内容提要

- Java程序的构成
- 基本数据类型及转换
- 运算符与表达式
- 流程控制
- 数组的使用

Java程序的构成

- 逻辑构成

Java源程序逻辑构成分为两大部分：程序头包的引用和类的定义。

■ 程序头包的引用

主要是指引用JDK软件包自带的包，也可以是自己定义的类。引用之后程序体中就可以自由应用包中的类的方法和属性等。

Java程序的构成

■ 类的定义

Java源程序中可以有多个类的定义，但必须有一个主类，这个主类是Java程序运行的入口点。在应用程序中，主类为包含main方法的类；在Applet中，主类为用户自定义的系统Applet类的扩展类。在Java源程序中，主类的名字同文件名一致。

类的定义又包括类头声明和类体定义。类体中包括属性声明和方法描述。

【例】下面是一个应用程序，也是一个Applet，既可以在命令行下运行，也可以嵌入到HTML网页中用appletviewer命令运行。运行时在界面上的第一个文本框中输入你的名字，按回车键后，在第二个文本框中会显示“XXX，欢迎你来到Java世界！”。

```
//程序文件名称为WelcomeApplet.java          注释语句

import java.applet.*;
import java.awt.*;
import java.awt.event.*;                    引入包

public class WelcomeApplet
                extends Applet implements ActionListener{
```

主类类头

```
Label lblName;
TextField txtName;
TextField txtDisp;
```

属 性

```
public void init()
{
    lblName = new Label("请输入您的名字");
    txtName = new TextField(8);
    txtDisp = new TextField(20);
    add(lblName);
    add(txtName);
    add(txtDisp);
    txtName.addActionListener(this);
}
```

init方法

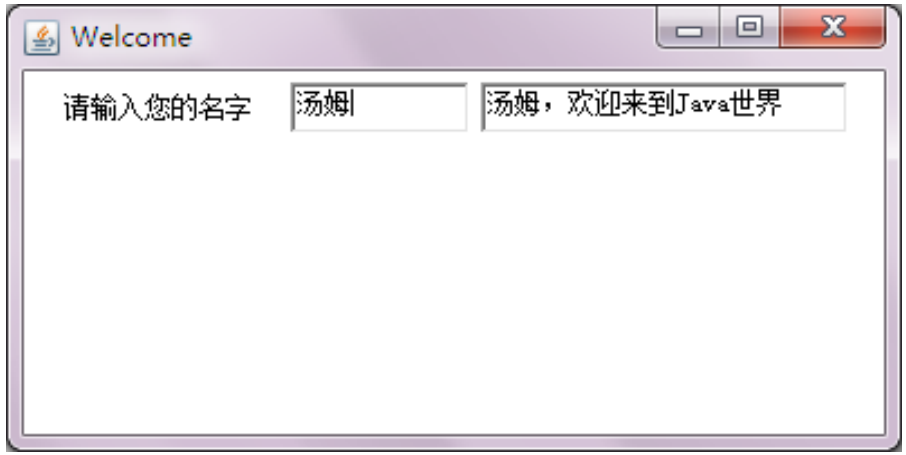
```
    public void actionPerformed(ActionEvent e){
        txtDisp.setText(txtName.getText()+"欢迎来到Java世界");
    }
```

actionPerformed方法

```
    public static void main(String[] args) {
        Frame f=new Frame("Welcome");
        f.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent evt){
                System.exit(0);
            }
        });

        WelcomeApplet a=new WelcomeApplet();
        a.init();
        f.add("Center",a);
        f.setSize(400, 300);
        f.setVisible(true);
        // a.start();
    }
}
```

main方法



Java程序的构成

● 物理构成

Java源程序物理上由三部分构成，分别为语句、块和空白。

(1) 语句指一行以分号“;”结束的语句。

(2) 块：用括号对{}界定的语句序列，可以嵌套使用。

(3)空白：语句之间、块内部或者块之间的空白行。空白不影响Java源程序的编译和运行，适当地运用空白，可以形成良好的代码风格。

Java程序的构成

● 注释语句

注释语句主要用来进行一些说明，或者标记一些无用的程序语句。有两种注释方法：

(1) 行注释为以//开始的行

(2) 块注释以/*开始和*/结束

Java编译器忽略注释后的程序语句或说明。

```
//程序文件名称为WelcomeApplet.java
/*程序文件名称为WelcomeApplet.java*/
/*
程序文件名称为
WelcomeApplet.java
*/
```

