JAVA程序设计


```
public class Cabbage {
    public static void main(String[] args) {
        final int sumCabbage = 20;
        int whiteRabbit = (int)(Math.random()*sumCabbage/2);
        int grayRabbit = (int)(Math.random()*sumCabbage/2);
        boolean finishJob = false;
        int digCabbage = sumCabbage;
        for(int i = 0; i<whiteRabbit;i++){
            System.out.println("小白兔拾取了第"+digCabbage+"颗白菜。");
            digCabbage--;
        }
        for(int i = 0; i<grayRabbit;i++){
            System.out.println("小灰兔拾取了第"+digCabbage+"颗白菜。");
            digCabbage--;
        }
        while(digCabbage>0) {
            System.out.println("老山羊拾取了第"+digCabbage+"颗白菜。");
            digCabbage--;
        }
        if(grayRabbit>sumCabbage/5)
            System.out.println("老山羊给了小灰兔一车白菜。");
        else
            System.out.println("小灰兔偷懒得不到白菜。");
        if(whiteRabbit>sumCabbage/5) {
            System.out.println("老山羊要给小白兔一车白菜。");
            finishJob = true;
        }
        else
            System.out.println("小白兔偷懒得不到白菜。");
        if(finishJob)
            System.out.println("小白兔向老山羊要了一包种子。");
    }
```

本章目录

本节目录

2.1 标识符、关键字和数据类型

1. 标识符和字符集

•Java对包、类、方法、参数和变量等要素命名时使用的字符串序列。

•Java语言的**标识符**是以字母、下划线或$符号开头的后面含有字母、下划线、$符号和数字的字符串。

```
public class Cabbage {
        public static void main(String[] args) {
                final int sumCabbage = 20;
                int whiteRabbit = (int)(Math.random()*sumCabbage/2);
                int grayRabbit = (int)(Math.random()*sumCabbage/2);
                boolean finishJob = false;
                int digCabbage = sumCabbage;
```

2.1 标识符、关键字和数据类型

1. 标识符和字符集

•Java语言的**标识符**是以字母、下划线或$符号开头的后面含有字母、下划线、$符号和数字的字符串。

Welcome _Money &1000000　张三，李四，Joan，John，汽车，飞机

•标示符首位不能是数字。

•标识符的长度没有限制，但Java系统最多可以识别前255个字符。

•Java标识符中的字母是**大小写相关**的。

•Java语言采用**Unicode**字符集。

•Unicode字符是**16bits**存储格式，可以是包括中文在内的多国文字。

- Java标识符使用惯例

类和接口——SimpleApp

类名和接口名通常用名词，且每个单词的首字母大写；

方法——processResult

方法名用动词开头的单词序列，首单词全部小写，后面的每个单词首字母大写；

常量——PI

常量名全部用大写字母；

变量——outputResult

所有的对象实例名和全局变量名都使用首单词全部小写，后面的每个单词首字母大写的格式；

良好的Java代码规范

- 尽可能多的加入注释，开始时可以写中文注释，最后完成时转为英文注释；
- 注意程序中的异常，捕获异常处理后打印名称，System.out.println(异常名称)；
- 定期备份工作，开始新的工作时，将上次的代码打包.rar备份，注明日期。

2. 关键字

- 关键字是由系统定义的一些字符串，代表语言中的特定含义。Java语言共规定了48个关键字，Java语言关键字都是小写的。
- 在Java中保留但已经不再使用的2个关键字：const，goto
- 在标识符中可以包含关键字，但是关键字不能是标识符。

abstract	boolean	break	byte	case	catch
char	class	continue	default	do	double
else	extends	false	final	finally	float
for	if	implements	import	instanceof	int
interface	long	native	new	null	package
private	protected	public	return	short	static
super	switch	synchronized	this	throw	throws
transient	true	try	void	volatile	while

3.常量

- 常量是在程序运行中其值保持不变的量。
- Java语言中允许用户使用两种常量：
  - ◆文字常量（Literal Constant）　3.14159
  - ◆符号常量（Symbolic Constant）PI
- Java语言的所有基本数据类型都可以定义常量，其取值范围内的值都可以被表示成文字常量。
- 用“final”修饰的Java语言标识符为符号常量，其值在赋值之后将不能再作改动。

```
final float PI=3.14159;
```

4. 变量

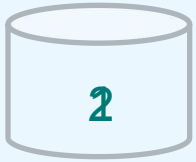
- 可以改变的量。
- 任何变量在使用之前必须经过**声明、创建和初始化**，否则将无法完成任何操作。
  - ◆变量的声明是要把代表变量的标识符作出说明
  - ◆变量的创建是为其分配存储空间
  - ◆变量的初始化：
    - ◆**方法体外声明的变量，系统可自动赋初值；**
    - ◆**方法体内声明的变量，需由语句赋初值。**

JAVA是一门强类型语言

4. 变量

•**变量**表示程序中数据可变的量，是内存中的一小块区域，使用变量名来访问这块区域。

```
int i = 1;
i = 2;
```



•任何变量在使用之前必须经过**声明、创建和初始化**，否则将无法完成任何操作。

```
double j = 2;
char c ;
```

5.数据类型

- 数据
  - 基本数据
    - 数值型
      - 整数类型（byte，short，int，long）
      - 浮点类型（float，double）
    - 字符类型（char）
    - 逻辑类型（boolean）
  - 引用数据
    - 类（class）
    - 接口（interface）
    - 数组

基本数据类型

Java语言共有4类8种基本数据类型。

（1）整数类型：byte、short、int和long

- 在Java语言中，共有4种整数类型的数据，分别用关键字**byte**、**short**、**int**和**long**声明，这4种整数类型的数据都是有符号数。每一种整数类型的数都可以用8进制、10进制或者16进制格式来表示。
- 整数类型的默认型为**int**型。

19　　28　　-36

数据类型	数据长度	取值范围
byte	8位	-2^{7}~$2^{7}-1$
short	16位	-2^{15}~$2^{15}-1$
int	32位	-2^{31}~$2^{31}-1$
long	64位	-2^{63}~$2^{63}-1$

（2）浮点类型：float和double

- 在Java语言中，共有2种浮点类型的数据，分别用关键字**float**和**double**声明，其数据长度分别为**32**位和**64**位。
- 浮点类型的默认型为**double**型。

数据类型	数据长度
float	**32位**
double	**64位**

-1.44144　　3.1415926

（3）字符类型：char

- 单个字符被定义为char类型，字符型数据必须用**单引号**括起来。Java语言使用**Unicode**字符，使用**16位无符号整数**来表示一个字符，其取值范围是0~65535 。 示例：‘a’ ‘z’ ‘周'

- Java语言把**字符串**定义为一个类——**String类**，它不包括在8种基本数据类型当中，字符串数据必须用**双引号**括起来，如“ this is a simple program! ”。

- “

Java语言支持**转义字符**

转义字符表

\b	退格	backspace	\u0008
\t	水平制表	tab	\u0009
\n	换行	linefeed	\u000a
\r	回车	carriage return	\u000d
\”	双引号	double quote	\u0022
\’	单引号	single quote	\u0027
\\	反斜线	backslash	\u005c

[例2.1] 打印一些常用的转义字符。
<SimpleApp1 .java>

```
public class SimpleApp1 {
    public static void main(String[] args) {
        System.out.print("转义字符打印");
        System.out.print("\r\n");
        System.out.print("\"");
        System.out.print("\t");
        System.out.print("\'");
        System.out.println();
        System.out.print("\\");
    }
}
```

输出结果：
转义字符打印
"	'
\

（4）逻辑类型：boolean

两种取值：“true”和“false”。

注意：在Java语言中，逻辑类型与整数类型不能进行直接转换。这与C和C++语言有明显的不同。

4. 变量

- 可以改变的量。
- 任何变量在使用之前必须经过**声明、创建和初始化**，否则将无法完成任何操作。
  - ◆变量的声明是要把代表变量的标识符作出说明
  - ◆变量的创建是为其分配存储空间
  - ◆变量的初始化：
    - ◆**方法体外声明的变量，系统可自动赋初值；**
    - ◆**方法体内声明的变量，需由语句赋初值。**

JAVA是一门强类型语言

成员变量 --- 系统赋默认值

局部变量 --- 需要显示赋值

基本数据类型变量默认值

数据类型	初始值	数据类型	初始值
boolean	**false**	**long**	**0L**
char	**‘\u0000’**	**float**	**0.0f**
byte	**0**	**double**	**0.0d**
short	**0**	各种引用类型	**null**
int	**0**		

基本数据类型的封装类型

（1）Integer类

1、Integer类的继承关系：

- Java.lang.Object → java.lang.Number → java.lang.Integer

2、常用成员属性：

- MAX_VALUE： $2^{31}-1$
- MIN_VALUE ： -2^{31}

3、构造方法：

- public Integer(int value)
- public Integer(String s)

（1）Integer类

4. 常用的成员方法

- public int compareTo(Integer anotherInteger)
- public static int parseInt(String s)
  throws NumberFormatException
- public static int parseInt(String s, int radix)
  throws NumberFormatException
- public static String toBinaryString(int i)
- public static String toString(int i)
- public boolean equals(Object obj)
- public static Integer valueof(int i)
- public int intValue()

（2）Float类

1、Float类的继承关系：

- Java.lang.Object → java.lang.Number → java.lang.Float

2、Float类的常用成员属性：

- MAX_VALUE：$(2-2^{-23})\cdot 2^{127}$
- MIN_VALUE ： 2^{-149}
- POSITIVE_INFINITY
- NEGATIVE_INFINITY
- SIZE

3、Float类的构造方法：

- public Float(float value)
- public Float(double value)
- public Float(String s)

（2）Float类

4. Float类的常用的成员方法

- public static float parseFloat(String s)
  throws NumberFromatException
- public String toString()
- public String toString(float f)
- public int compareTo(Float anotherFloat)
- public short shortValue()
- public int intValue()

（3）Character类

1、Character类的继承关系：

- Java.lang.Object → java.lang.Character

2、常用成员属性：

- MAX_VALUE：’\uFFFF’
- MIN_VALUE：’\u0000’

3、构造方法：

- public Character(char value)

4、常用的成员方法

- public static boolean isLetter(char ch)
- public static boolean isDigit(char ch)
- public static char toLowerCase(char ch)
- public static boolean isWhitespace(char ch)
- public static char toUpperCase(char ch)

