面向对象程序设计Java

江春华

电子科技大学计算机学院

内　容

第3章　流程控制与数组

1. 分支语句
2. 循环语句
3. 数组

Java语言中的流程控制语句提供了控制程序执行顺序的手段。流程控制是程序代码的重要部分。

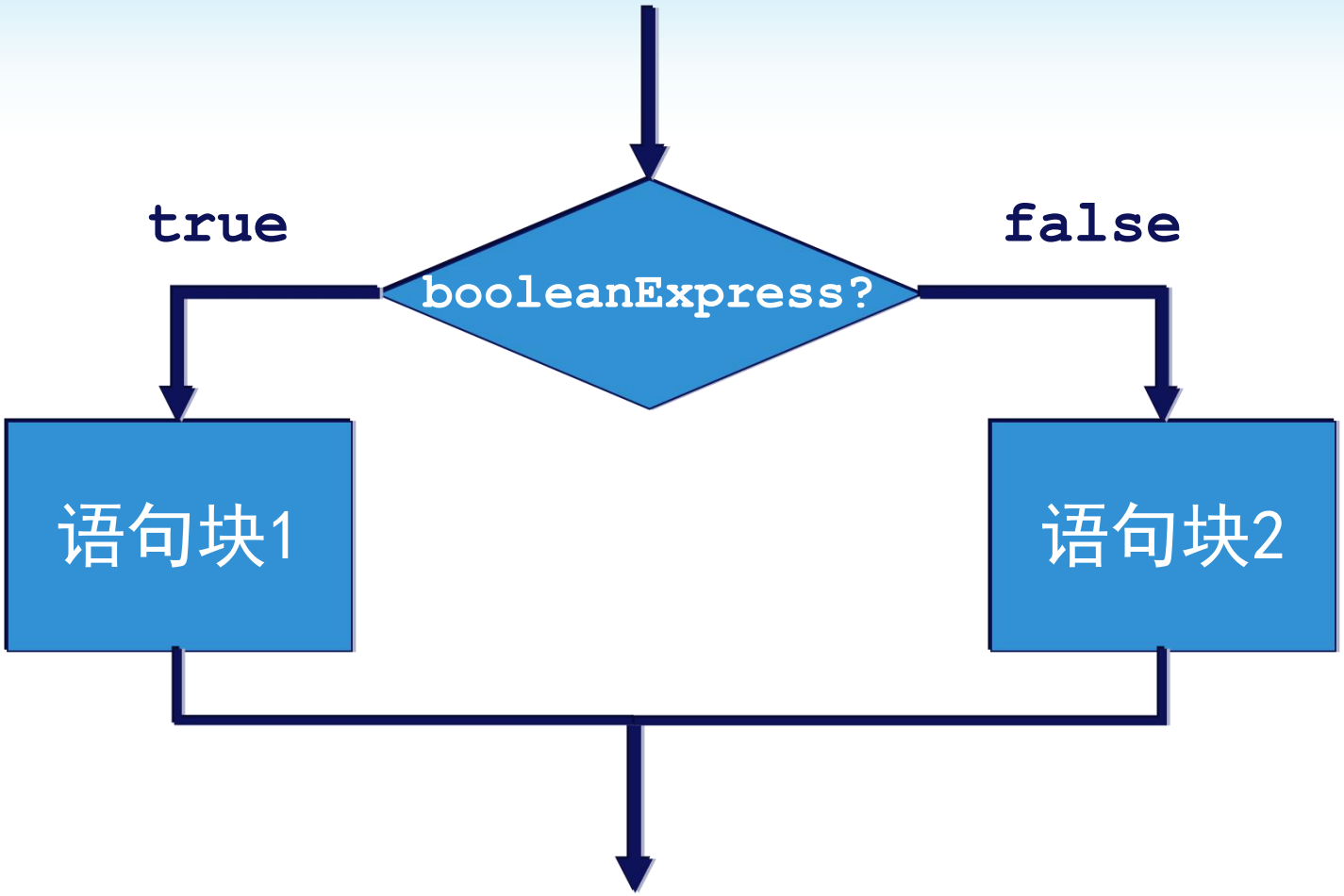
程序执行的顺序称为程序流。程序流即可以是语句自然的顺序，也可以是由控制语句执行跳转的逻辑顺序。流程控制就是根据一定条件判断而选择程序执行顺序序列。程序流由语句或语句块组成。

流控制语句分类

流控制语句分为：分支语句、循环语句、异常处理语句和跳转语句。

分支与跳转语句	if-else	break	switch	return
循环语句	while	do-while	for	continue
异常处理语句	try	catch	finally	throw

分支语句：if-else



分支语句：if-else

if语句

```
if(b1){
   block;
}
```

当b1为true时，执行block。

if-else语句

```
if(b1){
    block1;
}else{
    block2;
}
```

当b1为true时，执行block1，为false时，执行block2。

嵌套if语句

```
if(b1){
    block1;
}else if(b2){
    block2;
}else{
    block3;
}
```