Java程序的构成

● 标识符、关键字和转义符

Java语言中，标识符是赋予变量、类和方法等的名称。标识符由编程者自己指定，但需要遵循一定的语法规范：

(1) 标识符由字母(A~Z, a~z)、数字(0~9)、下划线(_)、美元符号($)组成，但美元符号用得较少。

(2) 标识符从一个字母、下划线或美元符号开始。

(3) 标识符大小写敏感，必须区别对待。

(4) 标识符没有最大长度的限制，但最好表达特定的意思。

(5) 标识符定义不能是关键字。

Java程序的构成

● 标识符、关键字和转义符

关键字又称保留字，是指Java语言中自带的用于标志数据类型名或者程序构造名等的标识符，如public、int等。

转义符是指一些有特殊含义的、很难用一般方式表达的字符，如回车、换行等。所有的转义符以反斜线(\)开头，后面跟着一个字符来表示某个特定的转义符。

引用方法	含义
\b	退格
\t	水平制表符Tab
\n	换行
\f	换页符
\r	回车
\'	单引号
\"	双引号
\\	反斜线

基本数据类型与表达式

- 变量与常量
- 基本数据类型
- 表达式与运算符
- 类型转换

变量与常量

- 常量
  - 常量是指整个运行过程中不再发生变化的量，例如数学中的π= 3.1415……，在程序中需要设置成常量。
  - 常量一旦被初始化以后就不可改变。
- 变量
  - 变量是指程序的运行过程中可以发生变化的量，通常用来存储中间结果。
  - 声明（创建）变量时，系统根据变量的数据类型在内存中开辟相应的存储空间并赋予初始值。
  - 变量还有作用域，超出它声明语句所在的块就无效。

基本数据类型

Java编程语言定义了八种基本的数据类型，共分为四类：整数类(byte、short、int、long)、文本类(char)、浮点类(double、float)和逻辑类(boolean)。

分 类	数据类型	关键字	占字节数(位)	缺省数值	取 值 范 围
整数类	字节型	**byte**	8	0	-2^{7}～$2^{7}-1$
	短整型	**short**	16	0	-2^{15}～$2^{15}-1$
	整型	**int**	32	0	-2^{31}～$2^{31}-1$
	长整型	**long**	64	0	-2^{63}～$2^{63}-1$
文本类	字符型	**char**	16	'\u 0000'	'\u 0000' ～ '\u FFFF'
浮点类	浮点型	**float**	32	0.0F	—
	双精度型	**double**	64	0.0D	—
逻辑类	逻辑型	**boolean**	8	false	true、false

基本数据类型

- 文字量

  直接出现在程序中并被编译器直接使用的值。

- 整数类(byte、short、int、long)及其文字量

  (1) 采用三种进制——十进制、八进制和十六进制

  2 —— 十进制值是2;

  077 —— 首位的0表示这是一个八进制的数值;

  0xBAAC —— 首位的0x表示这是一个十六进制的数值。

  (2) 具有缺省int。

  (3) 用字母“L”和“l”定义为long。

  (4) 所有Java编程语言中的整数类型都是带符号的数字。

基本数据类型

- 浮点类(float、double)及其文字量
  - 默认为double类型，如果一个数字包括小数点或指数部分，或者在数字后带有字母F或f(float)、D或d(double)，则该数字为浮点数。
  - 一个浮点文字量包括以下几个部分
    - 整数部分、小数点、小数部分
    - 指数 (e or E)、类型后缀 (f or F for float, d or D for double)
  - float 类型文字量举例：

    1e 1f 2.f .3f 0f 3.14f 6.02e23f 6.02e-23f
  - double 类型文字量举例：

    1e1 2. .3 0.0 3.14 1e-9d 1e137 1e-137

基本数据类型

- 逻辑类(boolean)及其文字量
  - boolean类型只有两种值，由文字量true和false表示。
  - 例如： boolean flag = true;

基本数据类型

- 文本类(char)及其文字量

(1) 代表一个16 bits Unicode字符。

(2) 必须包含用单引号(' ')引用的文字。

(3) 使用下列符号:

