执行

  int ai[]=new int[10];

- 可以在一条声明语句中创建多个数组

  String[] s1=new String[3], s2=new String[8];

数组元素的初始化

数组元素的初始化

- 声明数组名时，给出了数组的初始值，程序便会利用数组初始值创建数组并对它的各个元素进行初始化

```
int a[]={22, 33, 44, 55};
```

- 创建数组的时，如果没有指定初始值，数组便被赋予默认值初始值。
  - 基本类型数值数据，默认的初始值为0；
  - boolean类型数据，默认值为false;
  - 引用类型元素的默认值为null。
- 程序也可以在数组被构造之后改变数组元素值

使用数组

使用数组

- 引用数组的一个元素：
  arrayName[index]
  - 数组下标必须是 int , short, byte, 或者 char。
  - 下标从零开始计数。
- 元素的个数即为数组的长度，可以通过 arryName.length 得到。
- 元素下标最大值为 length – 1，如果超过最大值，将会产生数组越界异常（ArrayIndexOutOfBoundsException）

- 数组名是一个引用：

```
public class Arrays
{   public static void main(String[] args)
    {  int[] a1 = { 1, 2, 3, 4, 5 };
       int[] a2;
       a2 = a1;
       for(int i = 0; i < a2.length; i++)   a2[i]++;
       for(int i = 0; i < a1.length; i++)
          System.out.println( "a1[" + i + "] = " + a1[i]);
    }
}
```

运行结果：

```
a1[0] = 2
a1[1] = 3
a1[2] = 4
a1[3] = 5
a1[4] = 6
```

- 字符串引用构成的数组——每个元素都是引用

```
public class ArrayOfStringsDemo
{ public static void main(String[] args)
   { String[] anArray =
        { "String One", "String Two", "String Three"};
     for (int i = 0; i < anArray.length; i++)
     { System.out.println(anArray[i].toLowerCase());
     }
   }
}
```

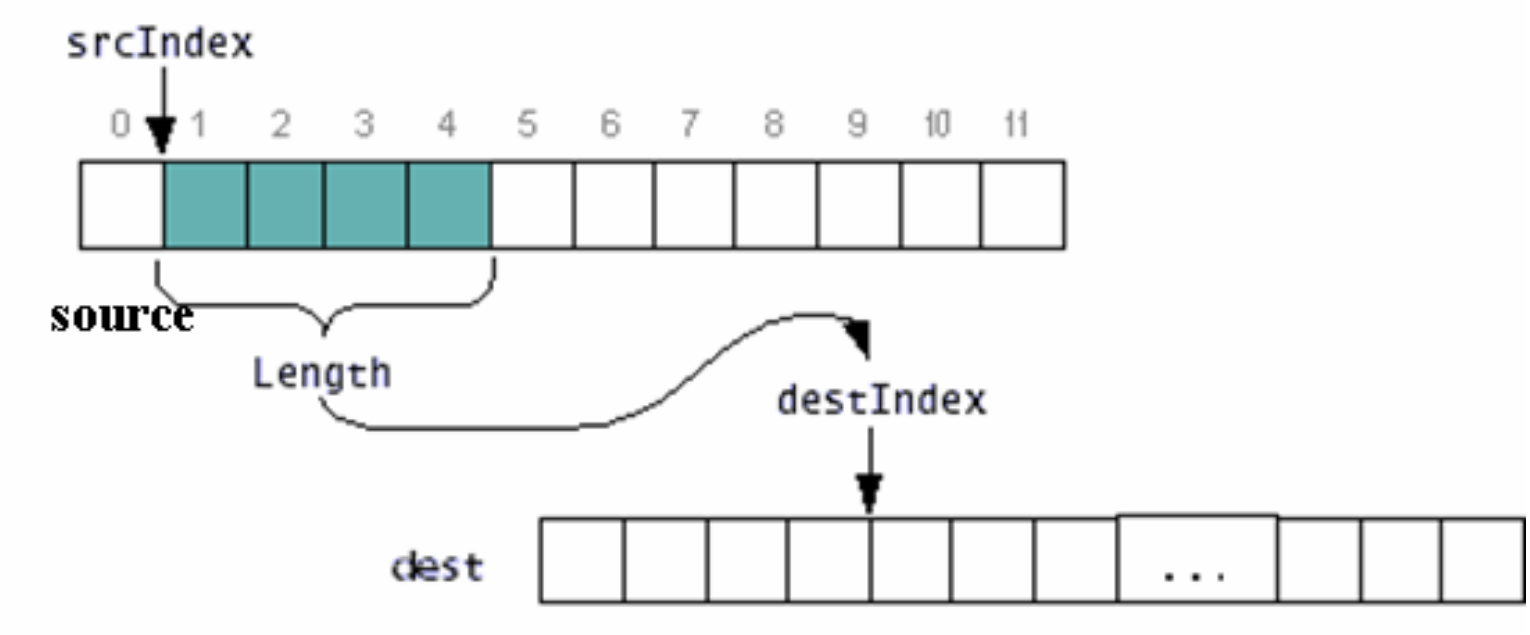
运行结果：

```
string one
string two
string three
```