（4）Boolean

1、Boolean类的继承关系：

- Java.lang.Object → java.lang.Boolean

2、常用成员属性：

- TRUE
- FALSE

3、构造方法：

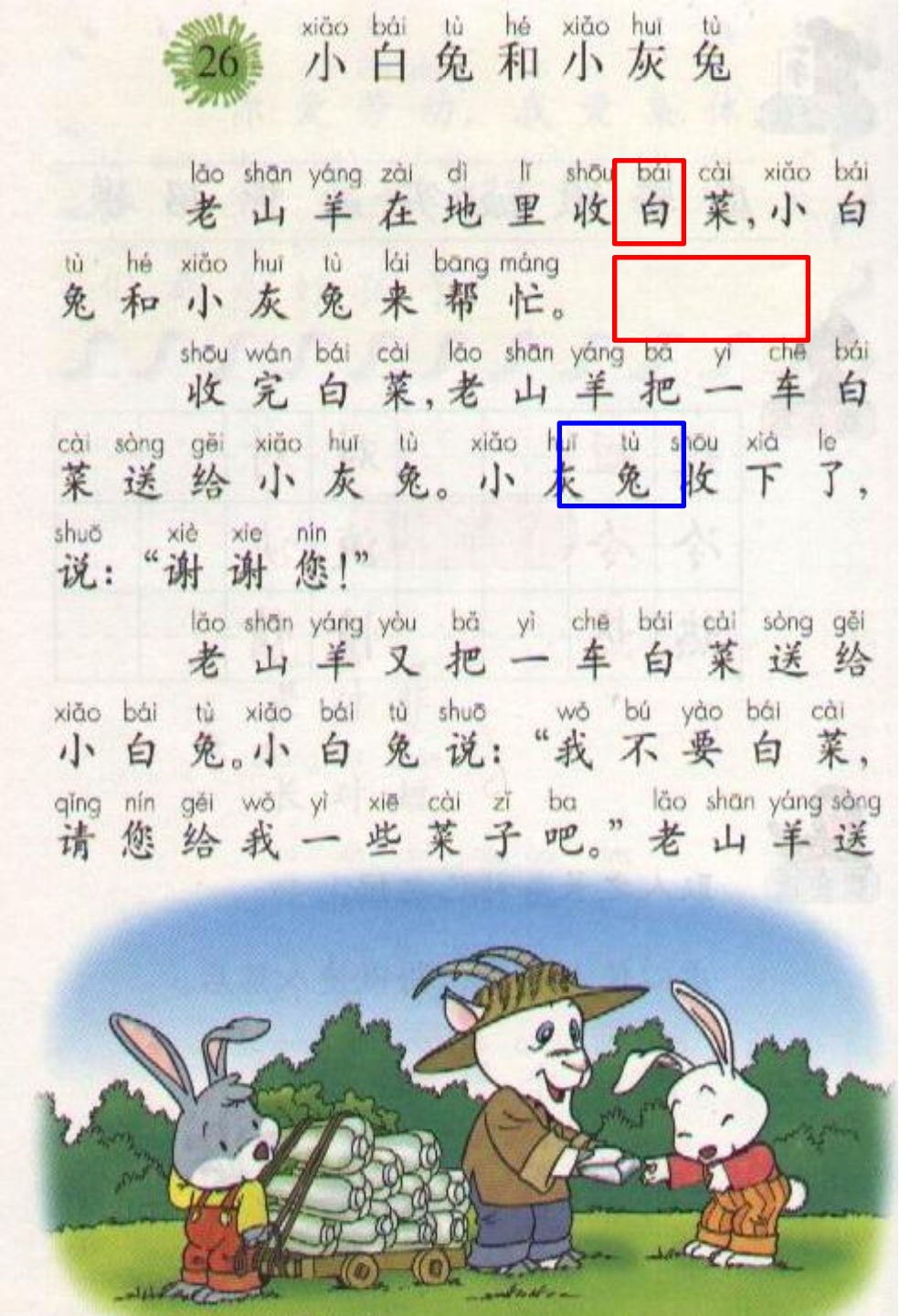
- public Boolean(boolean value)
- publie Boolean(String s)

4、常用的成员方法

- public static boolean parseBoolean(String s)
- public static String toString(boolean b)

其他引用数据类型

除了基本数据类型的封装类型之外，Java语言中还允许定义其他引用数据类型，这其中包括类、数组和接口类型，将在后面的相应章节中分别介绍。



26 小白兔和小灰兔

老山羊在地里收白菜，小白兔和小灰兔来帮忙。

收完白菜，老山羊把一车白菜送给小灰兔。小灰兔收下了，说："谢谢您！"

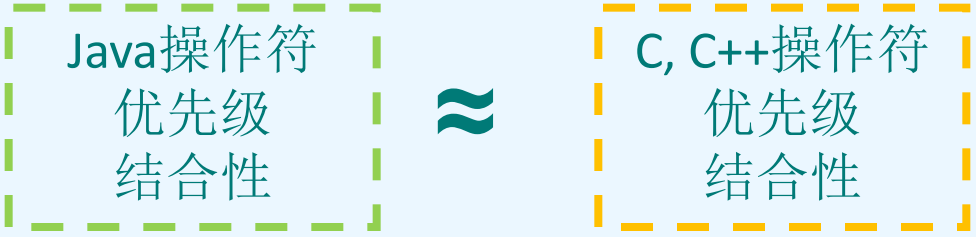
老山羊又把一车白菜送给小白兔。小白兔说："我不要白菜，请您给我一些菜子吧。"老山羊送

```
public class Cabbage {
    public static void main(String[] args) {
        final int sumCabbage = 20;
        int whiteRabbit = (int)(Math.random()*sumCabbage/2);
        int grayRabbit = (int)(Math.random()*sumCabbage/2);
        boolean finishJob = false;
        int digCabbage = sumCabbage;
        for(int i = 0; i<whiteRabbit;i++){
            System.out.println("小白兔拾取了第"+digCabbage+"颗白菜。");
            digCabbage--;
        }
        for(int i = 0; i<grayRabbit;i++){
            System.out.println("小灰兔拾取了第"+digCabbage+"颗白菜。");
            digCabbage--;
        }
        while(digCabbage>0) {
            System.out.println("老山羊拾取了第"+digCabbage+"颗白菜。");
            digCabbage--;
        }
        if(grayRabbit>sumCabbage/5)
            System.out.println("老山羊给了小灰兔一车白菜。");
        else
            System.out.println("小灰兔偷懒得不到白菜。");
        if(whiteRabbit>sumCabbage/5) {
            System.out.println("老山羊要给小白兔一车白菜。");
            finishJob = true;
        }
        else
            System.out.println("小白兔偷懒得不到白菜。");
        if(finishJob)
            System.out.println("小白兔向老山羊要了一包种子。");
    }
```

本节目录

2.2 操作符

•Java语言的操作符基本上继承了C和C++的操作符体系，从形式到功能，包括优先级和结合性与C和C++的操作符非常相似。



•**Java语言取消了指针操作符“*”和“&”，结构体成员操作符“->”，长度操作符“sizeof”。**

2.2 操作符

- 赋值操作符 =
- 数学操作符 +, -, * , /, %
- 关系操作符 <, >, <=, >=, = =, != (equals)
- 逻辑操作符 &&, ||, !
- 位操作符 &, |, ^, ~
- 移位操作符 >>, <<, >>>
- 字符串操作符 + +=
- 类型转换操作符
- 三元条件操作符 a?b:c

1. 赋值操作符

- 在Java中赋值操作是用‘=’进行的。

```
int a = 1;
int b = a+2;
```

- “取得右边的值，把它复制到左边”。

```
1 = a; \\不允许的
```

[例2.2] 引用类型赋值在堆和栈的区别。

<SimpleApp2 .java>

```
class Number {int i;}
public class SimpleApp2 {
      public static void main(String[] args) {
        //类为引用类型
        Number n1 = new Number();
        Number n2 = new Number();
        //基本类型赋值
        int j = 9; int k = 47;
        //引用类型赋值
        n1.i = 9;n2.i = 47;
        //打印 j\k\n1\n2的值
        System.out.println("第一次赋值 n1.i: " + n1.i +
          ", n2.i: " + n2.i+", j: "+j+", k: "+k);
        //将n2赋给n1，k赋给j
        n1 = n2;  j = k;
        System.out.println("大二次赋值 n1.i: " + n1.i +
          ", n2.i: " + n2.i+", j: "+j+", k: "+k);
        //改变n1和j的值看n2和k的值是否变化
        n1.i = 27; j = 27;
        System.out.println("第三次赋值 n1.i: " + n1.i +
          ", n2.i: " + n2.i+", j: "+j+", k: "+k);
      }
    } ///:~
```

输出结果：

第一次赋值
n1.i: 9, n2.i: 47,
j: 9, k: 47
大二次赋值
n1.i: 47, n2.i: 47,
j: 47, k: 47
第三次赋值
n1.i: 27, n2.i: 27,
j: 27, k: 47

2. 数学操作符

- 数学操作符并不改变操作数的值，而是返回一个必须赋给变量的值。

运算符	实际操作	例子	功能
+	加运算	a+b	求a与b相加的和
-	减运算	a-b	求a与b相减的差
*	乘运算	a*b	求a与b相乘的积
/	除运算	a/b	求a除以b的商
%	取模运算	a%b	求a除以b的余数
++	自加1	a++	将a的值加1后再放回变量a
--	自减1	a--	将x的值减1后再放回变量x

```
a++; //a = a+1;
a--; //a = a-1;
```

2.数学操作符

数学操作符 + - * / % +=

- 一元操作符

操作符	实际操作	例　子	功　能
+	正值	+x	
-	负号	-x	对 x 取负
++	加 1	x++, ++x	将 x 的值加 1 后再放回变量 x
--	减 1	x--, --x	将 x 的值减 1 后再放回变量 x

- 自增、自减操作符

  i = 1； j=1

  a = i++ ,b = ++j

  结果 a=1,b=2;

[例2.3] 前递增（++i）和后递增（i++）的区别。

<SimpleApp3.java>

```
public class SimpleApp3 {
    public static void main(String[] args) {
        int i = 2;
        System.out.println("i初始化为2");
        //后递增
        System.out.println("做后递增运算后i++本身的值为 " + i++);
        System.out.println("做后递增运算后i的值为:  " + i);
        int j = 2;
        System.out.println("j初始化为2");
        //前递增
        System.out.println("做前递增运算后++j本身的值为 " + ++j);
        System.out.println("做前递增运算后j的值为:  " + j);

    }
} ///:~
```

输出结果:

```
i初始化为2
做后递增运算后i++本身的值为 2
做后递增运算后i的值为:  3
j初始化为2
做前递增运算后++j本身的值为 3
做前递增运算后j的值为:  3
```

2. 数学操作符

- 二元操作符

  算术操作符作为二元操作符，这种操作符并不改变操作数的值，而是返回一个必须赋给变量的值。二元算术操作符具有左结合性。

运 算 符	实际操作	例　　子	功　　能
+	加运算	a+b	求 a 与 b 相加的和
–	减运算	a–b	求 a 与 b 相减的差
*	乘运算	a*b	求 a 与 b 相乘的积
/	除运算	a/b	求 a 除以 b 的商
%	取模运算	a%b	求 a 除以 b 的余数

3. 关系操作符

== != >= <= > <

- Java语言中关系运算的结果是逻辑型，当关系成立时结果为true，否则为false。

```
int a = 1;
int b = 2;
a == b ;   //false
```

若想对比两个对象的实际内容是否相同，又该如何操作呢？此时，必须使用所有对象都适用的特殊方法equals()。但这个方法不适用于“基本类型”，那些类型直接使用==和!=即可

[例2.4] ==和equals()方法的区别。

<SimpleApp4.java>

```
public class SimpleApp4 {
    public static void main(String[] args) {
    int i = 1;int j = 1;
    System.out.println("基本类型数据i和j用==来比较是否相等结果为");
    //基本数据类型用 == 和 ！=就可用来比较两个数据是否相等
    System.out.println(i == j );
    //引用类型比较不能用 == 和 ！=
    Integer n1 = new Integer(47);
    Integer n2 = new Integer(47);
    //调用==方法来比较引用类型
    System.out.println("引用数据类型n1和n2用==来比较是否相等结果为");
    System.out.println(n1 == n2);
    //调用equals（）方法来比较引用类型
    System.out.println("引用数据类型n1和n2" +
            "用equals（）方法来比较是否相等结果为");
    System.out.println(n1.equals(n2));
    }
    }
```