'a'—— 一个字符。

'\t'—— 一个制表符。

'\u???? '——一个特殊的Unicode字符，????应严格使用四个十六进制数进行替换。例如：\u002A——星号*, \u002B——加号+

基本数据类型

- 字符串（String）及其文字量
  - String 是一个类，String类JDK标准类集合中的一部分

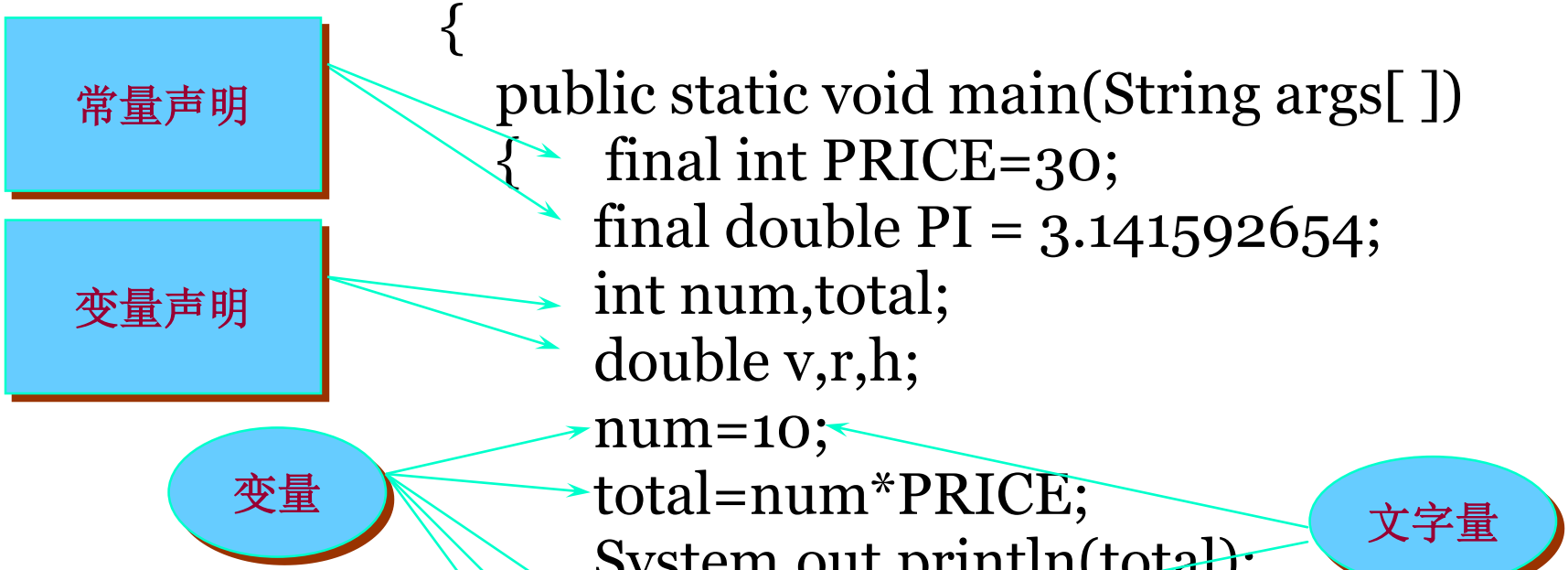
    例如： String animal = "walrus";
  - 字符串文字量

    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
```

基本数据类型




```
public class ex1_1
{
   public static void main(String args[ ])
   {    final int PRICE=30;
        final double PI = 3.141592654;
        int num,total;
        double v,r,h;
        num=10;
        total=num*PRICE;
        System.out.println(total);
        r=2.5;
        h=3.2;
        v=PI*r*r*h;
        System.out.println(v);
   }
}
```

运算符与表达式

- Java常用的运算符分为五类：算术运算符、赋值运算符、关系运算符、布尔逻辑运算符、位运算符。位运算符除了简单的按位操作外，还有移位操作。按位操作返回布尔值。
- 表达式是由常量、变量、对象、方法调用和操作符组成的式子。表达式必须符合一定的规范，才可被系统理解、编译和运行。表达式的值就是对表达式自身运算后得到的结果。
- 根据运算符的不同，表达式相应地分为以下几类：算术表达式、关系表达式、逻辑表达式、赋值表达式，这些都属于数值表达式。

运算符与表达式

- 算术运算符
  - 加法运算符： + 和 -
  - 乘法运算符： *, /, 和 % （取模）
  - 运算符： ++ 和 --，例如： i++; --j;
- 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

运算符与表达式

- 关系运算符
  - 关系表达式的类型永远是布尔类型（boolean）.
  - 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

