第3章 数据运算、流控制和数组

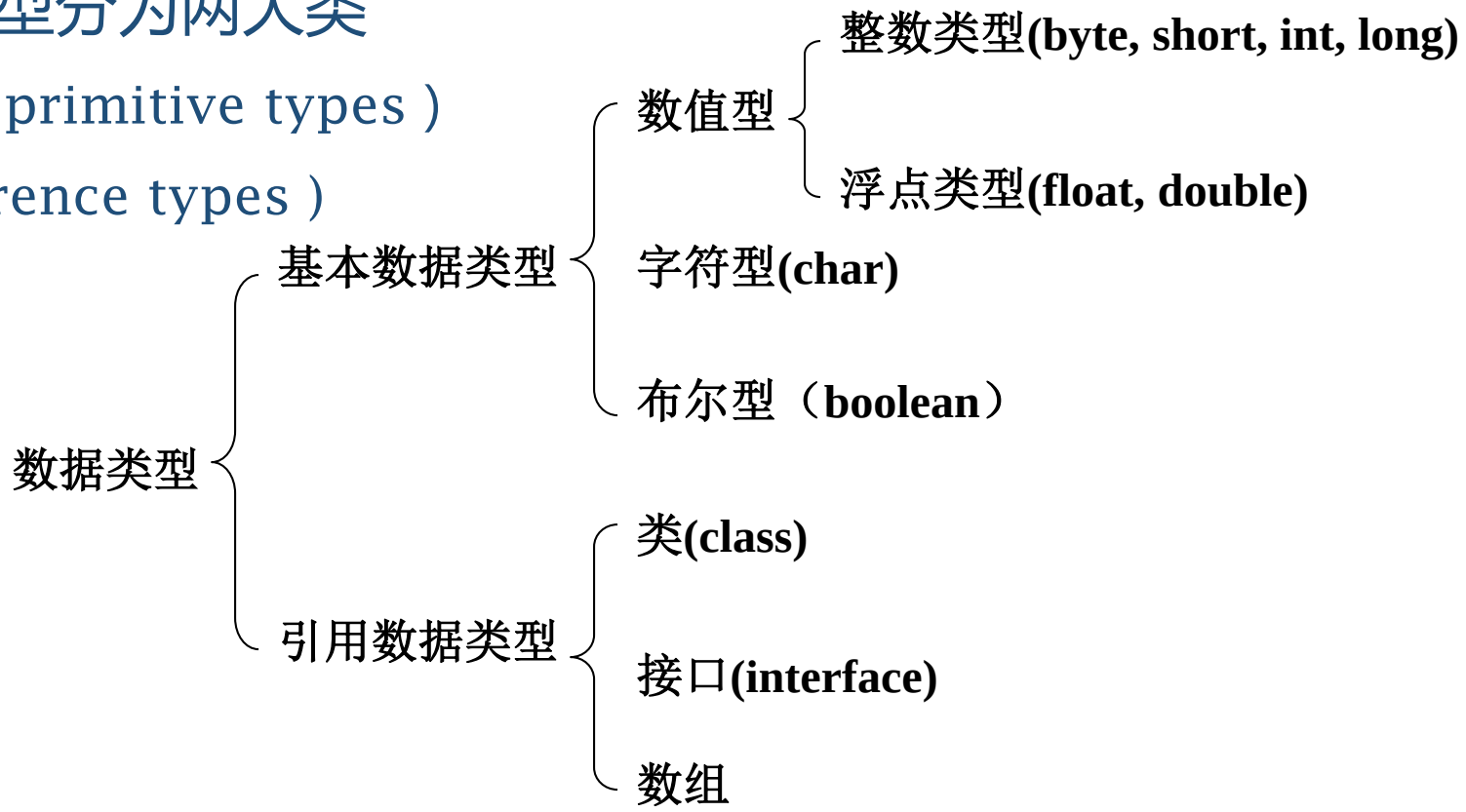
- 3.1 数据类型、变量与常量
- 3.2 运算符与表达式
- 3.3 流程控制语句
- 3.4 数组