输出结果：

基本类型数据i和j用==来比较是否相等结果为true

引用数据类型n1和n2用==来比较是否相等结果为false

引用数据类型n1和n2用equals（）方法来比较是否相等结果为true

4. 逻辑操作符

逻辑操作符：！ & | ^ && ||

在C和C++中，用整型数据替代逻辑型数据；而JAVA中整型数据与逻辑型数据不存在互换关系。

逻辑运算符

运 算 符	运 算	范 例	结 果
&	AND（与）	false&true	false
|	OR（或）	false|true	true
^	XOR（异或）	true^false	true
!	NOT（非）	!true	false
&&	AND（短路）	false&&true	false
||	OR（短路）	false||true	true

“&”和“&&”的区别在于，如果使用前者连接，那么无论任何情况，“&”两边的表达式都会参与计算。如果使用后者连接，当“&&”的左边为 false，则将不会计算其右边的表达式。

```
while(x = y) {
//...
}
```

程序的意图是测试是否“相等”（==），而不是进行赋值操作。在C和C++中，若y是一个非零值，那么这种赋值的结果肯定是true。这样使可能得到一个无限循环。在Java里，这个表达式的结果并不是布尔值，而编译器期望的是一个布尔值，而且不会从一个int数值中转换得来。所以在编译时，系统就会提示出现错误，有效地阻止我们进一步运行程序。所以这个缺点在Java里永远不会造成更严重的后果。唯一不会得到编译错误的时候是x和y都为布尔值。在这种情况下，x = y属于合法表达式。而在上述情况下，则可能是一个错误。

5.位操作符和移位操作符

●位操作符和位移操作符用来对二进制位进行运算，操作数应是整数类型，结果也是整数类型。下表给出了各操作符及其功能。

操作符	实际操作	例子	功　　能
~	按位取反	~a	对 a 按二进制每位取反
&	与运算	a&b	对 a 和 b 按二进制位每位进行与运算
|	或运算	a|h	对 a 和 b 按二进制位每位进行或运算
^	异或运算	a^b	对 a 和 b 按二进制位每位进行异或运算
<<	左移	a<<b	对 a 左移 b 位，低位用 0 填充
>>	有符号右移	a>>b	对 a 右移 b 位，高位用原高位重复
>>>	无符号右移	a>>>b	对 a 右移 b 位，高位用 0 填充

6.字符串操作符

●操作符“+”可以实现两个或多个字符串的连接，也可实现字符串与其他类对象的连接，在连接时，其他类对象会被转换成字符串。另外，操作符“+=”把两个字符串连接的结果放进第一个字符串里。

7. 类型转换操作符

●在一个表达式中可能有不同类型的数据进行混合运算，这是允许的，但在运算时，**Java将不同类型的数据转换成相同类型**，再进行运算。

●不同类型数据之间的转换规则分自动类型转换和强制类型转换。

- 自动类型转换。

类　型　1	类　型　2	转换后的类型
byte 或 short	int	int
byte 或 short 或 int	long	long
byte 或 short 或 int 或 long	float	float
byte 或 short 或 int 或 long 或 float	double	double
char	int	int

8.三元条件操作符　a?b:c

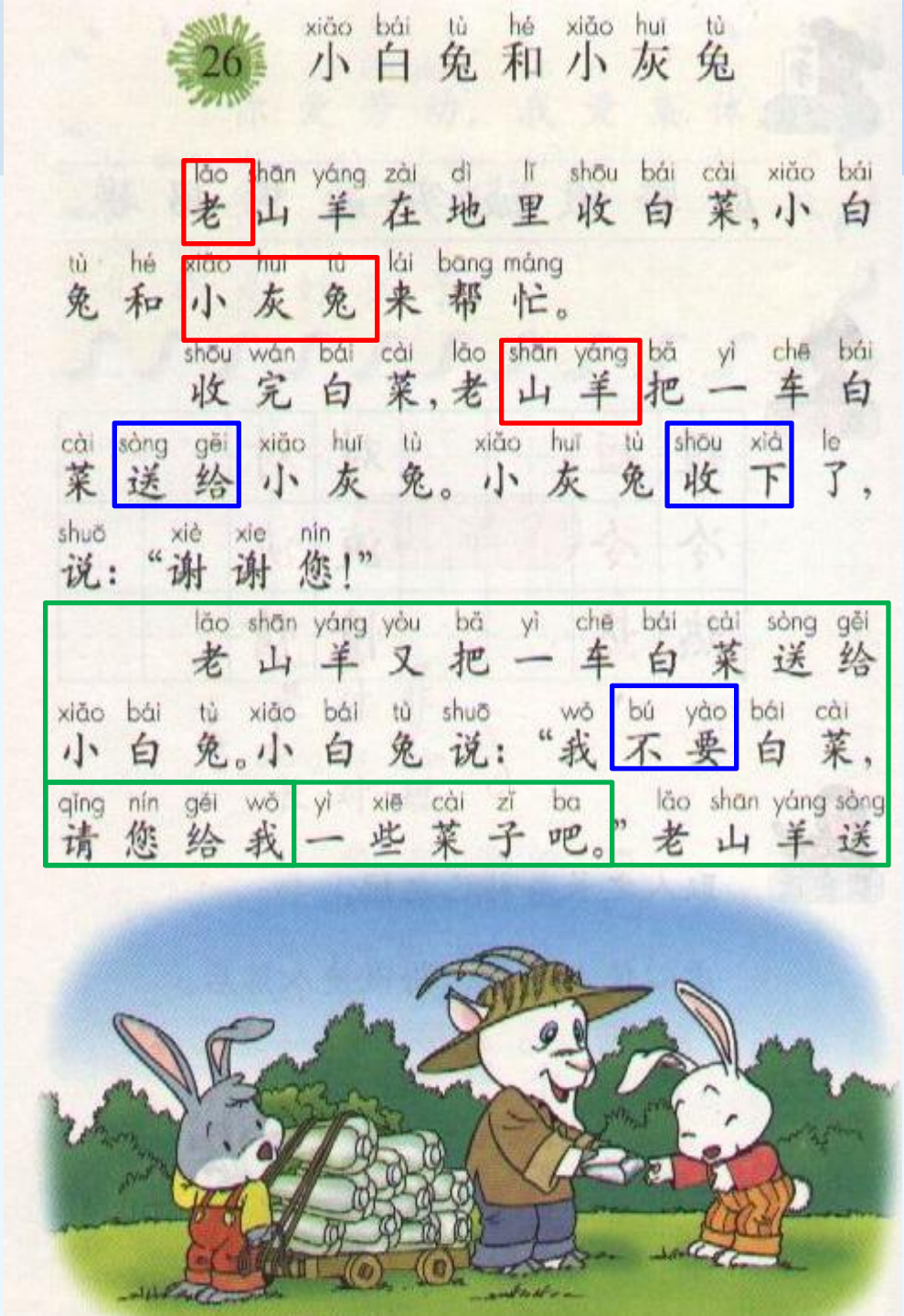
- 三元条件操作符的格式为：布尔表达式？值1: 值2（a?b:c）
- 若“布尔表达式”的结果为true，就计算“值1”；若　　“布尔表达式”的结果为false，计算的就是“值2”，而且“值1”“值2”成为最终由操作符产生的值。
- 如：(a>b)?a:b
  这个表达式将返回a和b中较大的那个数值。

9.Java语言操作符的优先级与结合性（一）

优先级	运算符	操作数类型	结合性
1	. [] ()		
2	++ -- - ~ ! (type)	算术 整型 逻辑 任意	右结合
3	* / %	算术	左结合
4	+ - +	算术 字符串	左结合
5	<< >> >>>	整型	左结合

9. Java语言操作符的优先级与结合性（二）

6	< <= > >= instanceof	算术 对象	左结合
7	== !=	基本类型	左结合
8	&	整型、逻辑	左结合
9	^	整型、逻辑	左结合
10	|	整型、逻辑	左结合
11	&&	逻辑	左结合
12	||	逻辑	左结合
13	?:	任意	右结合
14	= += -= *= /= %= <<= >>= >>>= &= |= ^=	任意	右结合



26 xiǎo bái tù hé xiǎo huī tù
小白兔和小灰兔

lǎo shān yáng zài dì lǐ shōu bái cài, xiǎo bái
老山羊在地里收白菜，小白
tù hé xiǎo huī tù lái bāng máng
兔和小灰兔来帮忙。

shōu wán bái cài, lǎo shān yáng bǎ yì chē bái
收完白菜，老山羊把一车白
cài sòng gěi xiǎo huī tù. xiǎo huī tù shōu xià le
菜送给小灰兔。小灰兔收下了，
shuō xiè xie nín
说：“谢谢您！”

lǎo shān yáng yòu bǎ yì chē bái cài sòng gěi
老山羊又把一车白菜送给
xiǎo bái tù. xiǎo bái tù shuō: “wǒ bú yào bái cài,
小白兔。小白兔说：“我不要白菜，
qǐng nín gěi wǒ yì xiē cài zi ba.” lǎo shān yáng sòng
请您给我一些菜子吧。”老山羊送