当b1为true时，执行block1，b1为false且b2为true时，执行block2，b1和b2都为false时，执行block3

UESTC

2. 分支语句：switch

分支语句：switch

多分支语句switch与if-else语句一样，是根据相关表达式的值选择程序流程。它与if语句不同处在于多种情况可供程序流程选择。

switch所用的表达式为int类型相容的数据表达式，它可以是byte、short、char或者int类型的值，特别要指出的是不能是布尔型的值。

分支语句：switch

格式:

```
switch (intexpression){
    case int1:
                statement or block (1)
                break;
    case int2:
                statement or block (2)
                break;
    ...
    default:
                statement or block(d)
}
```

break语句

break常用于switch语句的中，用break语句起跳出switch语句的作用。

break语句不仅能用在switch语句，也可以用在循环语句，都同样起到结束它所在语句块流程。

处在break语句之后的语句将会被跳过而不被执行。

break语句

break语句既可以用于中断或跳出它所在语句块，还可以用于中断或跳出它所指定的块。

如果break语句中有语句标记，该语句就会中断或跳出这个语句标记所指定的块。

语句标记是指在语句块的第一条语句前面加上一个标记，作为该语句的标记。语句标记是一个Java的合法标识符，在使用时需要在它的后面加上一个“：”。

break语句

语句标记形式如下：

```
blockLabel:{
                codeBlock;
            }
```

带语句标记的break语句：

```
    break   blockLabel;
```

循环语句

Java程序中循环语句有三种：**while**语句、**do-while**语句和**for**语句。循环语句实现了一定条件下的代码重用。

循环语句的程序流程从循环的开始到循环的结束都依赖于循环的控制条件判断。在每次循环时循环控制的布尔表达式值为true时，循环得以继续，布尔表达式值为false时循环结束。

循环语句

循环是由四个部分组成，根据不同的循环语句，它们之间执行顺序有所不同，这四个组成部分是：

初始化(initalization)	为循环设置初始量
判断(condition)	布尔表达式的值决定循环是否继续
循环体(body)	循环的代码段(语句)
迭代(iteration)	每循环一次，改变循环控制变量的值

循环语句：while

while循环语句的格式是：

```
[initalization]
while (expressBool){
    statements;
    [iteration;]
}
```

循环语句：while

while语句循环执行的顺序是：

1.执行初始化initalization(如果有)；

2.计算表达式expressBool的值；

3.若expressBool值为true，则执行循环体statements；

4.执行迭代部分iteration(如果有)；

5.返回到2；

6.若expressBool值为false，则终止while循环。

循环语句：do-while

do-while循环语句的格式是：

```
[initalization]
do {
   statements;
   [iteration;]
}while (expressBool);
```

循环语句：do-while

do-while语句循环执行的顺序是：

1.执行初始化initalization(如果有)；

2.执行循环体statements;

3.执行迭代部分iteration(如果有)；

4.计算表达式expressBool的值;

5.若expressBool值为true，则返回到2;

6.若expressBool值为false，则终止do-while循环。

循环语句：for

for循环语句的格式是：

```
for(initalize; condit; iterat){
        statements;
}
```

循环语句：for

for语句循环执行的顺序是：

1.执行初始化initalize;

2.计算表达式condit的值;

3.若condit值为true，则执行循环体statements;

- 执行迭代部分iterat;
- 返回到2;

4.若condit值为false，则终止for语句。

循环语句示例

while语句实现输出0℃ ~30℃中以5℃为间隔的摄氏到华氏温度转换表。

```
class TempConverWh
  public static vo                tri
    int fahr,cels
    System.out.println("Cel\tFahr\n"
    cels=0;
    while (cels<=30) {
      fahr=cels*9/5+32;
      System.out.println(cels+"\t"+fahr);
      cels += 5;
    }
  }
}
```

初始化

判断条件

循环体

迭代

循环中的continue语句

continue语句用于循环结构中，当程序执行contiune语句时，程序流程就结束本次循环，充当了循环体的最后一条语句作用。

continue语句也可以带语句标记，它的作用是结束该语句标记的外层循环的本次循环。

continue语句使用格式如下：

```
continue  [outerLabel];
```

数组

在Java中，数组是引用类型。数组类型是一种有序数据的集合，数组中在每一维上的元素具有相同的数据类型。

数组通过数组名和它的下标对数组元素访问，数组元素的下标不能越界。

数组是一个对象，数组声明不能创建对象本身，而创建一个引用。数组元素由new语句或数组初始化软件动态分配。

数组声明

Java的数组声明采用与C语言类似的形式。数组可分为一维数组和多维数组。它们的声明的形式为：

```
type   arrayName[][[]...];
```