```
Point  p;
if (e instanceof Point ) //若e 所引用的对象是Point类型的
        p=(Point)e;
```

  - 相等关系运算符: == , !=

运算符与表达式

- 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：！(一元运算符，例如:!a)

    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。

运算符与表达式

- 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
  - 例如：result＝(score>=60)? “通过”：“未通过”;

运算符与表达式

- 位逻辑运算符
  - 按位与：&　（位都为1则为1，否则为0）
  - 按位或：|　（位有一个为1则为1，否则为0）
  - 按位异或：^　（位不同则为1，相同则为0）
  - 按位取反：~　（0变为1，1变为0）
- 位移运算符
  - 左移：<<　（x<<n的结果是x乘以2的n次方）
  - 右移：>>　（x>>n的结果是x除以2的n次方）
  - 不带符号右移：>>>

运算符与表达式

● 运算符优先级

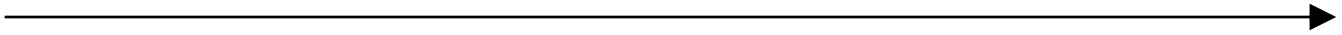
优先级	含义描述	运 算 符	结合性
1	分隔符	[] () ; ,	
2	单目运算、字符串运算	++ -- + - ~ ! (类型转换符)	*右到左
3	算术乘除运算	* / %	左到右
4	算术加减运算	+ -	左到右
5	移位运算	<< >> >>>	左到右
6	大小关系运算、类运算	< > <= >= instanceof	左到右
7	相等关系运算	== !=	左到右
8	按位与	&	左到右
9	按位异或	^	左到右
10	按位或	|	左到右
11	与	&&	左到右
12	或	||	左到右
13	三目条件运算	?:	*右到左
14	简单、复杂赋值运算	= *= /= %= += -= <<= >>= >>>=	*右到左

类型转换

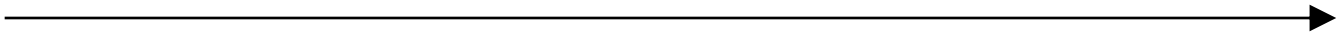
- 每个表达式都有类型
- 如果表达式的类型对于上下文不合适
  - 有时可能会导致编译错误
  - 有时语言会进行隐含类型转换

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。

类型转换

- 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
  - float prc=Float.parseFloat(“5.0”);
- 强制转换
  - 将一个表达式转换为指定的类型，例如： (float)5.0
- 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

类型转换

- 不能从Double自动转化为float
  - float f1=1D; //编译出错
  - float f2=1.2; //编译出错
  - float f3=1.2F; // 正确

- 宽类型数据到窄类型数据的转换应使用：强制转换
  - float f1=(float) 1D;
  - float f2=(float)1.2;

标准输入输出简介

- 标准输入流 System.in
  - public static final InputStream in

  例如：

  byte[] b= new byte(10);

  System.in.read(b); //从输入流中读取字节至缓冲b
- 标准输出流 System.out
  - public static final PrintStream out

  例如： System.out.print("Hello world!");

  System.out.println( ); //换行

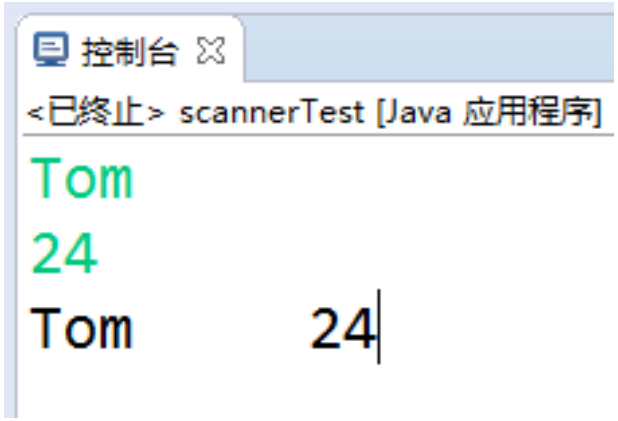
  System.out.println("Hello world! ");//输出并换行

标准输入输出简介

- 例：从键盘输入数据，并输出到屏幕

```
import java.util.Scanner;
public class scannerTest {
   public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        String name = scan.next();
        int age = scan.nextInt();
        scan.close();


        System.out.println(name + "\t" + age);
   }
}
```



流程控制

- 流程控制分为三种基本结构：
  - 顺序结构是指命令行顺序执行，最常见的格式；
  - 分支结构是一种选择结构，根据条件的值选择不同的执行流程，可以得到不同的结果。
    - 单分支语句(if-else语句)
    - 多分支语句(switch语句)
    - 条件运表达式（a？b：c)
  - 循环结构是指对于一些重复执行的语句，用户指定条件或次数，由机器自动识别执行。
    - 次数循环语句(for语句)
    - 条件循环语句(while语句、 do …while语句)

流程控制：分支语句

- if-else语句

```
if(布尔表达式1){
    语句或块1;
}
else if(布尔表达式2){
    语句或块2;
}
else {
    语句或块3;
}
```

```
public class TestIf{
    public static void main(String args[]){
            int x = 70;
           String  result= "" ;
            if(x>=60)
                result= "及格" ;
            else
                result= "不及格" ;
           System.out.println(result);
    }
}
```

流程控制：分支语句

- switch语句

```
switch(表达式1){
    case 表达式2:
        语句或块2;
        break;
    …
    case 表达式n:
        语句或块n;
        break;
    default:
        语句或块n+1;
        break;
}
```

其中：

(1) 表达式1的值必须与整型兼容、或为字符串类型，或为枚举类型。

(2) case分支要执行的程序语句。

(3) 表达式2~n是可能出现的值，必须为常量。

(4) 不同的case分支对应着不同的语句或块序列。

(5) break表示跳出switch。

流程控制：分支语句

- switch语句

```
switch(表达式1){
    case 表达式2:
        语句或块2;
        break;
    …
    case 表达式n:
        语句或块n;
        break;
    default:
        语句或块n+1;
}
```

```
switch(x){
        case 1:
            result= "初级" ;
             break;
        case 2:
            result= "中级" ;
             break;
        case 3:
        case 4:
            result= "高级" ;
             break;
        default:
            result= "级别错误" ;
}
System.out.println(result);
```

流程控制：循环语句

- 两种for循环语句
  - 实现已知次数的循环，有两种形式：一般的for循环和增强的for循环
  - 一般的for循环，其基本格式为：