本节目录

2.3 表达式与语句

1. 表达式

•表达式是操作符、常量和变量的遵循语法规则的组合。

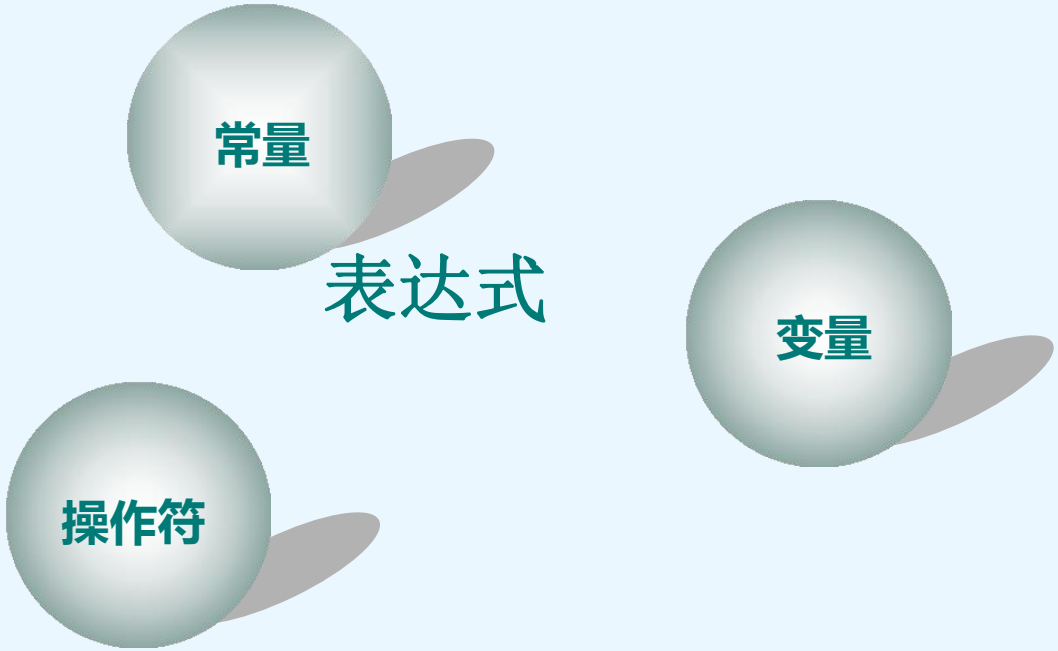
示例

a = 0

a+5.0

(a-b)*c-3

i<30&&i%10!=0



•Java语言的表达式既可以出现在选择条件测试、循环条件测试、变量说明、方法的调用参数等场合，也可以单独组成语句。

2. 语句和语句块

●Java中的每条语句均以分号“;”结束，可以是单行或多行代码。只有一个分号“;”的语句称为空语句。

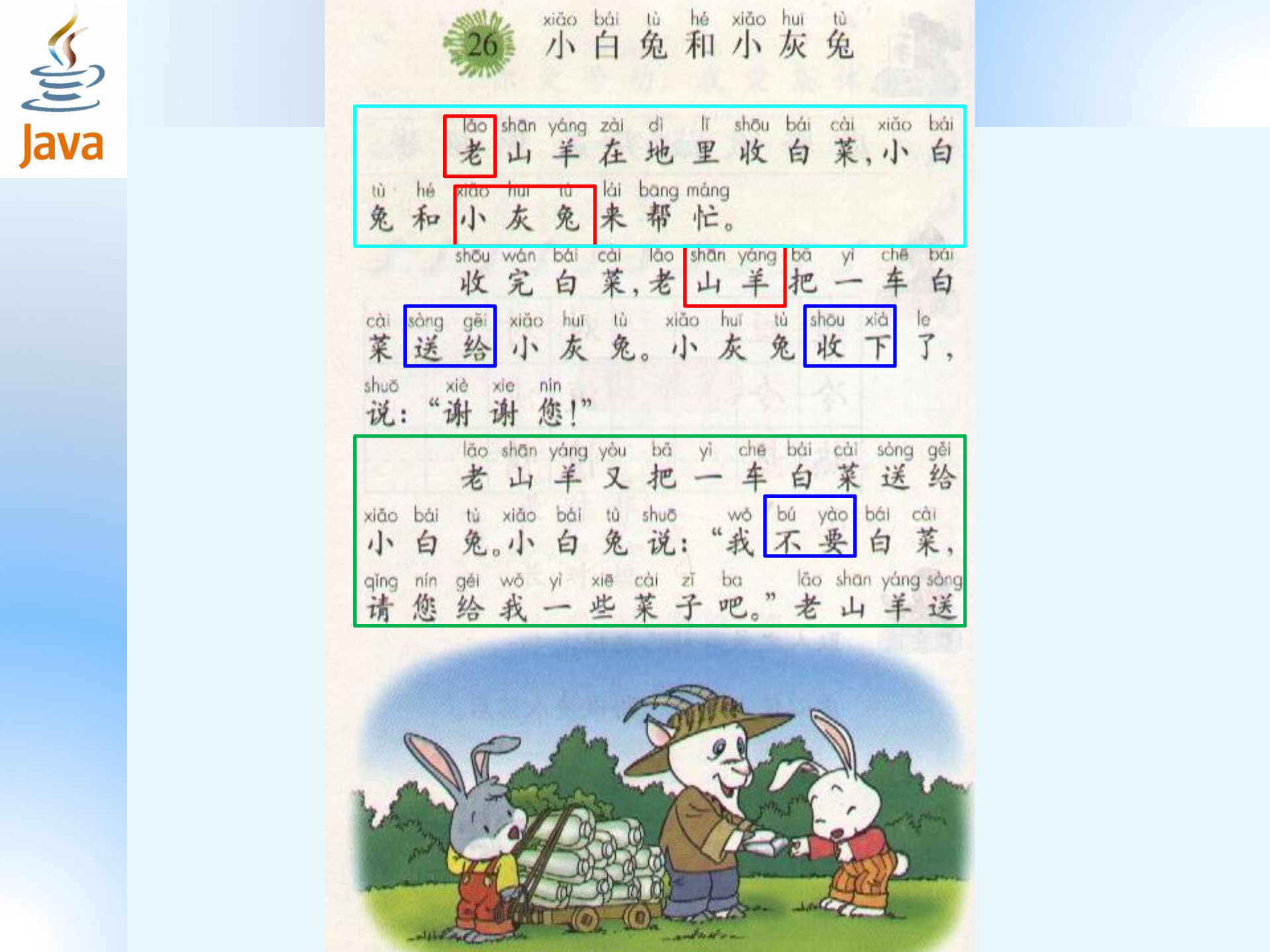
```
int a = 1;
double b = 1.2;
double c = a*b+1;
;
```

●语句块是以左大括号和右大括号为边界的语句集合。

```
{
    int a = i++;
    {int b = ++j;}
}
```

●Java语句含基本语句和控制流语句。

•**Java语言没有goto 语句。**


26
xiǎo bái tù hé xiǎo huī tù
小 白 兔 和 小 灰 兔
lǎo shān yáng zài dì lǐ shōu bái cài xiǎo bái
老 山 羊 在 地 里 收 白 菜，小 白
tù hé xiǎo huī tù lái bāng máng
兔 和 小 灰 兔 来 帮 忙。
shōu wán bái cài lǎo shān yáng bǎ yì chē bái
收 完 白 菜，老 山 羊 把 一 车 白
cài sòng gěi xiǎo huī tù xiǎo huī tù shōu xià le
菜 送 给 小 灰 兔。小 灰 兔 收 下 了，
shuō xiè xie nín
说："谢 谢 您！"
lǎo shān yáng yòu bǎ yì chē bái cài sòng gěi
老 山 羊 又 把 一 车 白 菜 送 给
xiǎo bái tù xiǎo bái tù shuō wǒ bú yào bái cài
小 白 兔。小 白 兔 说："我 不 要 白 菜，
qǐng nín gěi wǒ yì xiē cài zi ba lǎo shān yáng sòng
请 您 给 我 一 些 菜 子 吧。"老 山 羊 送

本节目录

2.4 控制流程

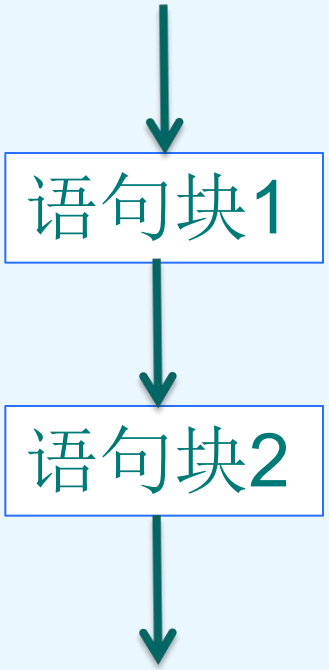
结构化程序设计中的三种控制结构

- 顺序结构
- 分支结构：

  IF_ELSE、SWITCH
- 循环结构：

  WHILE、DO_WHILE、FOR

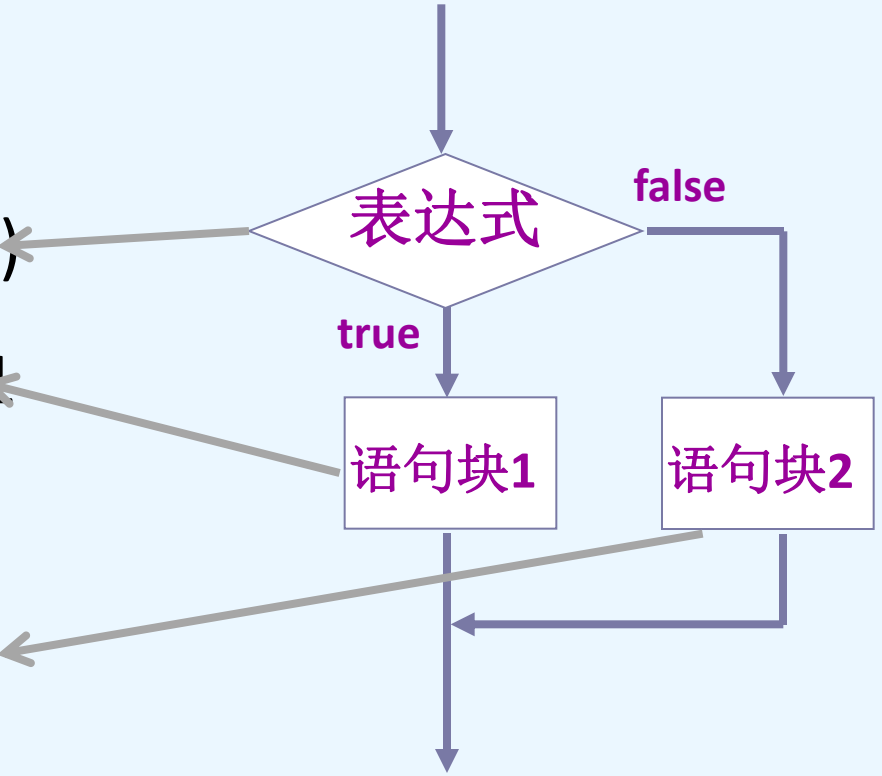
1. 顺序结构



2. 分支结构

(1) if-else语句

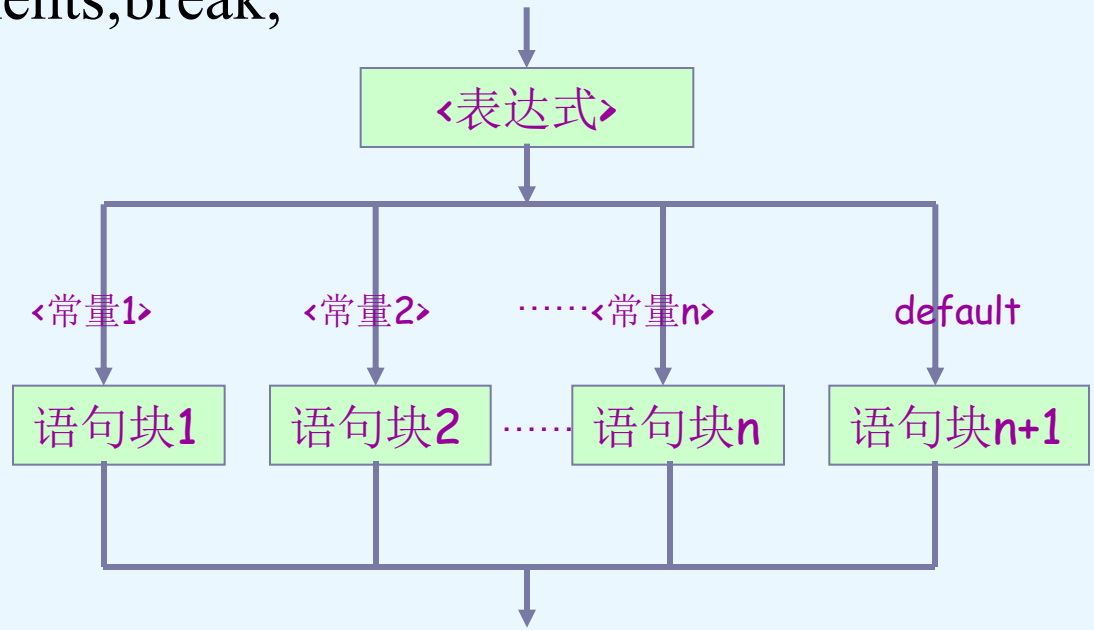
```
if ( boolean_expression )
    statement_or_block1
else
    statement_or_block2
```