3.1 数据类型、变量与常量

数据类型、变量与常量

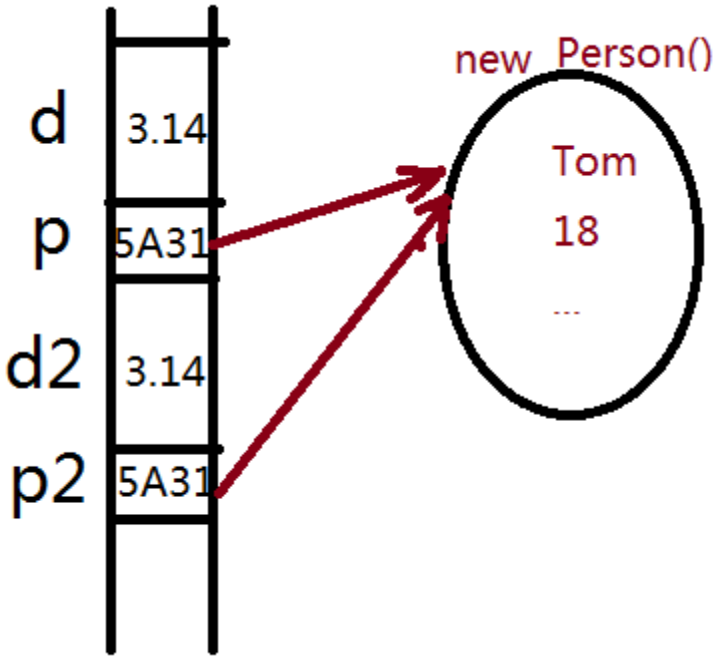
Java数据类型划分

- 数据类型决定数据的存储方式和运算方式
- Java中的数据类型分为两大类
  - 基本数据类型 (primitive types)
  - 引用类型(reference types)



两种类型的差别

- 基本类型： 变量在栈，在“这里”
- 引用类型： 变量引用到堆，在“那里”
  - double d = 3;
  - Person p = new Person();



- 赋值时
  - double d2 = d; 复制的是值
  - Person p2 = p; 复制的是引用
-

Java基本数据类型

- Java中定义了四类/八种基本数据类型
  - 整数型---- byte, short, int, long
  - 浮点数型---- float, double
  - 逻辑型---- boolean
  - 字符型---- char

逻辑型

- boolean类型适于逻辑运算，一般用于程序流程控制
- boolean类型数据只允许取值true或false
  - 不可以0或非0的整数替代true和false
  - if(a=5)在java中是不允许的
- 用法举例：

```
boolean  b = false;
if(b==true)  {
        //do something
}
```

字符型

- char型数据用来表示通常意义上“字符”
- 字符常量是用单引号括起来的单个字符
  - char c = 'A';
- Java字符采用Unicode编码，每个字符占两个字节，
  - 可用十六进制编码形式表示
  - char c1 = '\u0061';
- Java语言中还允许使用转义字符'\'来将其后的字符转变为其它的含义
  - char c2 = '\n';　　//代表换行符

转义符

- 转 义 字 符　　含　　义
- \ddd 1到3位八进制数所表示的字符(ddd)
- \uxxxx　　1到4位十六进制数所表示的字符(xxxx)
- \'　　单引号字符
- \"　　双引号字符
- \\　　反斜杠字符
- \r　　回车
- \n　　换行
- \f　　走纸换页
- \t　　横向跳格
- \b　　退格

整数类型(1)

- Java各整数类型有固定的表数范围和字段长度，而不受具体操作系统的影响，以保证Java程序的可移植性

类　型	占用存储空间	表数范围
byte	1字节	-128 ~ 127
short	2字节	$-2^{15} \sim 2^{15}-1$
int	4字节	$-2^{31} \sim 2^{31}-1$
long	8字节	$-2^{63} \sim 2^{63}-1$

整数类型(2)

- Java语言整型常量的三种表示形式：
  - 十进制整数，如12, -314, 0。
  - 八进制整数，要求以0开头，如012
  - 十六进制数，要求0x或0X开头，如0x12
  - 二进制数，以0b或0B开头，如0b00010010（Java7以上）
- Java语言的整型常量默认为int型，如：
- int i =3;
- 声明long型常量可以后加‘ l ’或‘ L ’，如：
- long l = 3L;

- Java中没有“无符号数”
  - 可以用long来处理无符号整数（uint）

浮点型(1)

- Java浮点类型有固定的表数范围和字段长度

类　型	占用存储空间	表数范围
float	4字节	-3.403E38~3.403E38
double	8字节	-1.798E308~1.798E308

浮点型(2)

- Java浮点类型常量有两种表示形式
  - 十进制数形式，必须含有小数点，例如:
  - 3.14　　314.0　　.314
    - Java7以上：　123_456.789_000（千分位分割符用下划线表示）
  - 科学记数法形式，如
  - 3.14e2　　3.14E2　　314E2
- Java浮点型常量默认为double型,
  - 如要声明一个常量为float型，则需在数字后面加f或F，如：
- double d = 3.14;
- float f = 3.14f;

变量声明和赋值

```
public class Test {
        public static void main (String args []) {
            boolean b = true;     //声明boolean型变量并赋值
            int x, y=8;           // 声明int型变量
            float f = 4.5f;       // 声明float型变量并赋值
            double d = 3.1415;    //声明double型变量并赋值
            char c;               //声明char型变量
            c = '\u0031';         //为char型变量赋值
            x = 12;               //为int型变量赋值
        }
}
```

标识符（Identifier）

- 名字就是标识符：任何一个变量、常量、方法、对象和类都需要有名字。
- 标识符要满足如下的规定：
  - （1）标识符可以由字母、数字和下划线(_)、美元符号($)组合而成；
  - （2）标识符必须以字母、下划线或美元符号开头，不能以数字开头。
- 标识符最好与其意义相符，以增加程序的可读性
- 应注意Java是大小写敏感的语言。
  - 按Java惯例，类名首字母用大写（Pascal)
  - 其余的（包名、方法名、变量名）首字母都小写(camel)
  - 少用下划线
  - 变量、常量随使用随定义