```
for(初始化表达式; 测试表达式; 修改循环变量){
    语句或块;
}
```

  - 其执行顺序如下：

(1) 首先运行初始化表达式。

(2) 然后计算测试表达式，如果表达式为true，执行语句或块；如果表达式为false，退出for循环。

(3) 修改循环变量，转步骤(2)。

流程控制：循环语句

【例】用for循环计算1～100之间整数的和

```
public class TestFor{
    public static void main(String args[]){
        int sum = 0;
        for(int i = 1; i<=100; i++)
            sum += i;
        System.out.println("1+2+…+100=" + sum);
    }
}
```

流程控制：循环语句

- 两种for循环语句
  - 增强的for循环用于遍历集合或数组，其格式为：

```
for(<元素类型> 循环变量x : 集合/数组S ){
    语句或块;
}
```

  - 其执行顺序如下：

(1) 如果S为空，则不进行循环；否则用S的第一个元素初始化x。

(2) 执行循环体。

(3)将S的下一个元素赋给x，转(2)，直到顺序访问S中的最后一个元素，结束循环。

流程控制：循环语句

【例】用增强的for循环遍历数组

```
public class TestFor2{
    public static void main(String args[]){
        int sum = 0;
        int[] S={2,5,9,3,7};
        for(int x : S)
            sum += x
        System.out.println("sum=" + sum);
    }
}
```

流程控制：循环语句

- while循环语句
  - 受条件控制的循环，其基本格式为：

```
while(布尔表达式) {
        语句或块;
}
```

  - 当布尔表达式为true时，执行语句或块，否则跳出while循环。

```
//1+2+…+100
int sum = 0, i = 1;
while(i<=100){
        sum += i;
    i++;
}
```

流程控制：循环语句

- do…while循环语句
  - 受条件控制的循环，其基本格式为：

```
do{
        语句或块;
} while(布尔表达式)
```

```
//1+2+…+100
int sum = 0, i = 1;
do{
    sum += i;
    i++;
} while(i<=100)
```

  - 先执行语句或块，然后再判断布尔表达式。与while语句不同，当布尔表达式一次都不为true时，while语句一开始判断就跳出循环，不执行语句或块，而在do语句中则要执行一次。

流程控制：循环语句

- 终止循环或跳过本次循环而开始下一次循环
  - break;
  - continue;

```
//1+2+…+100
int sum = 0, i = 1;
while(true) {
    if(i>100)  break;
    sum += i;
     i++;
}
```

```
//1+2+4+5+6+8…+100
// 跳过是3或7的整数倍的数
int sum = 0, i = 1;
while(true) {
   if(i%3==0|| i%7==0)
        continue;
    sum += i;
     i++;
}
```

变量作用域问题

- Java循环等代码块内部可以访问到外部的变量，但是外部无法访问内部的变量，因为内部变量是局部的。

C/C++是合法的

```
//报错：for中重复声明变量i
int i = 0;
for(int i = 0;i<10;i++){…}
```

```
// 可以访问外部变量
int i = 0;
for(;i<10;i++){…}
```

```
// 编译通过
for(int i = 0;i<10;i++){…}
int i = 0; // for中的变量i消失了
```

```
// 可以访问外部变量
int i = 0;
for(i=0;i<10;i++){…}
```

```
//报错：k 无法解析为变量
for(int i=0;i<10;i++){ int k=0;}
k++;
```

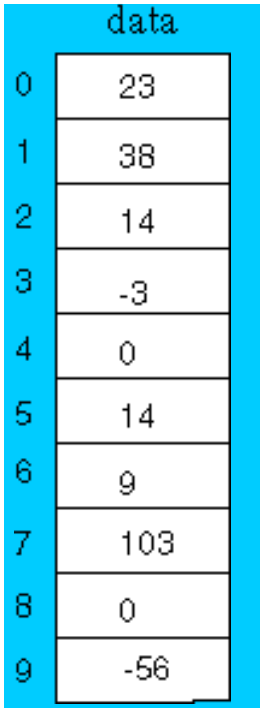
外部不能访问内部的变量

数组

- 数组的概念
- 数组的声明
- 数组的创建
- 数组的初始化
- 数组的引用
- 多维数组

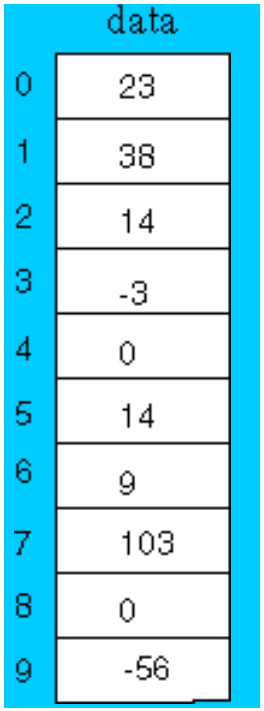
数组的概念

- 数组由同一类型的一连串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。
- Java数组是一个对象
  - 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



数组的概念

- 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。
  - 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）。



数组的声明

- 声明（Declaration）
  - 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