2. 分支结构

（2）switch语句

```
switch(expression){
    case    const1:statements;break;
    case    const2:statements;break;
        ......
    default:statements;
}
```



其中expression只能是整数类型或字符型，不能是浮点类型。

Example

```
public class IfElse {

    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int a = 0;
        int b = 0;
        if(a<b) {
            System.out.println("执行if语句");
        }
        else {
            System.out.println("执行else语句");
        }

    }

}
```

Problems | Javadoc | Declaration

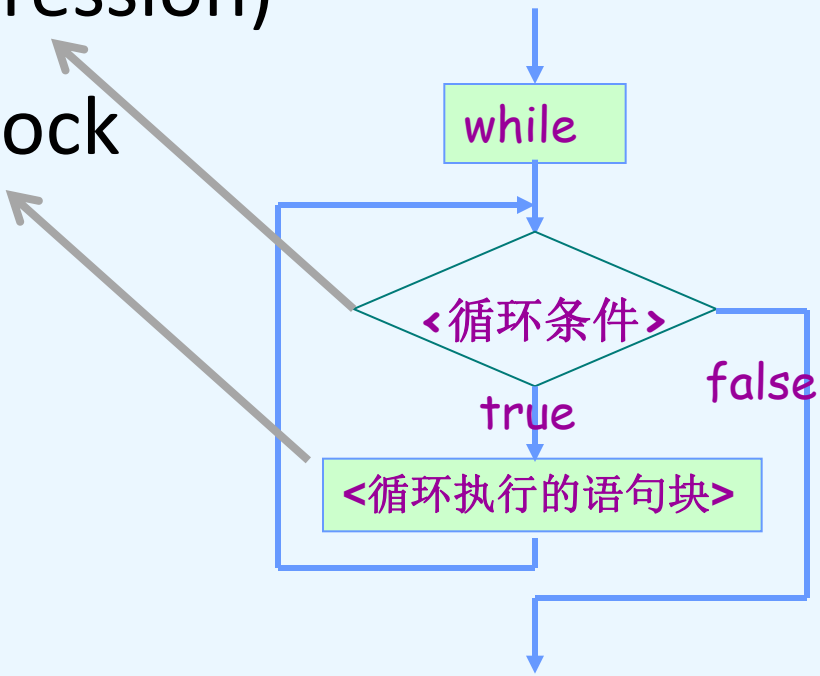
<terminated> IfElse [Java Application] C:\Prog

执行else语句

3. 循环结构

(1) while循环语句

```
while (boolean_expression)
    statement_or_block
```



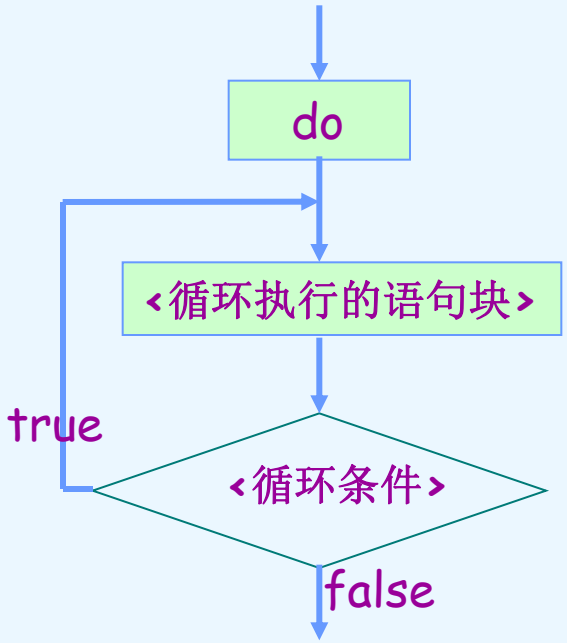
```java
public class WhileTest {

    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        boolean signIn = true;
        int courseCount = 1;
        while (signIn&(courseCount<17)) {
            System.out.println("I will come to the java course"+courseCount);
            courseCount+=1;
        }
    }

}
```

```
I will come to the java course8
I will come to the java course9
I will come to the java course10
I will come to the java course11
I will come to the java course12
I will come to the java course13
I will come to the java course14
I will come to the java course15
I will come to the java course16
```

3. 循环结构

（2）do-while循环语句

```
do
    statement_or_block
while(boolean)
```

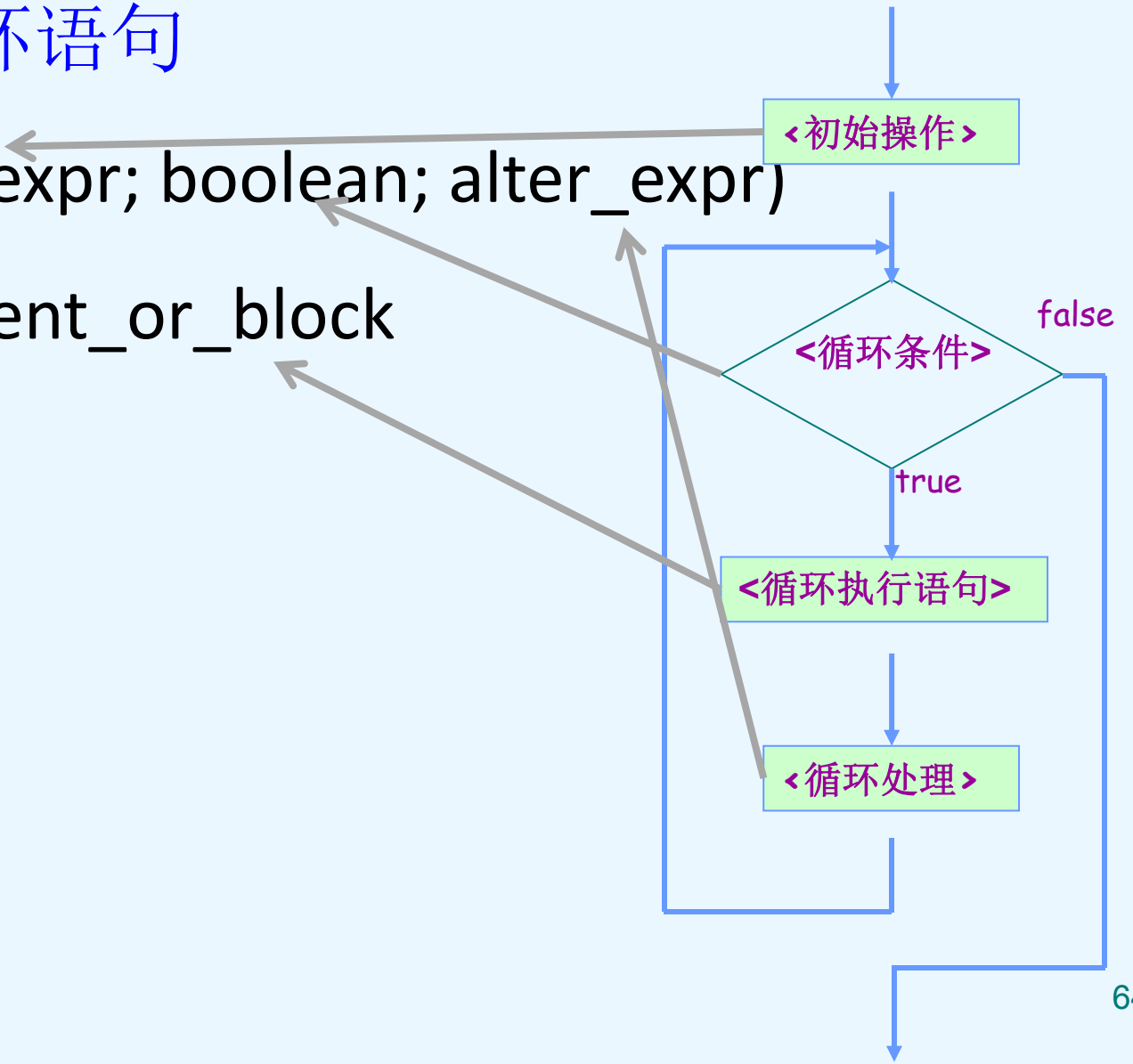


3. 循环结构

(2) for循环语句

- for(init_expr; boolean; alter_expr)