或另一等价形式：

```
type[][[]...]   arrayName;
```

数组实例化

在Java语言中，数组的声明是不能确定数组大小。数组的实例化即存储单元的分配是由new运算符实现。

数组通过数组名和它的下标对数组元素访问，数组元素的下标不能越界。

数组实例化示例：

```
int[] a = new int[3];
```

数组a有元素：a[0]、a[1]、a[2]。

数组实例化

数组在实例时，同时也有了初始化的值。

例：`int[] a = new int[3];`

数组a的三个元素有值都为0。

数组在创建时，也可显式初始化。

例：`int[] a = {1,2,3};`

数组a的三个元素的值分别为1，2，3。

数组实例化后就有了确定的元素，每个数组有一个属性length，其值就是这个数组的元素的数量。例：a.length的值为3。

多维数组

Java编程语言没有提供多维数组。它是通过创建数组的数组(和数组的数组的数组)。

数组通过数组名和它的下标对数组元素访问，数组元素的下标不能越界。

数组是一个对象，数组声明不能创建对象本身，而创建一个引用。数组元素由new语句或数组初始化软件动态分配。

多维数组实例化

虽然在声明数组的格式中，允许方括号在数组名的左边或者右边，但这种方式不适合数组句法的其它部分。

必须首先将低位维初始化，再能对它后面的各位依次初始化。

利用对每维元素的分步初始化，可以创建非矩形数组的数组。

多维数组使用示例

```
int[][] tDim = new int [3][];
```

矩形数组：

```
tDim[0] = new int[2];
tDim[1] = new int[2];
```

非矩形数组：

```
tDim[0] = new int[3];
tDim[1] = new int[6];
tDim[2] = new int[9];
```

```
tDim.length=3
```

矩形数组：tDim[0]和tDim[1]的length都为2

非矩形数组：tDim[0]、tDim[1]和tDim[2]的length分别为3、6、9

字符串

字符串是一串字符组成的数据，并用""包括起来。字符串常量是String类型的对象。

类String是Java语言的基础数据类型，它具有一定的特殊性。

Java编译器在对字符串数据与其它类型数据使用“+”运算符连接操作编译时，总是首先将其它类型数据转换为字符串类型，然后再进行字符串连接。

例：　"Age: "+18 ==> "Age: 18"

字符串

字符串常量对方法的访问示例：

```
"Hello".toUpperCase() ==> "HELLO"
"Hello".length() ==> 5
```

字符串常量不是char类型一维数组，不存在'\0'结束符。字符串和char数组可以通过相应方法转换：

```
char[] data = "Car".toCharArray();
```

则：data = {'C','a','r'}

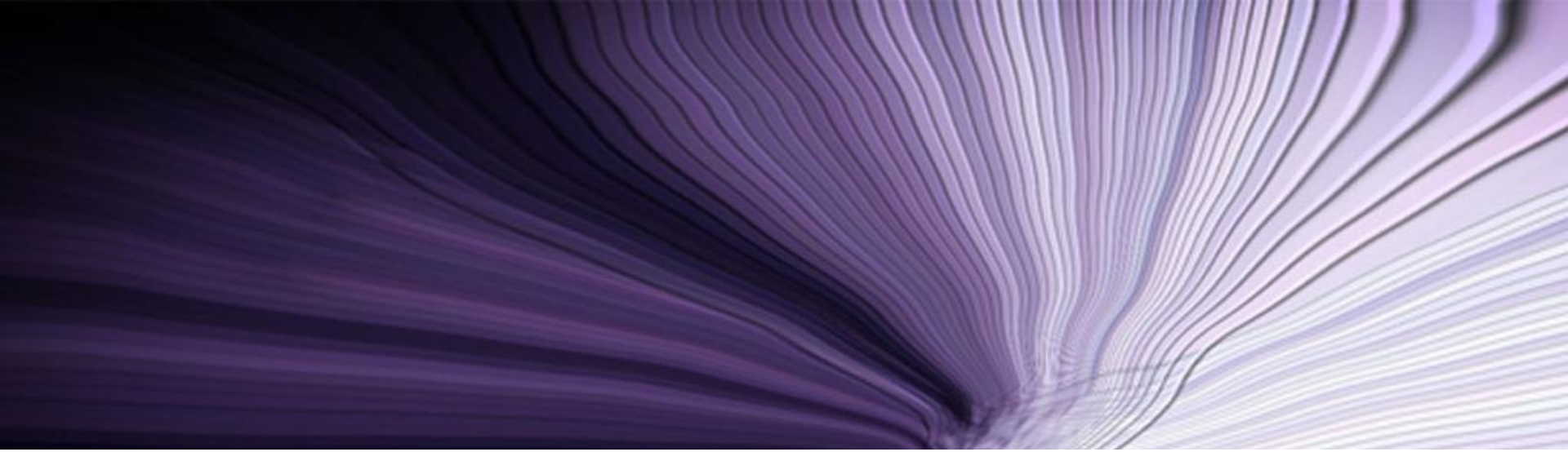
```
copyValueOf(data) ==> "Car"
```

思考问题

1. Java是如何处理异常，异常类层次是怎样的？
2. 异常的抛出有哪两种方式？
3. 如何在异常中传递信息？

作业：

本章习题： 1 ~ 8题



Q&A

电子科技大学计算机学院