3.2 运算符与表达式

运算符与表达式

运算符

- 算术运算符: +，—，*，/，%，++，--
- 关系运算符: >，<，>=，<=，==，!=
- 逻辑运算符: !，&，|，^，&&，||
- 位运算符: &，|，^，~ ，>>，<<，>>>
- 赋值运算符: = 扩展赋值运算符:+=，—=，*=，/=
- 字符串连接运算符: +

算术运算符

- +，—，*，/，%，++，——
- 有关 /　　15/4 (整除)　15.0/2 (实数除法)
- 有关 %　　100%3　100%-3　-100%-3　-100%3
- 有关%的含义　　偶数 a %2, 整除 a%7, 个位 a%10
- 有关++, --　　a=5; a++;　b=a*2
- 　　a=5; b = ++ a *2;
- 　　a=5; b = a++ *2;
- ^不是乘方

逻辑运算符(1)

- 逻辑运算符功能
- ！ 逻辑非　　& 逻辑与　　| 逻辑或
- ^ 逻辑异或　　&& 短路与　　|| 短路或
- 逻辑运算符功能说明:

a	b	!a	a&b	a|b	a^b	a&&b	a||b
true	true	false	true	true	false	true	true
true	false	false	false	true	true	false	true
false	true	true	false	true	true	false	true
false	false	true	false	false	false	false	false

逻辑运算符(2)

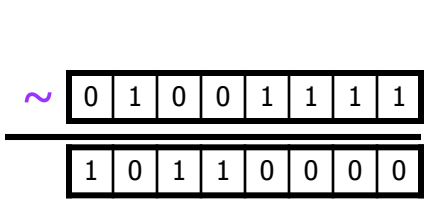
- 短路(short-circuit)逻辑运算符
- && -- 第一个操作数为假则不判断第二个操作数
- || -- 第一个操作数为真则不判断第二个操作数
- MyDate d;

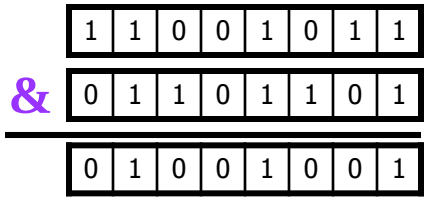
```
if ((d!=null) && (d.day >31)) {
        //do something with d
}

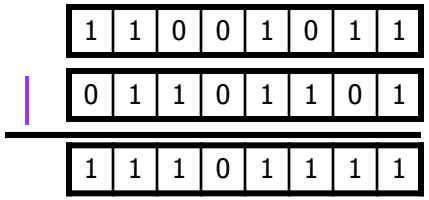
if(i <0 || i >31 ) {
        System.out.println("非法赋值");
}
```

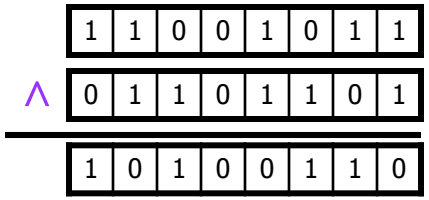
位运算符

- 位运算符功能
- ~ 取反 & 按位与 | 按位或 ^ 按位异或
- 位运算符功能说明:









移位运算符(1)

- 左移
  - "a<<b; "将二进制形式的a逐位左移b位，最低位空出的b位补0；
- 带符号右移
  - "a>>b; "将二进制形式的a逐位右移b位，最高位空出的b位补原来的符号位；
- 无符号右移
  - "a>>>b;"将二进制形式的a逐位右移b位，最高位空出的b位补0。

移位运算符(2)

- 移位运算符性质
  - 适用数据类型:byte、short、char、int、long
  - 对低于int型的操作数将先自动转换为int型再移位（整型提升，对所有的运算都是这样）
  - 对于int型整数移位a>>b，系统先将b对32取模，得到的结果才是真正移位的位数
  - 对于long型整数移位时a>>b ，则是先将移位位数b对64取模

移位运算符(3)

- 移位运算符应用举例
- 示例： BitwiseOperators.java

2227 =	00000000	00000000	00001000	10110011
2227<<3 =	00000000	00000000	01000101	10011000
2227>>3 =	00000000	00000000	00000001	00010110
2227>>>3 =	00000000	00000000	00000001	00010110
-2227 =	11111111	11111111	11110111	01001101
-2227<<3 =	11111111	11111111	10111010	01101000
-2227>>3 =	11111111	11111111	11111110	11101001
-2227>>>3 =	00011111	11111111	11111110	11101001

赋值运算符(1)