如何复制一个数组?

- 复制数组或数组的部分元素

public static void arraycopy(Object *source* , int *srcIndex* , Object *dest* , int *destIndex* , int *length*)



- 复制数组或数组的部分元素

```
public class ArrayCopyDemo
{ public static void main(String[] args)
  { char[] copyFrom = { 'd', 'e', 'c', 'a', 'f', 'f', 'e',
                        'i', 'n', 'a', 't', 'e', 'd'};
    char[] copyTo = new char[7];
    System.arraycopy(copyFrom, 2, copyTo, 0, 7);
    System.out.println(new String(copyTo));
  }
}
```

多维数组

多维数组

int[][] gradeTable;

……

gradeTable[0][1] 为42
gradeTable[3][4] 为93
gradeTable[6][2] 为78

Student	Week				
	0	1	2	3	4
0	99	42	74	83	100
1	90	91	72	88	95
2	88	61	74	89	96
3	61	89	82	98	93
4	93	73	75	78	99
5	50	65	92	87	94
6	43	98	78	56	99

- 二维数组的声明和构造——多种方式
  - int[ ][ ] myArray ;
    - myArray 是一个指向2维整数数组的引用。其初始值为null。
  - int[ ][ ] myArray = new int[3][5] ;
    - 定义引用时，同时构造数组并初始化引用。这个数组所有元素的初始值为零。
  - int[ ][ ] myArray = { {8,1,2,2,9}, {1,9,4,0,3}, {0,3,0,0,7} };
    - 定义引用、构造数组，初始化数组元素。

二维数组的长度及每行的长度

```
class UnevenExample3
{
   public static void main( String[] arg )
   {
   // 声明并构造一个2维数组
   int[][] uneven =
                { { 1, 9, 4 },
                  { 0, 2},
                    { 0, 1, 2, 3, 4 } };
   //数组的长度 (行数)
   System.out.println("Length of array is: " + uneven.length );
   // 数组每一行的长度（列数）
   System.out.println("Length of row[0] is: " + uneven[0].length );
   System.out.println("Length of row[1] is: " + uneven[1].length );
   System.out.println("Length of row[2] is: " + uneven[2].length );
   }
}
```

运行结果：

```
Length of array is: 3
Length of row[0] is: 3
Length of row[1] is: 2
Length of row[2] is: 5
```

结束语

- 这一节介绍了Java的数组。从抽象数据结构的角度，高级语言中的数组结构都是类似的，但是语法上，Java的数组与C语言还是有很大不同的，大家要特别注意。
- 下一节我们将学习算法的流程控制，这与C语言的流程控制语句是基本相同的。