```
Type[ ] arrayName;
  例如：int[ ] intArray;
        String[ ]  stringArray;
Type  arrayName[ ];
  例如：int intArray[ ];
        String  stringArray[ ];
```

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

arryName=new Type[componets number];

- 例如：

```
int[ ] ai;    ai=new int[10]; //各元素初始化为0
String[ ] s;   s=new String[3]; //各元素初始化为null，因为此时String对象没有创建
```

- 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。
- 可以在数组被构造之后改变数组元素值。

数组的引用

- 通过下面的表达式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** 引用。
- 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）。

数组的引用

```
int values[ ] = new int[7];
int index;
index  = 0;
values[ index ] = 71;
index  = 5;
values[ index ] = 23;
index  = 3;
values[ 2+2] = values[ index-3 ];
```

- values[-1]　非法
- values[7]　非法
- values[1.5]　非法
- values[0]　合法
- values[6]　合法

数组的引用

【例】验证整型数组元素默认初始化为0

```
public class MyArray {
    public static void main(String[ ] args){
        int myArray[ ];                    //声明数组
        myArray=new int[10];               //创建数组

        //验证整型数组元素默认初始化为0
        System.out.println("Index\t\tValue");
        for(int i=0; i<myArray.length;i++)
            System.out.println(i+"\t\t"+myArray[i]);
        //myArray[10]=100;      //将产生数组越界异常
    }
}
```

数组的引用

```
class Gauss
{   public static void main(String[] args)
    {   int[ ] ia = new int[101];
        for (int i = 0; i < ia.length; i++)
            ia[i] = i;
        int sum = 0;
        for (int i = 0; i < ia.length; i++)
            sum += ia[i];
        System.out.println(sum);
    }
}
```

输出:

5050

数组的引用

- 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; //引用同一个数组
       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
```

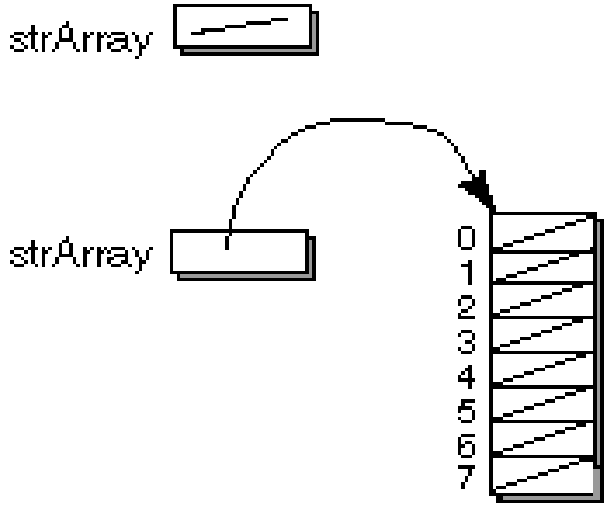
数组的引用

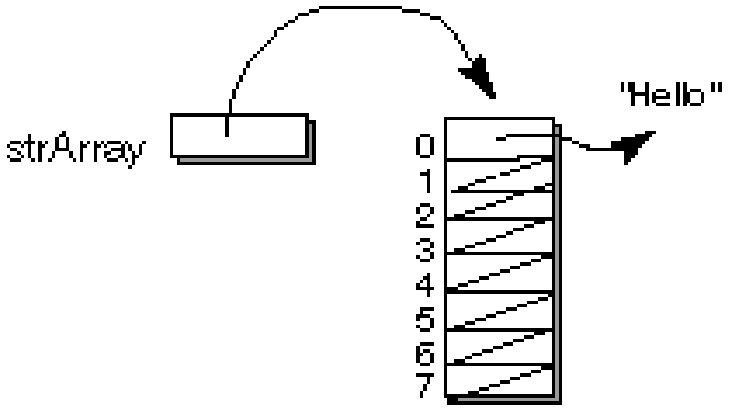
- 字符串引用构成的数组：

```
String[ ] strArray;

strArray = new String[8] ;

strArray[0]= "Hello";
```





- 注意，此时并没有创建对象元素，仅仅是创建了一个可以存储对象引用的数组！
- 注意与基本数据类型数组的区别。

数组的引用

【例】使用字符串数组

```
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());
    }
}
```