- 赋值运算符=
  - 当“=”两侧的数据类型不一致时，可以适用默认类型转换或强制类型转换(casting)原则进行处理
  - long l = 100;
  - int i = (int)l;
  - 特例:可以将整型常量直接赋值给byte, short, char等类型变量，而不需要进行强制类型转换，只要不超出其表数范围
  - byte b = 12; //合法
  - byte b = 4096; //非法

赋值运算符(2)

- 扩展赋值运算符

运算符	用法举例	等效的表达式
+=	a += b	a = a+b
-=	a -= b	a = a-b
*=	a *= b	a = a*b
/=	a /= b	a = a/b
%=	a %= b	a = a%b
&=	a &= b	a = a&b
|=	a |= b	a = a|b
^=	a ^= b	a = a^b
<<=	a <<= b	a = a<<b
>>=	a >>= b	a = a>>b
>>>=	a >>>= b	a = a>>>b

字符串连接运算符 +

- "+" 除用于算术加法运算外，还可用于对字符串进行连接操作
- ```
  int i = 300 +5;
  ```
- ```
  String s = "hello, " + "world!";
  ```
- "+"运算符两侧的操作数中只要有一个是字符串(String)类型，系统会自动将另一个操作数转换为字符串然后再进行连接
- ```
  int i = 300 +5;
  ```
- ```
  String s = "hello, " + i  + "号";
  ```
- ```
  System.out.println(s);   //输出：hello, 305号
  ```