  statement_or_block

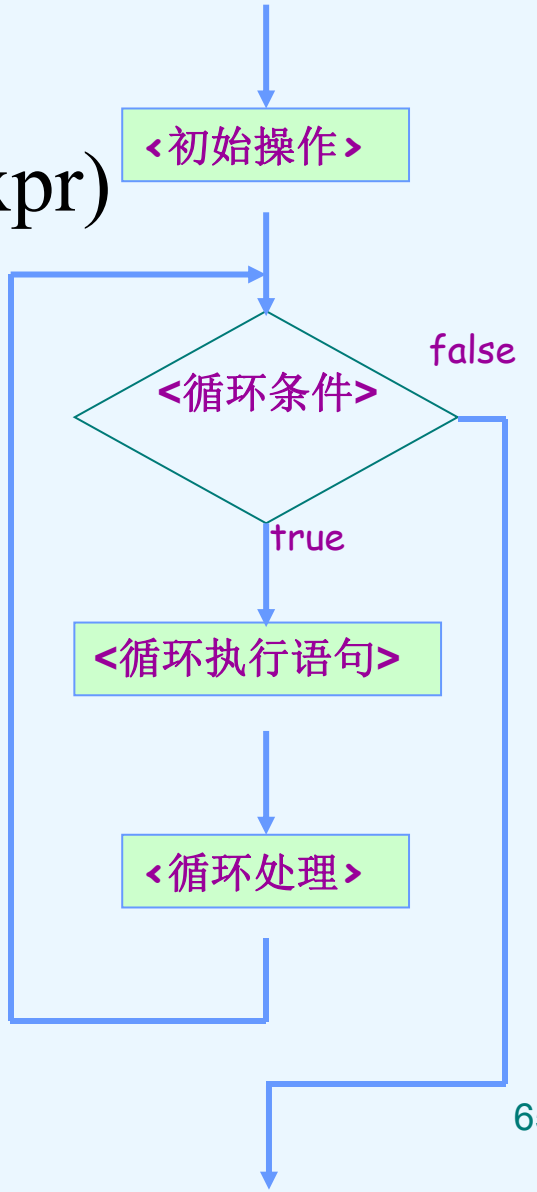


3. 循环语句

（3）for循环语句

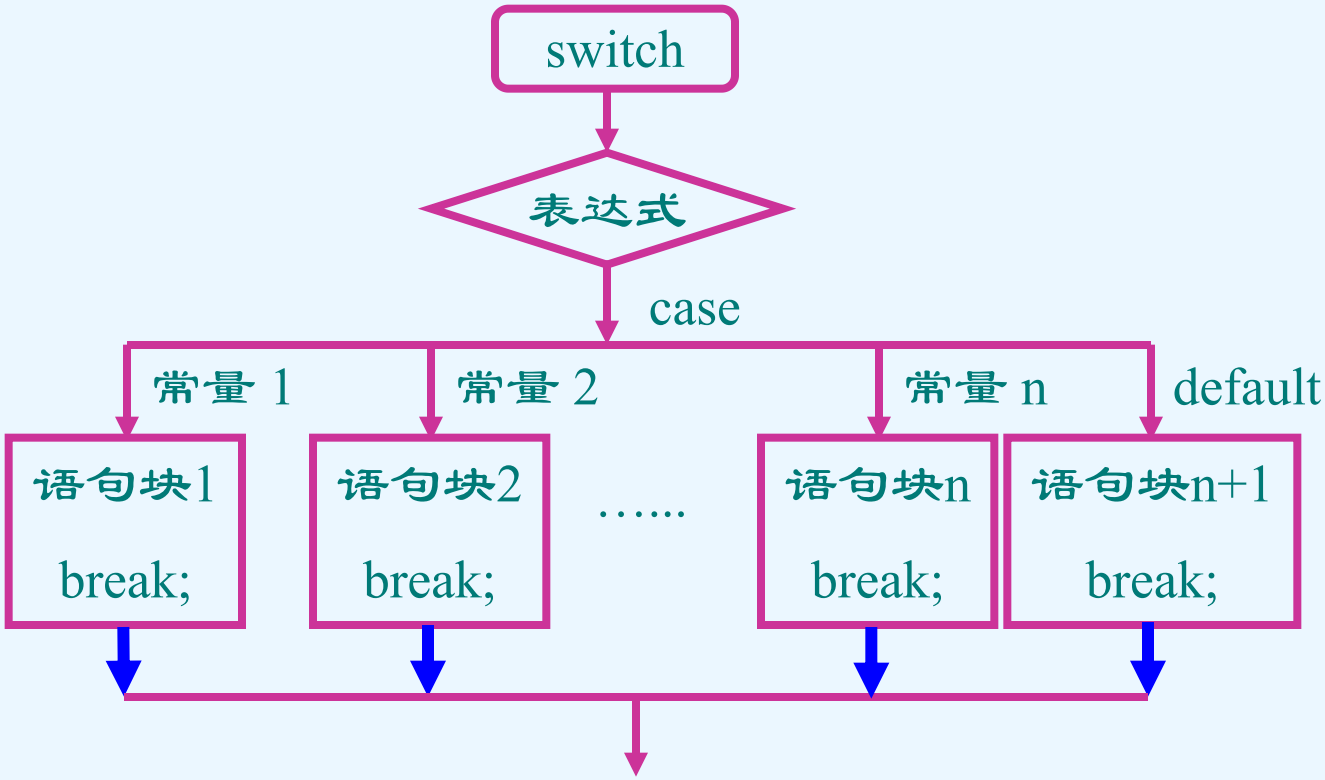
- for(init_expr; boolean; alter_expr)