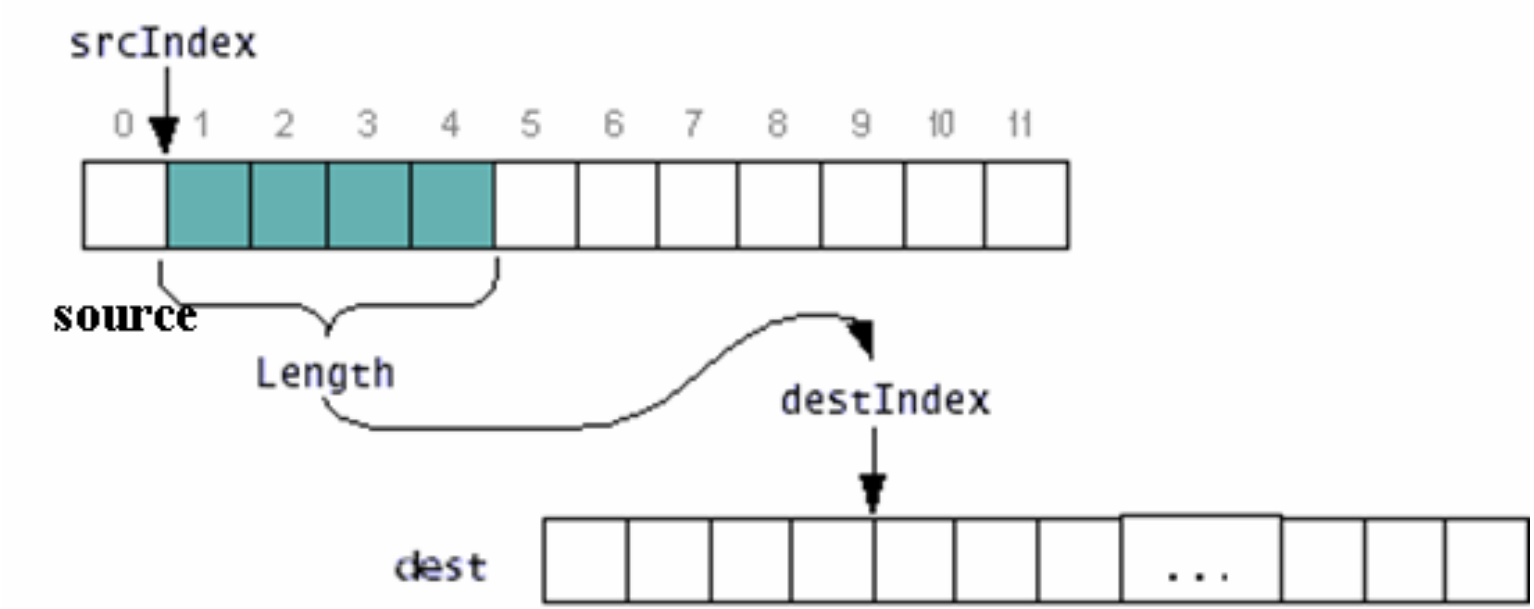
运行结果:

string one
string two
string three

数组的引用

- 数组的复制

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



数组的引用

【例】复制数组

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

运行结果：

caffein

多维数组

- 一个二维表格：

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

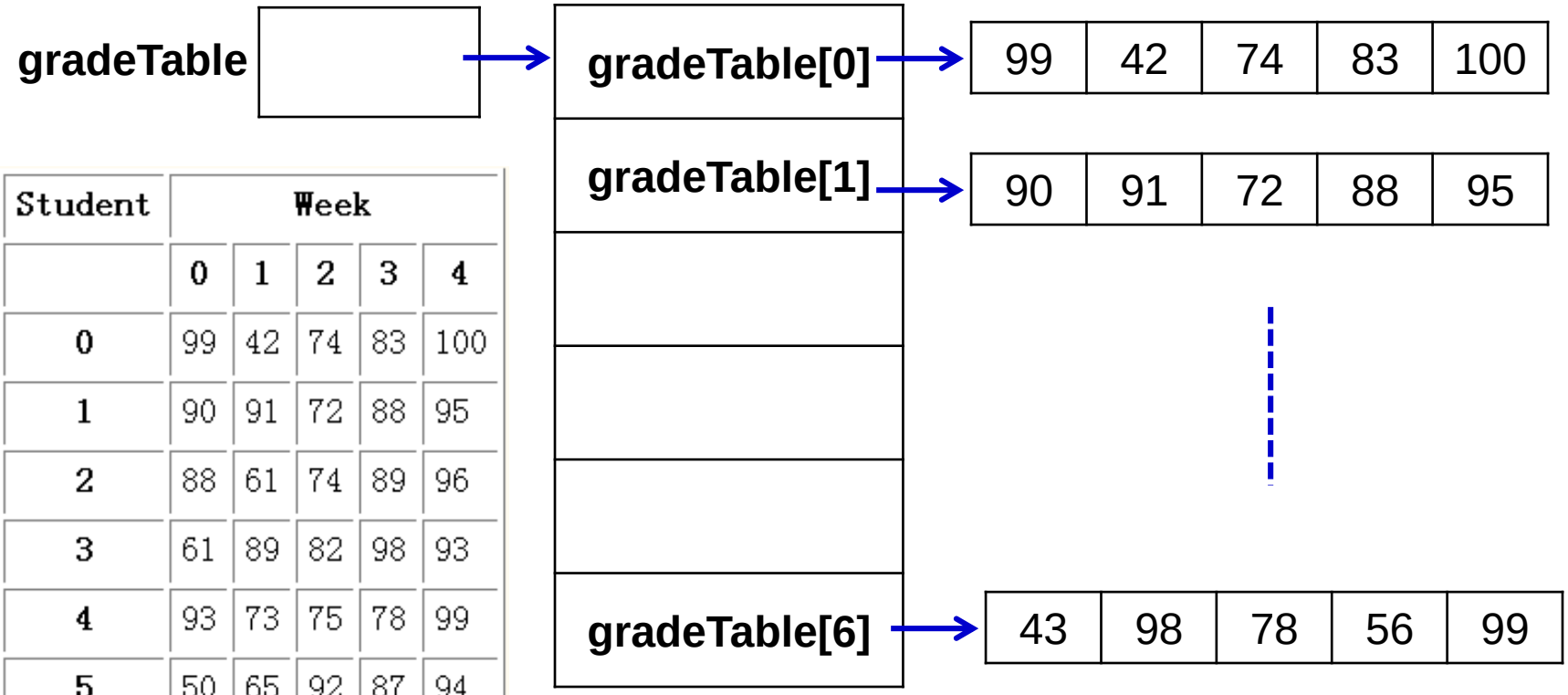
多维数组

- 二维数组的声明和构造
  - int[ ][ ] myArray ;
    - myArray 可以存储一个指向2维整数数组的引用。其初始值为null。
  - int[ ][ ] myArray = new int[3][5] ;
    - 建立一个数组对象，把引用存储到myArray。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} };
    - 建立一个数组并为每一个元素赋值。

多维数组

- 二维数组是引用（指向）一维数组的数组

```
int[ ][ ] gradeTable; // 假定已经构造好
```



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

多维数组

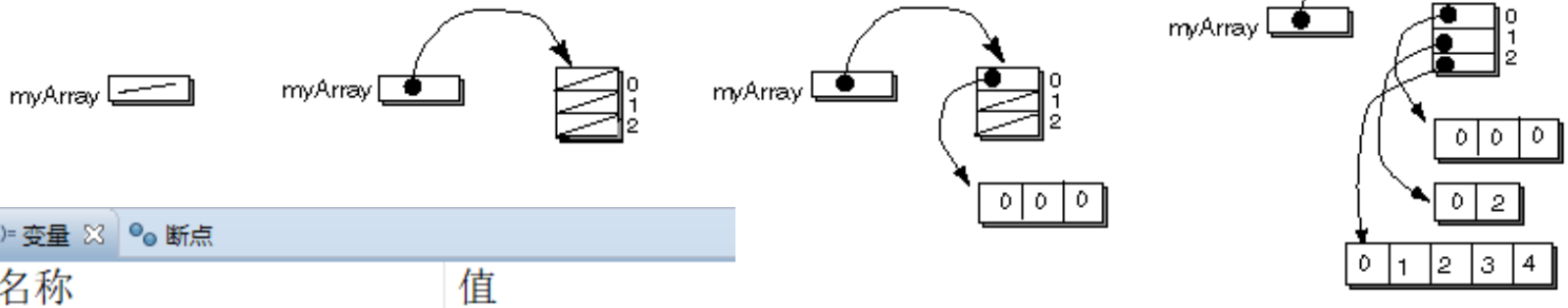
- 二维数组的长度
  - **gradeTable.length**为7， 每行（为一个一维数组）的长度**gradeTable[i].length**为5
  - int[ ][ ] uneven = { { 1, 9, 4 },
    { 0, 2},
    { 0, 1, 2, 3, 4 } };
    - uneven.length为3
    - 第1行数组的长度uneven[0].length为3
    - 第2行数组的长度uneven[1].length为2
    - 第3行数组的长度uneven[2].length为5

多维数组

- 分步构造二维数组

```
int[ ][ ] myArray;
myArray = new int[3][ ] ;
myArray[0] = new int[3];
```

```
int[ ] x = {0, 2};
int[ ] y = {0, 1, 2, 3, 4} ;
myArray[1] = x ;
myArray[2] = y ;
```



myArray[1]与x引用的对象具有相同标识

习 题

1. 编写程序，统计课程编号为1001、2001和3001的平均成绩并输出。学生成绩表如图所示。【课程编号存储在一个字符串数组中coursesID，成绩存储在一个二维数组scores，第k个课程编号与成绩数组中第k行对应】
2. 给上述3门课的平均成绩评定等级并输出：优（91~100）、良(80~90)、中(70~79)、及格（60~69）和不及格（0~59）。
3. 输出九九乘法表。

B

学号	课程编号	成绩
02034001	1001	90
02034002	1001	85
02037006	1001	66
02037007	1001	51
02037005	1001	69
02035001	1001	78
02034003	1001	88
02037002	1001	94
02034001	3001	68
02034002	3001	88
02037007	3001	81
02037006	3001	90
02037005	3001	65
02035001	3001	74
02037002	3001	64
02034003	3001	46
02034001	2001	68
02034002	2001	90
02037005	2001	85
02035001	2001	88
02037002	2001	68
02034003	2001	70
02034002	1002	60

谢谢大家!