- 该运算符大大简化了字符串的处理
  ```

表达式

- 表达式是符合一定语法规则的运算符和操作数的序列
  - a
  - 5.0 + a
- (a-b)*c-4
- i<30 && i%10!=0
- 表达式的类型和值
  - 对表达式中操作数进行运算得到的结果称为表达式的值
  - 表达式的值的数据类型即为表达式的类型
- 表达式的运算顺序
  - 首先应按照运算符的优先级从高到低的顺序进行
  - 优先级相同的运算符按照事先约定的结合方向进行

运算符优先级与结合性

- 适当使用括号

Separator	. ( ) { } ; ,

Associative	Operators
R to L	++ -- ~ ! (data type)
L to R	* / %
L to R	+ -
L to R	<< >> >>>
L to R	< > <= >= instanceof
L to R	== !=
L to R	&
L to R	^
L to R	|
L to R	&&
L to R	||
R to L	?:
R to L	= *= /= %= += -= <<= >>= >>>= &= ^= |=

表达式中的类型转换

- 当有不同种类的混合运算时:
- int→long→float→double
- 整型提升
  - 所有的byte, short, char 参与算术运算等转为int

问题

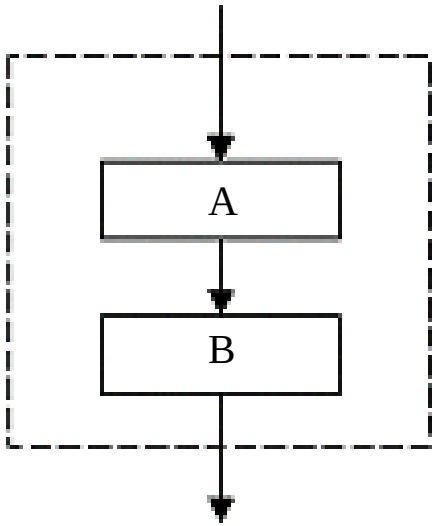
- x=3,y=4,z=5;
- !(x+y)+z-1&&y+z/2
  - 在javascript及TurboC中是不同的
  - http://emuch.net/fanwen/427/56083.html
- a=2;　b=a++ + ++a;
- a++的副作用
- 查看反汇编的代码
  - 使用javap -c 类名

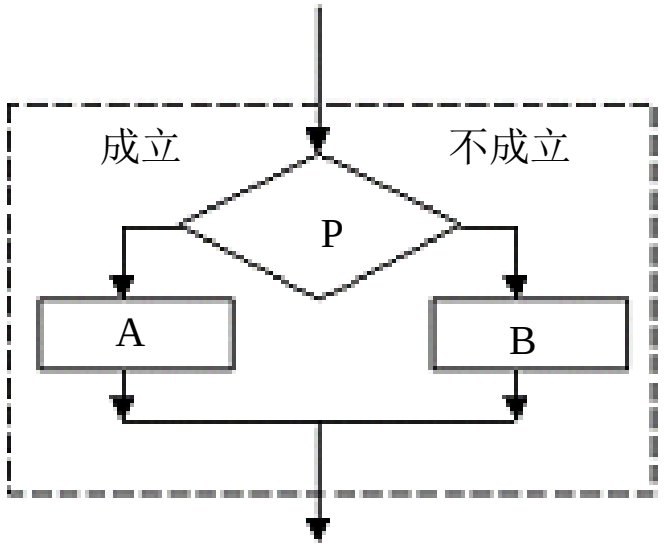
3.3 流程控制语句

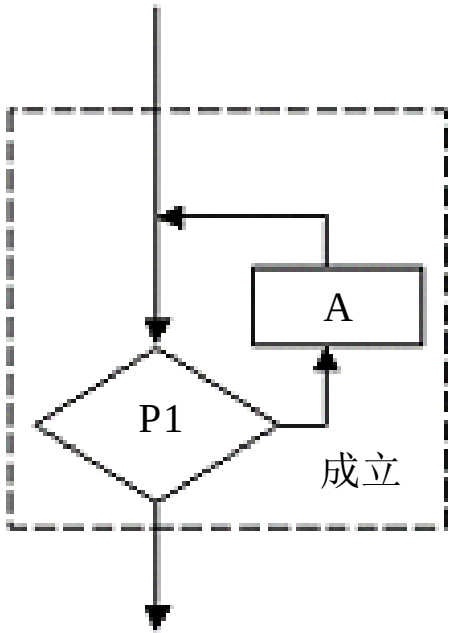
流程控制语句

程序的三种基本流程

- 顺序　　分支　　循环







简单语句

- 最简单的语句
  - 方法调用语句
  - 赋值语句，注意分号（ ； ）
- 如：
- System.out.println(“Hello World”);
- a = 3+x;
- b = a>0?a:-a;
- s = TextBox1.getText();
- d = Integer.parseInt(s);

问题

- 没有“表达式语句”这个概念
- x+y;是不合法的

程序的注释

- Java中可以采用三种注释方式：
- // 用于单行注释。注释从//开始，终止于行尾；
- /* … */ 用于多行注释。注释从/*开始，到*/结束，且这种注释不能互相嵌套；
- /** … */ 是Java所特有的doc注释。它以/**开始，到*/结束。

JavaDoc

- 其中，第3种注释主要是为支持JDK工具javadoc而采用的。javadoc能识别注释中用标记@标识的一些特殊变量，并把doc注释加入它所生成的HTML文件。常用的@标记如下。
- @see：引用其他类。
- @version：版本信息。
- @author：作者信息。
- @param：参数名说明。
- @return：说明。
- @exception：异常说明。
- 对于有@标记的注释，javadoc在生成有关程序的文档时，会自动地识别它们，并生成相应的文档。
- 如：
- 生成：
- javadoc HelloDate.java

分支语句--if

- if(条件表达式)
- 　语句块；　　// if分支
- else
- 　语句块；　　// else分支

- 例：LeapYear.java

问题

- if(x>=0) if(x>0) y=1;else y=0;else y=-1;

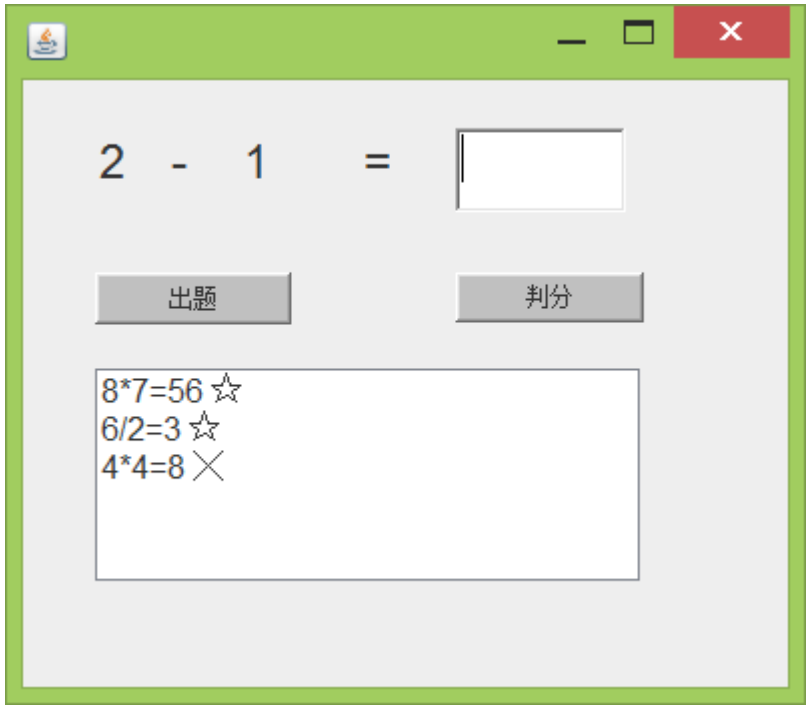
分支语句-- switch语句

- 使用switch要注意：
  - 变量类型是整数、字符、字符串（String）
  - case后面是常量
  - 注意 break
- 例：GradeLevel.java 分数等级
-