  statement_or_block



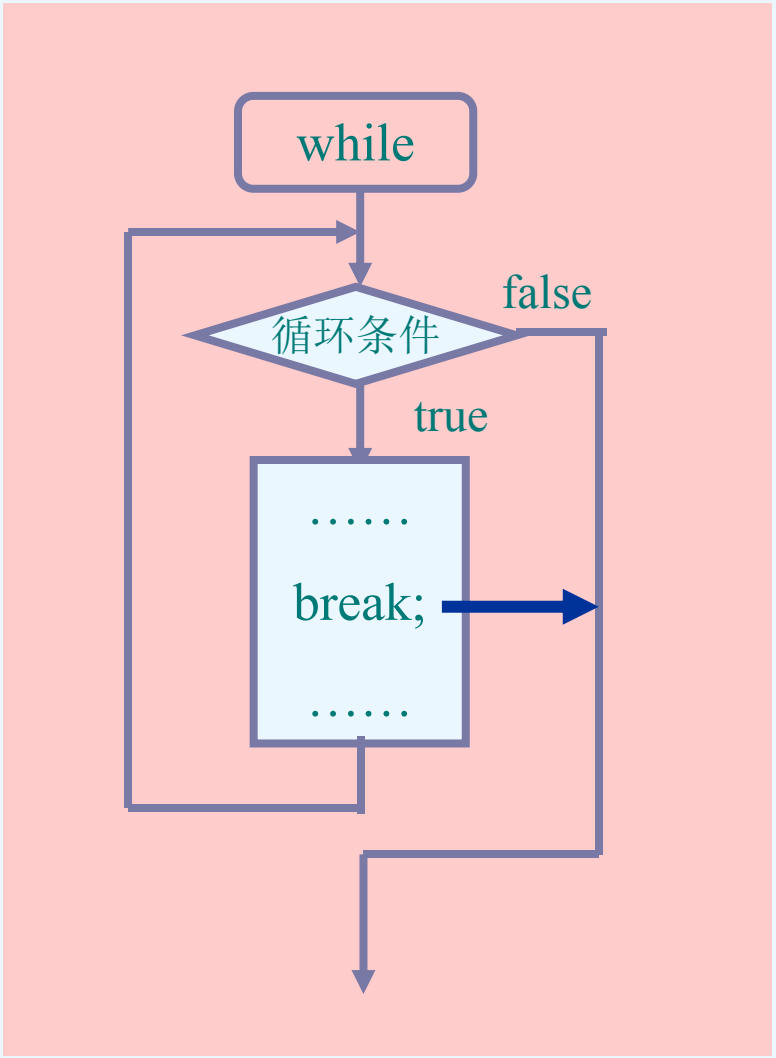
4. break语句和continue语句

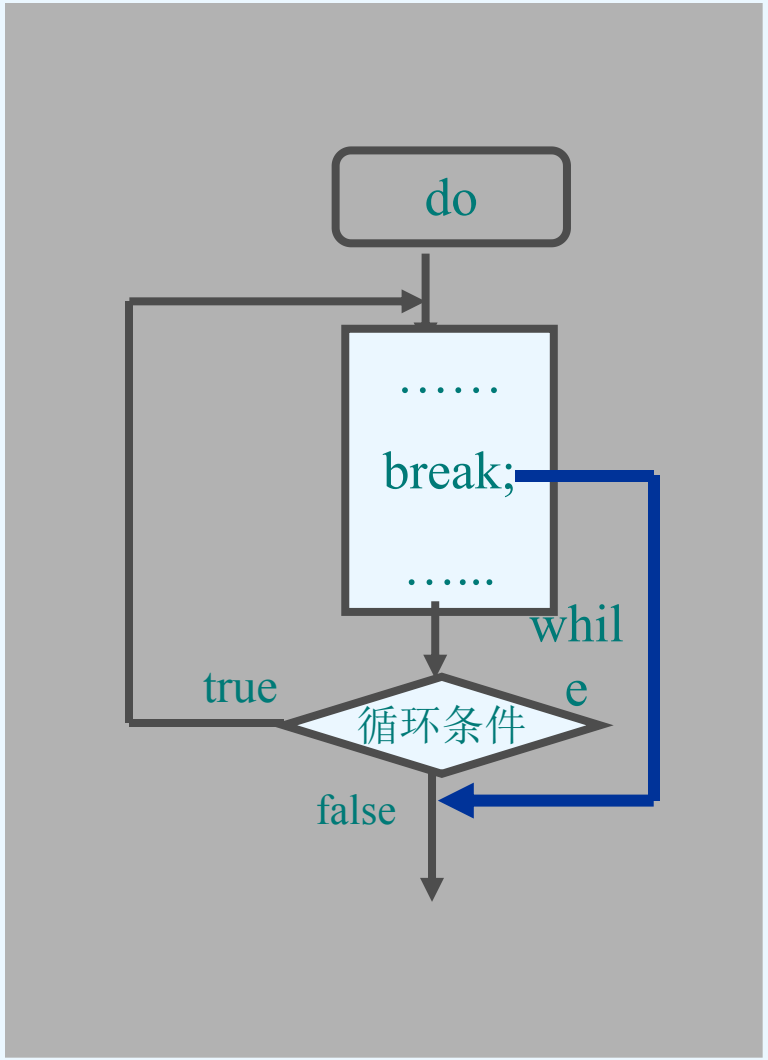
- 不带label的break语句为跳出一个语句层；

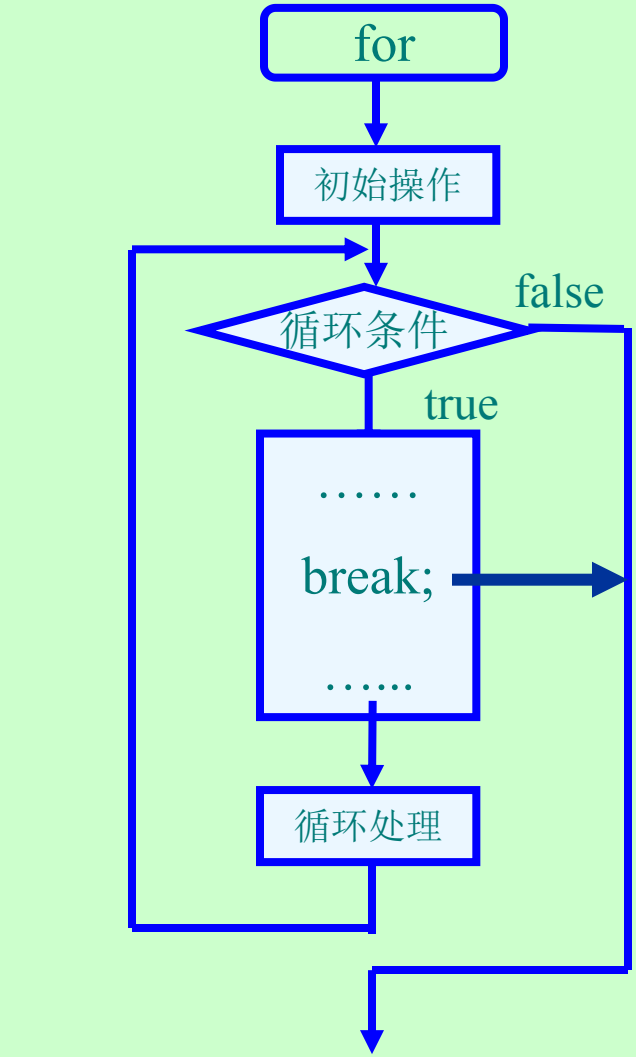


4. break语句和continue语句

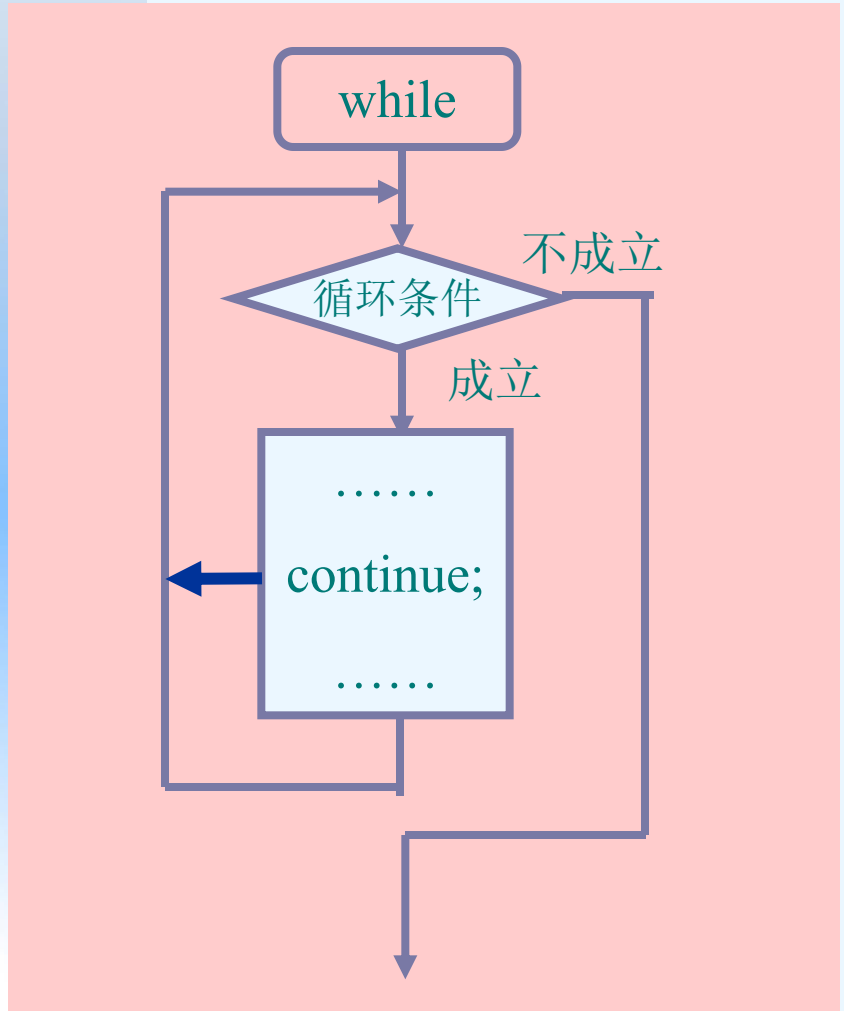
- 不带label的break语句为跳出一个语句层；

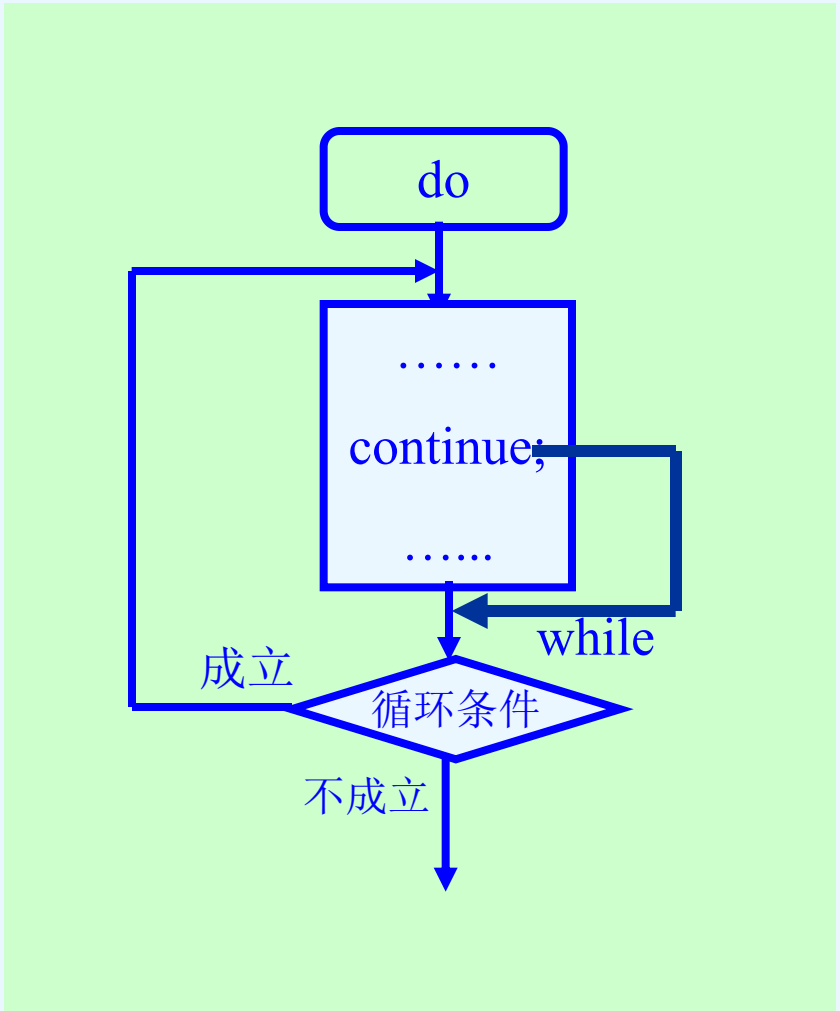


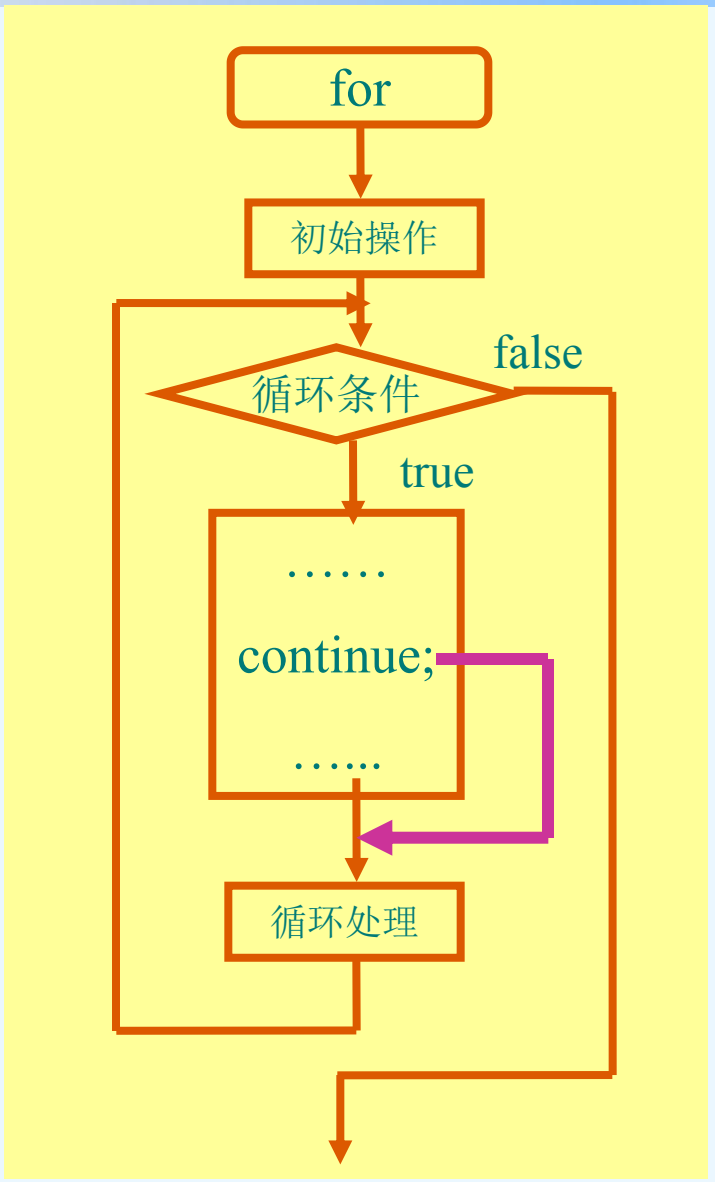



for
初始操作
循环条件
false
true
……
break;
……
循环处理

- 不带label的continue语句与C和C++中完全一样，可以结束某个循环中的一个周期的剩余部分，开始下一个循环；

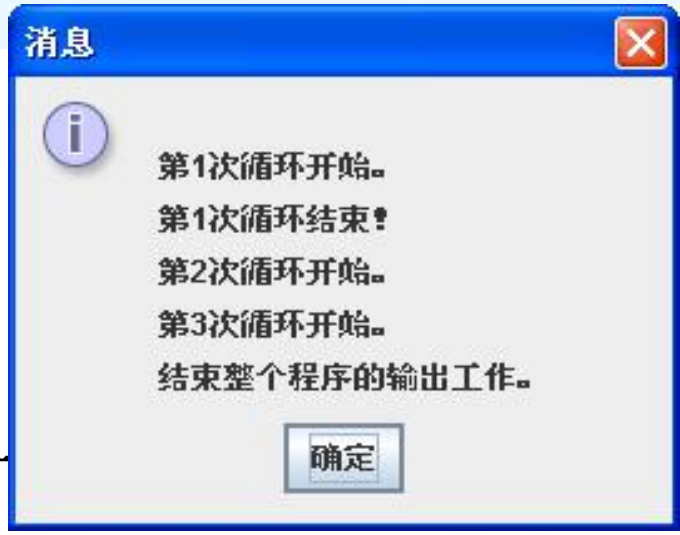





for
初始操作
循环条件
false
true
……
continue;
……
循环处理

[例2.2]break语句和continue语句在循环结构中的流程控制。

<SimpleApp2.java>




```
import javax.swing.JOptionPane;
public class SimpleApp2
{
    public static void main(String[] args)
    {
        String output = "";
        for(int i=1;i<=10;i++) //计划进行十
        {
            output +="\n第"+i+"次循环开始。";
            if(i==2)continue; //第二次循环到此跳过循环体中剩余的部分
            if(i==3)break;    //第三次循环到此结束循环
            output +="\n第"+i+"次循环结束！";
        }
        output +="\n结束整个程序的输出工作。";
        JOptionPane.showMessageDialog(null,output);
        System.exit(0);
    }
}
```