算法的流程控制

<1.4>

1. <标题>（静音）
2. <人像>（欢迎）

流程控制语句

- if语句
- switch语句
- for语句
- while语句
- do while语句

if语句

- 只有if分支，没有else分支

```
if (boolean-expression) {
   // statement1;
 }
```

- if-else语句

```
if (boolean-expression) {
    // statement1 ;
}
else {
// statement2 ;
}
```

- if-else语句的特殊形式

```
if (boolean expression) {
    //statement1;
}
else if (boolean expression) {
    //statement2;
}
else if (boolean expression){
    //statement;
}
...
else {
    //statement;
}
```

以条件运算符代替简单的if_else

- if-else

```
If (a>b)
 System.out.println("The larger one is: "+a);
else
 System.out.println("The larger one is: "+b);
```

- 用条件运算符重写:

```
System.out.println("The larger one is: " + (a>b)?a:b);
```

switch语句

- 语法形式

```
switch (switch-expression) {
    case value1:   statements for case1; break;
    case value2:   statements for case2; break;
    ...
    case valueN:   statements for caseN; break;
    default:       statements for default case; break;
}
```

- 注意问题
  - switch-expression、常量值value1到valueN必须是整形或字符型
  - 如果表达式的值和某个case后面的值相同，则从该case之后开始执行，直到break语句为止
  - default是可有可无的，若没有一个常量与表达式的值相同，则从default之后开始执行

for语句

- 是Java三个循环语句中功能较强、使用较广泛的一个
- for循环可以嵌套
- 语法格式如下

```
for (start-expression; check-expression; update-expression) {
    //body of the loop;
}
```

  - start-expression完成循环变量和其他变量的初始化工作
  - check-expression是返回布尔值的条件表达式，用于判断循环是否继续
  - update-expression用来修整循环变量，改变循环条件
  - 三个表达式之间用分号隔开

- 增强for循环
  - 用来对数组或者集合对象进行遍历
  - 语法格式：

```
for (Type name : 数组或集合类型对象) {
//循环体;
}
```

- while语句
  - 语法形式：

```
while (check-expression) {
    //body of the loop;
}
```

  - 解释
    - 条件表达式(check-expression)的返回值为布尔型
    - 循环体可以是单个语句，也可以是复合语句块
  - 执行过程
    - 先判断check-expression的值，为真则执行循环体
    - 循环体执行完后再无条件转向条件表达式做计算与判断；当计算出条件表达式的值为假时，跳过循环体执行while语句后面的语句。若为真，则继续执行循环

- do-while语句
  - 语法形式：

```
do {
   //body of the loop;
} while (check-expression);
```

  - 与while语句很类似，不同的是它首先无条件的执行一遍循环体，再来判断条件表达式的值，若表达式的值为真，则再运行循环体，否则跳出do-while循环，执行下面的语句
  - 特点：它的循环体至少要执行一次

break语 句

- 功能
  - 跳出循环，不再执行剩余部分
- 适用场合
  - 在switch 结构中，用来终止switch语句的执行
  - 在for循环及while循环结构中，用于终止break语句所在的最内层循环；与标号一同使用时，将跳出标号所标识的循环
  - 也可用在代码块中，用于跳出它所指定的块

- continue语句
  - 必须用于循环结构中
  - 停止本次迭代，回到循环起始处，开始下一次迭代过程
  - 有两种使用格式
    - 不带标号的continue语句
      - 终止当前这一轮的循环，跳出本轮循环剩余的语句，直接进入当前循环的下一轮
    - 带标号的continue语句
      - 使程序的流程直接转入标号标明的循环层次

结束语

- 这一节介绍了在Java的方法中如何控制算法的流程。Java的流程控制语句与C语言的流程控制语句是基本相同的。

第1章小结

1. <标题>（静音）
2. <人像>（欢迎）

小结

- 本章内容
  - Java与面向对象程序设计简介
  - 基本数据类型与表达式
  - 数组
  - 流程控制
- 复习要求
  - 下载、安装J2se
  - 熟悉命令行方式编译、运行Java程序
  - 熟悉一种集成开发环境