```
switch(exp){
        case const1:
                statement1;
                break;
        case const2:
                statement2;
                break;
        … …
        case constN:
                statementN;
                break;
        [default:
                statement_dafault;
                break;]
}
```

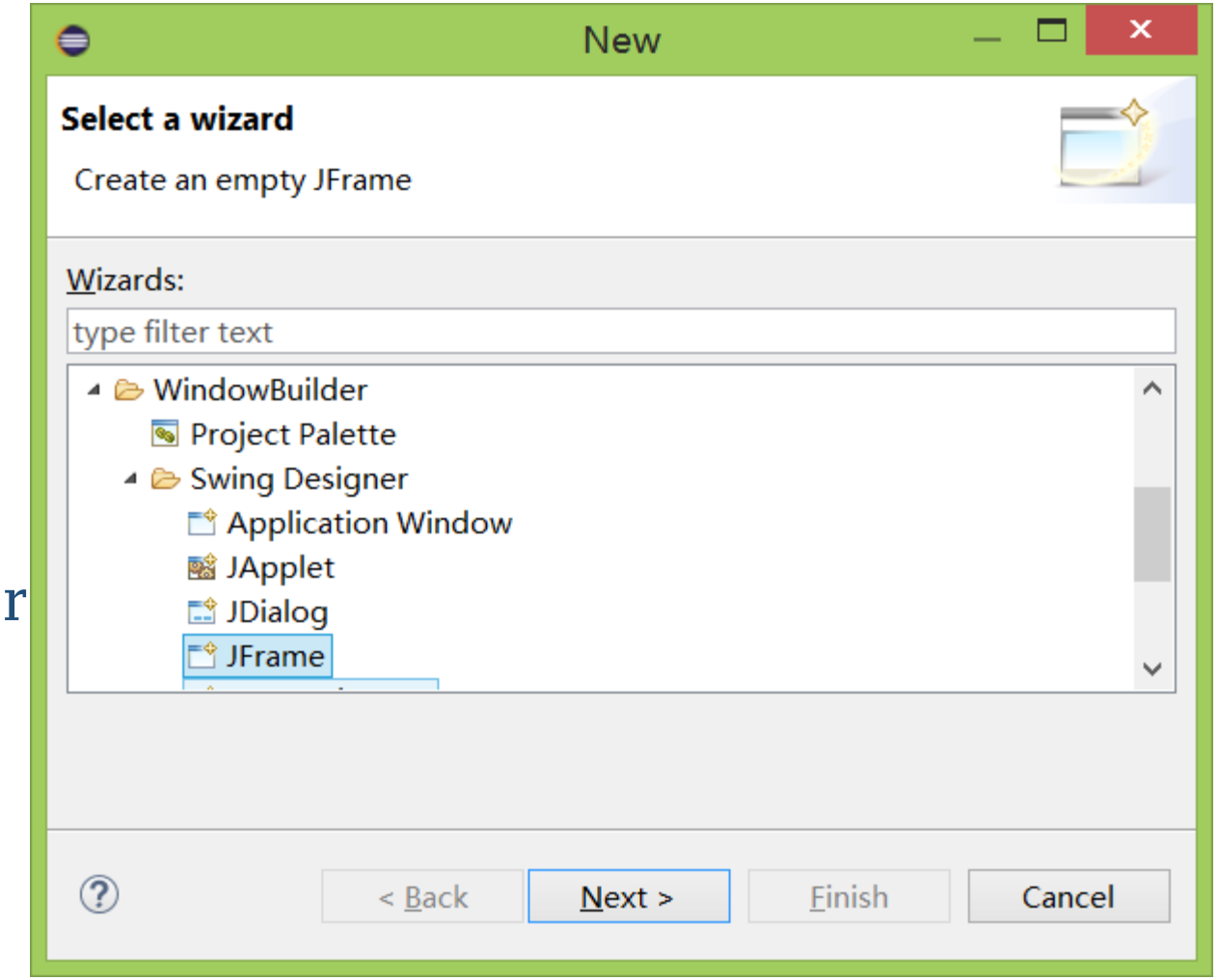
应用举例

- AutoScore.java 自动出题并判分



IDE中的窗体设计

- Eclipse中
  - 项目上点右键
  - New—Other—Windows Builder
    - —Swing Designer—Jframe
  - 在窗体上右键，Layout,(absolute)
  - 加上按钮等组件，设置其属性
  - 组件上右键，Add New Event Handler
- Netbeans类似