- 带label的break语句的格式为

break label

label为一个标识符，标定一条语句，带label的break语句的用法的作用是跳出label所标定的块。

- 带label的continue语句的格式为

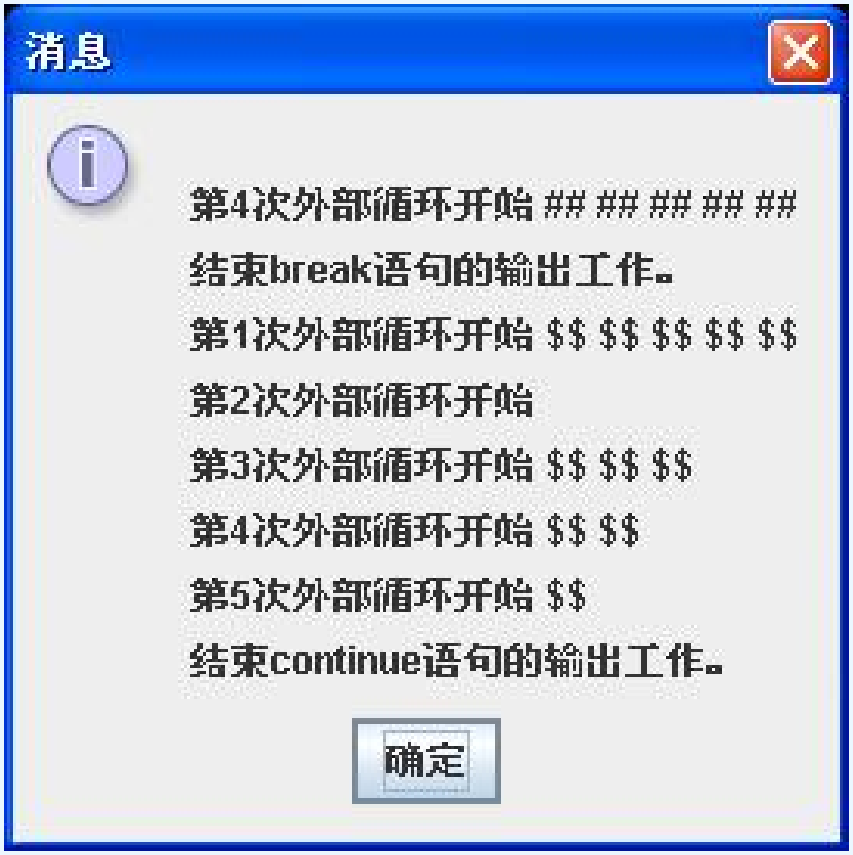
continue label

这里的label仍然是一个标定语句的标识符，带label的continue语句的作用是结束标号指定的循环中的一个周期的剩余部分，开始下一个循环。

[例2.3]带标号的break语句和continue语句在循环结构中的流程控制。<SimpleApp3.java>

```
import javax.swing.JOptionPane;
public class SimpleApp3
{   public static void main(String[] args)
    {       String output = "";
    L01:{
                    for(int i=1;i<=10;i++)      //计划进行十次外部循环
                    {
                            output = "\n第";
                            output +=i;
                            output +="次外部循环开始 ";
                            for(int j=1;j<=8;j++){  //计划进行八次内部循环
                                    if(i==4&&j==6)
                                            break L01;
                                    output += "## ";
                            } //结束内部循环
                            output +="第";
                            output +=i;
                            output +="次外部循环结束";
                    } //结束外部循环
            } //结束L01
            output +="\n结束break语句的输出工作。";
```

```
L02:for(int i=1;i<=5;i++)          //计划进行五次外部循环
        {
                output +="\n第";
                output +=i;
                output +="次外部循环开始 ";
                for(int j=1;j<=6;j++){      //计划进行六次内部循环
                        if(i==2||i+j==7)
                                continue L02;
                        output += "$$ ";
                } //结束内部循环
                output +="第";
                output +=i;
                output +="次外部循环结束";
        } //结束外部循环
        output +="\n结束continue语句的输出工作。";

        JOptionPane.showMessageDialog(null,output);
        System.exit(0);
    }
}
```


消息
第4次外部循环开始 ## ## ## ## ##
结束break语句的输出工作。
第1次外部循环开始 $$ $$ $$ $$ $$
第2次外部循环开始
第3次外部循环开始 $$ $$ $$
第4次外部循环开始 $$ $$
第5次外部循环开始 $$
结束continue语句的输出工作。
确定

5. return语句

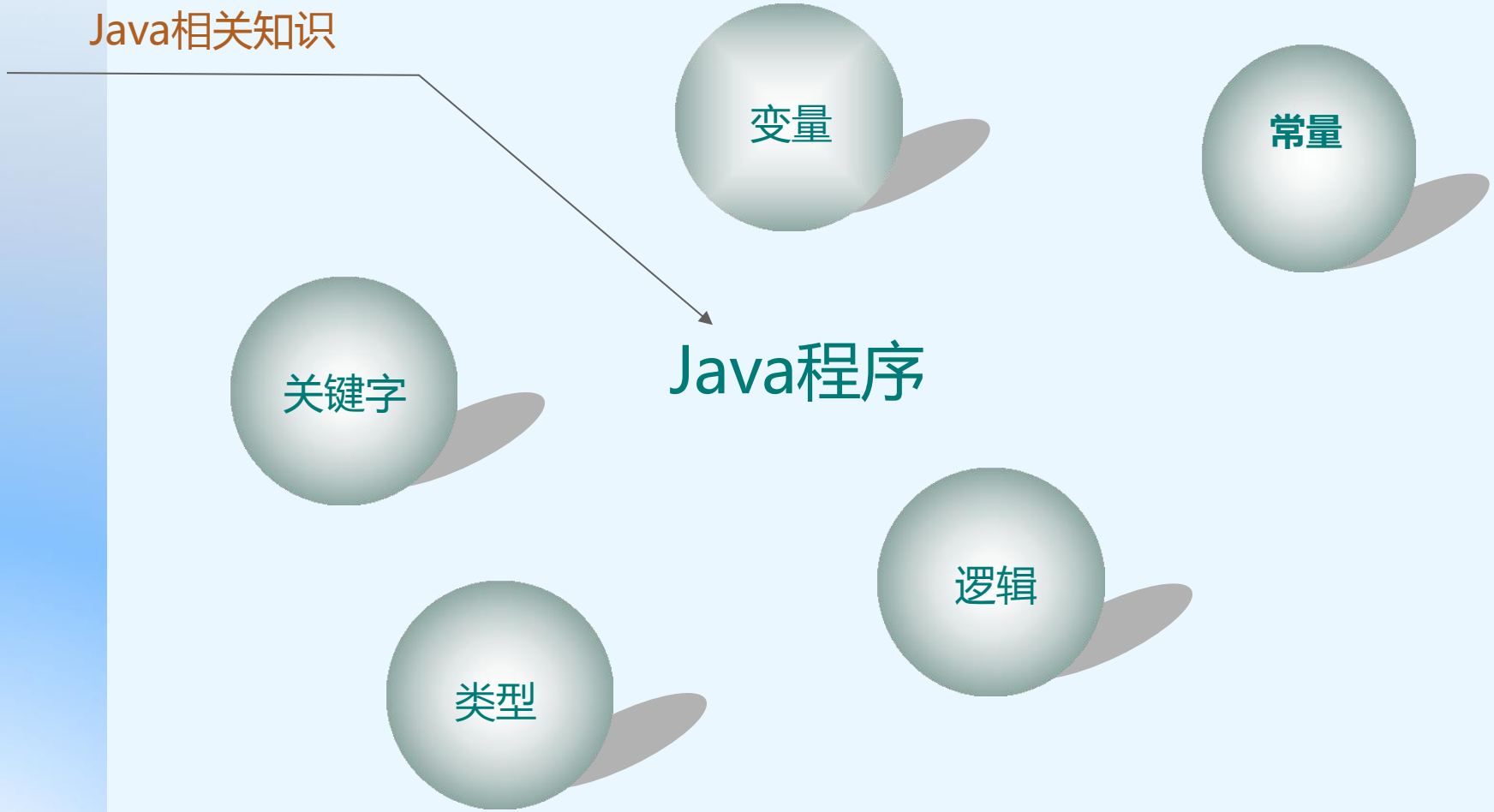
return variable_or_expression;

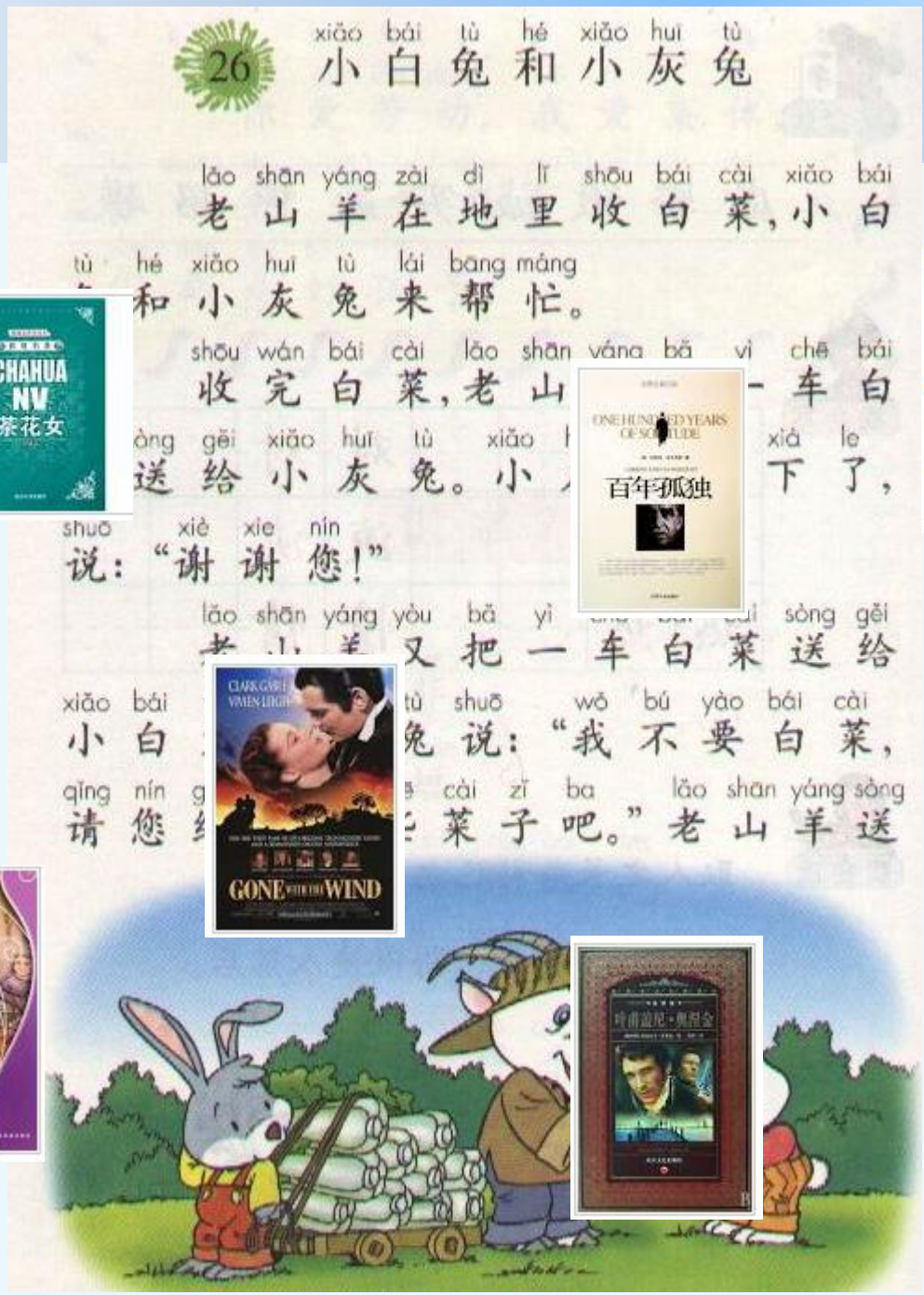
这个语句用在方法体中，它的作用是返回一个与方法的返回类型一致的值给方法的调用。

return;

这个语句不返回任何数值，只向调用方法的语句返回控制。

Java结构化程序设计




26
xiǎo bái tù hé xiǎo huī tù
小白兔和小灰兔
lǎo shān yáng zài dì lǐ shōu bái cài, xiǎo bái
老山羊在地里收白菜，小白
tù hé xiǎo huī tù lái bāng máng
兔和小灰兔来帮忙。
shōu wán bái cài, lǎo shān yáng bǎ yì chē bái
收完白菜，老山羊把一车白
cài sòng gěi xiǎo huī tù. xiǎo
菜送给小灰兔。小
xià le,
下了，
shuō xiè xie nín
说：“谢谢您！”
lǎo shān yáng yòu bǎ yì
老山羊又把一
chē bái cài sòng gěi
车白菜送给
xiǎo bái
小白
tù shuō wǒ bú yào bái cài,
兔说：“我不要白菜，
qǐng nín g
请您
cài zi ba. lǎo shān yáng sòng
菜子吧。”老山羊送

CHAHUA
NV
茶花女

ONE HUNDRED YEARS OF SOLITUDE
百年孤独


德伯家的苔丝

GONE WITH THE WIND


悲惨世界


神秘岛

Have a nice Weekend