  - 新建—Other—Swing GUI窗体
  - 布局可设为null

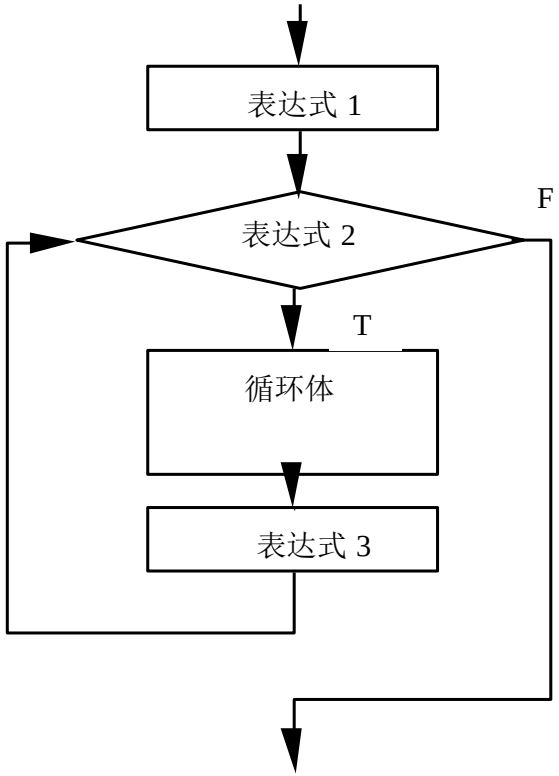


循环语句

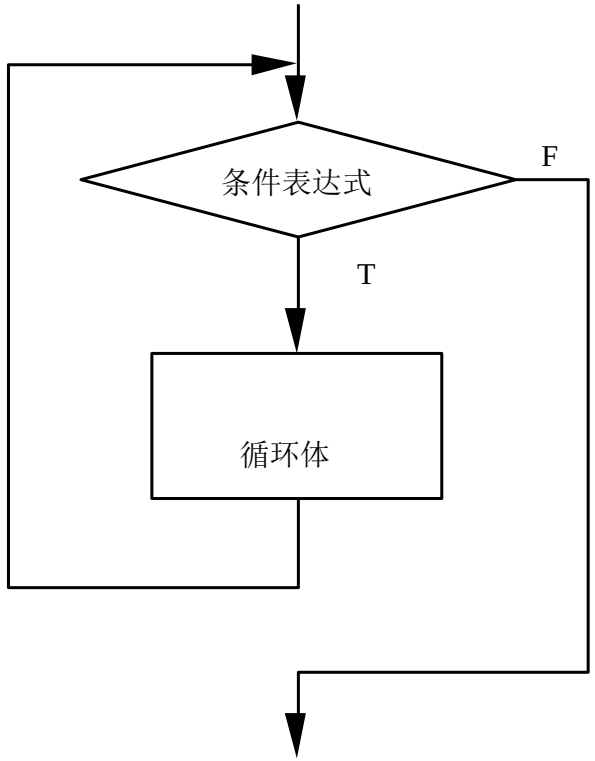
- 循环语句功能
  - 在循环条件满足的情况下，反复执行特定代码
- 循环的五个要素
  - 初始化部分（init_statement）
  - 循环条件部分（test_exp）
  - 循环体部分（body_statement）
  - 迭代部分（alter_statement）
  - 结束后处理

循环语句的三种写法

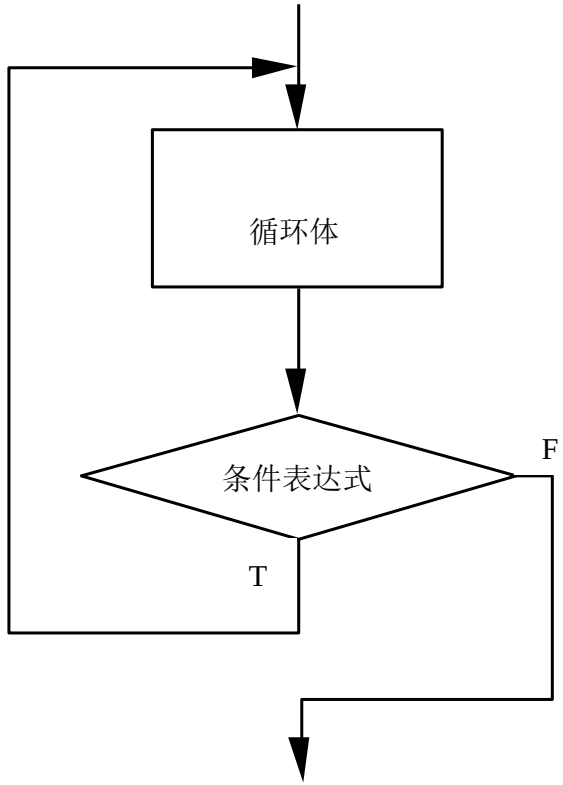
- for 循环



while 循环



do/while 循环



for 循环语句

- 语法格式
- for (init_statement; test_exp; alter_statement) {
- body_statement
- }
- 应用举例
- int result = 0;
- for(int i=1; i<=100; i++) {
- result += i;
- }
- System.out.println("result=" + result);
-

while 循环语句

- 语法格式

```
[init_statement]
while( test_exp) {
        body_statement;
        [alter_statement;]
}
```

- 应用举例

```
int result = 0;
int i=1;
while(i<=100) {
    result += i;
    i++;
}
System.out.println("result=" + result);
```

- 注意不要死循环

do/while 循环语句

- 语法格式

```
[init_statement]
do {
    body_statement;
    [alter_statement;]
 } while( test_exp);
```

- 应用举例

```
int result = 0, int i=1;
do{
    result += i;
    i++;
}while(i<=100);
System.out.println("result=" + result);
```

示例

- 画圆 Circle99Frame.java

Goto语句及其弊端

- 有关Goto语句的争论
- Java中的解决方式
  - 在循环中：break 标号， continue 标号
  - 其中，在循环前面可以用标号来表明是哪重循环

特殊流程控制语句

- break 语句
  - break语句用于终止某个语句块的执行
  - { .....
  - break;
  - ......
  - }
  - break语句出现在多层嵌套的语句块中时，可以通过标签指明要终止的是哪一层语句块
  - label1: { .....
  - label2: { .....
  - label3: { .....
  - break label2;
  - ......
  - }
  - }
  - }

特殊流程控制语句

- break 语句用法举例
- public class TestBreak{
  - public static void main(String args[]){
    - for(int i = 0; i<10; i++){
    - if(i==3)
    - break;
    - System.out.println(" i =" + i);
    - }
    - System.out.println("Game Over!");
  - }
  - }

特殊流程控制语句

- continue 语句
  - continue语句用于跳过某个循环语句块的一次执行
  - continue语句出现在多层嵌套的循环语句体中时，可以通过标签指明要跳过的是哪一层循环

- continue语句用法举例1

```
public class ContinueTest {
        public static void main(String args[]){
    for (int i = 0; i < 100; i++) {
        if (i%10==0)
            continue;
        System.out.println(i);
        }
    }
}
```

特殊流程控制语句

- 使用break或continue 标号
- 示例： 求100以内的质数
- Prime100Continue.java

3.4 数组

数组

数组概述

- 数组是多个相同类型数据的组合
- 一维数组的声明方式：
- int[] a;
- double []b
- Mydate []c;
- 注意方括号写到变量名的前面，也可以写到后面
- 问题：有意义吗

数组初始化

- 数组定义 与 为数组元素分配空间 分开进行

```
int  []a   = new int[3];
a[0] = 3;
a[1] = 9;
a[2] = 8;

MyDate []dates = new MyDate[3];
dates[0] = new MyDate(22, 7, 1964);
dates[1] = new MyDate(1, 1, 2000);
dates[2] = new MyDate(22, 12, 1964);
```

一维数组声明

- Java语言中声明数组时不能指定其长度(数组中元素的个数)，例如：
- int a[5]; //非法
- 数组是引用类型
  - int [ ] a = new int[5];
  - 这里 a 只是一个引用

数组初始化

- 静态初始化：
  - 在定义数组的同时就为数组元素分配空间并赋值。

```
int[] a = { 3, 9, 8};
或写为  int[] a = new int[]{ 3, 9,  8 };

MyDate[]  dates= {
        new MyDate(22, 7, 1964),
        new MyDate(1, 1, 2000),
        new MyDate(22, 12, 1964)
};
```

注：最后可以多一个逗号。如{3,9,8,}

数组元素的默认初始化

- 数组是引用类型，
- 数组一经分配空间，其中的每个元素也被按照成员变量同样的方式被隐式初始化。例如：
  - (数值类型是0， 引用类型是null)
    - int []a= new int[5];
    - //a[3]则是0

数组元素的引用

- 数组元素的引用方式
  - index为数组元素下标，可以是整型常量或整型表达式。如a[3]，b[i]，c[6*i];
  - 数组元素下标从0开始；长度为n的数组合法下标取值范围： 0 ~ n-1；
- 每个数组都有一个属性length指明它的长度，例如：a.length 指明数组a的长度(元素个数)；
  - int[] ages = new int[10];
  - for (int i=0; i<ages.length; i++)
  - {

```
System.out.println( ages[i] );
```

  - }

增强的for语句

- Enhanced for语句可以方便地处理数组、集合中各元素
- 如：
  - int[] ages = new int[10];
  - for (int age : ages)
  - {
    System.out.println(age);
  - }
  - 这种语句是只读式的遍历

复制数组

- System.arraycopy方法提供了数组元素复制功能：

```
//源数组
int[] source = { 1, 2, 3, 4, 5, 6 };
// 目的数组
int []dest = { 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 };
// 复制源数组中从下标0开始的source.length个元素到
// 目的数组，从下标0的位置开始存储。
System.arraycopy( source, 0, dest, 0, source.Length );
```

多维数组

二维数组举例：

```
int [][] a =
{{1,2},{3,4,0,9},{5,6,7}};
```

j / i	j = 0	j = 1	j = 2	j = 3
i = 0	1	2		
i = 1	3	4	0	9
i = 2	5	6	7	

- 二维数组是数组的数组

```
int [][] t = new int [3][];
t[0] = new int[2];
t[1] = new int[4];
t[2] = new int[3];
```

多维数组的声明和初始化应按从高维到低维的顺序进行

```
int t1[][] = new int [][4];  //非法，这与C++不同
```

示例

- 36选7
- Rnd_36_7.java

- 后面的例子视频中没有讲到

- 请注意一下

示例

- 筛法求素数
- 要点：使用boolean数组

示例

- 排块游戏
- 要点
  - 按钮的数组
  - 按钮的生成、加入、事件
  - 按钮的Tag
  - 函数的书写
  - 注释的书写