讲义02：Java语言基础与流程控制

華南農業大學
South China Agricultural University

主要内容

- 讲解内容
  - 标识符与关键字、变量、常量
  - 数据类型、数据类型转换
  - 命令行窗口的输入、输出
  - 运算符与表达式
  - 语句与流程控制
- 内容来源资料
  - 教材第3章、第4章
  - API文档：
    - java.util.Scanner
    - java.lang.System

1 基本数据类型

数据类型	关键字	存储大小	取值范围
逻辑类型	boolean		true，false
整数类型	byte	8 bit	$-2^{7} \sim 2^{7}-1$
	short	16 bit	$-2^{15} \sim 2^{15}-1$
	int	32 bit	$-2^{31} \sim 2^{31}-1$
	long	64 bit	$-2^{63} \sim 2^{63}-1$
浮点类型	float	32 bit	$1.4*10^{-45} \sim 3.4*10^{38}$
	double	64 bit	$4.9*10^{-324} \sim 1.7*10^{308}$
字符类型	char	16 bit	Unicode字符集

实例演示各数据类型

2 标识符与关键字

- Java语言标识符的规则
  - 标识符由字母、数字、下划线和美元符号组成的字符串。
  - 第一个字符必须为：字母、下划线或$。
  - 标识符不能是关键字。
  - 标识符不能是true、false和null。
  - 标识符可以有任何长度。
- Java标识符区别字母的大小写。
- 标识符的作用：命名

2 标识符与关键字

Java语言共有50个关键字

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

3 变量与常量

变量用于在程序中存储数据，每个变量必须有确定的名称和数据类型。语法形式：

```
数据类型 变量名1，……，变量名n;
```

```
int x;
double radius, interestRate;
char a;
boolean passed = false;
```

注：按照习惯，变量名用小写字母。如果一个名字由多个单词构成，则除第一个单词外，其它单词首字母大写。

3 变量与常量

常量表示永远不变的值，包括：直接量和定名常量

定名常量语法：final 数据类型 常量名 = 值;

✓
```
final double PI = 3.14;
```

✓
```
final double PI;
PI = 3.14;
```

✗
```
final double PI;
PI = 3.14;
PI = 3.1415;
```

注：按照习惯，定名常量用大写字母，名字由多个单词组成时，用下划线连接。

4 简单数据类型的级别与类型转换

数值型数据类型级别由低(左)向高(高)排列

byte short char int long float double

自动类型转换：级别低的类型向级别高的转换时。例如：

```
float x = 100;
int k = ‘A’;
```

强制类型转换：级别高的类型向级别高的转换时。例如：

```
int x = (int)100.0;
byte k = (byte)200;
```

5 命令行的输入和输出

输出使用System.out对象的3个方法：

System.out.print(字符串)
输出字符串后不换行

System.out.println(字符串)
输出字符串后自动换行

System.out.printf(格式字符串, 值1, 值2, … , 值n);
输出方式与C语言的printf函数类似

5 命令行的输入和输出

输入使用java.util.Scanner类的对象的方法

方法	功能描述
byte nextByte()	输入方法返回值类型的基本类型数据
short nextShort()	
int nextInt()	
long nextLong()	
float nextFloat()	
double nextDouble()	
String next()	输入一个字符串，以空白字符结束
String nextLine()	输入一行字符串，即以回车结束

5 命令行的输入和输出

Scanner对象输入的使用方式：

源程序的最前面(类外部)导入Scanner类,语句：

```
import java.util.Scanner;
```

在需要输入的方法中定义输入对象变量并创建对象,语句：

```
Scanner input = new Scanner(System.in);
```

其中：System.in表示标准输入设备，默认是键盘

需要输入的地方调用恰当方法输入并赋值给变量，例如：

```
int age = input.nextInt();
double score = input.nextDouble();
String name = input.next();
String address = input.nextLine();
```

6 运算符与表达式

算术运算　+ - * / %

增量和减量运算：++var　var++　--var　var--

赋值运算：=

简捷赋值运算：+=　-=　*=　/=　%=

> E1 op= E2　等价于　E1 = (T)((E1) op (E2))
> 其中T是E1的类型，E1的值只被计算一次

条件运算：

```
var = booleanExpression ? Expression1 : Expression2;
```

6 运算符与表达式

不同类型数据算术运算时的转换规则：

1. 如果运算对象之一是double，就将另一个转换为double型。
2. 否则，如果运算对象之一是float，就将另一个转换为float型。
3. 否则，如果运算对象之一是long，就将另一个转换为long型。
4. 否则，两个运算对象都转换为int型。

6 运算符与表达式

Java提供6种比较运算符，又称为关系运算符.

运算符	名称	举例	结果
<	小于	1<2	true
<=	小于等于	1<=2	true
>	大于	1>2	false
>=	大于等于	1>=2	false
==	等于	1==2	false
!=	不等于	1!=2	true

6 运算符与表达式

Java提供4种逻辑运算符.

非

p	! p
true	false
false	true

与

p1	p2	p1 && p2
true	true	true
true	false	false
false	true	false
false	false	false

或

p1	p2	p1 || p2
true	true	true
true	false	true
false	true	true
false	false	false

异或

p1	p2	p1 ^ p2
true	true	false
true	false	true
false	true	true
false	false	false

6 运算符与表达式

结合性

双目运算符：左结合
赋值运算符：右结合
单目运算符：右结合
三目运算符：右结合

优先级

优先级	运算符
最高级	var++和 var--
	+、-（正、负），++var、--var
	（type）（类型转换）
	!
	*、/、%
	+、-
	<、<=、>、>=
	==、!=
	&
	|
	&&
	||
最低级	=、+=、-=、*=、/=、%=

6 运算符与表达式

Java表达式计算的规则：

规则1：可能的情况下，从左向右计算所有子表达式。

规则2：根据运算符的优先级进行运算

规则3：对优先级相同的相邻运算符，根据结合方向运算。

示例：分析下面两个表达式计算后变量x的值。

```
int a = 0;
int x = a++ + a++ + a++;
```

x的值：3

```
int a = 0;
int x = ++a + ++a + ++a;
```

x的值：6

7 语句概述

- 方法调用语句
  System.out.println(“Hello”);
- 表达式语句
  x = 123;
- 复合语句(语句块)
  { 一条或多条语句 }
- 空语句
  ;
- package语句 与 import语句
- 流程控制语句

7.1 if语句

- 简单if语句

```
if(布尔表达式) {
    语句(组);
}
```

- if…else语句

```
if(布尔表达式) {
    真值 - 语句(组);
}
else {
    假值 - 语句(组);
}
```

- if语句的嵌套规则
  else总是与同一块中最近的未匹配的if子句匹配.

7.2 switch语句

```
switch (switch-expression) {
  case value1: statement(s)1;
               break;
  case value2: statement(s)2;
               break;
  ……
  case valueN: statement(s)N;
               break;
  default: statement(s) for default;
}
```

说明:

switch-expression的计算结果必须是char、byte、short或int, value1, …,valueN的类型必须与switch-expression的类型相同.

7.3 循环语句

- while 语句

```
while (循环条件) {
  语句 (组) ;
}
```

- do - while 语句

```
do {
  语句 (组) ;
} while (循环条件);
```

- for 语句

```
for(初始化操作; 循环条件; 每次循环后的操作) {
  语句 (组) ;
}
```

- 循环语句的嵌套规则

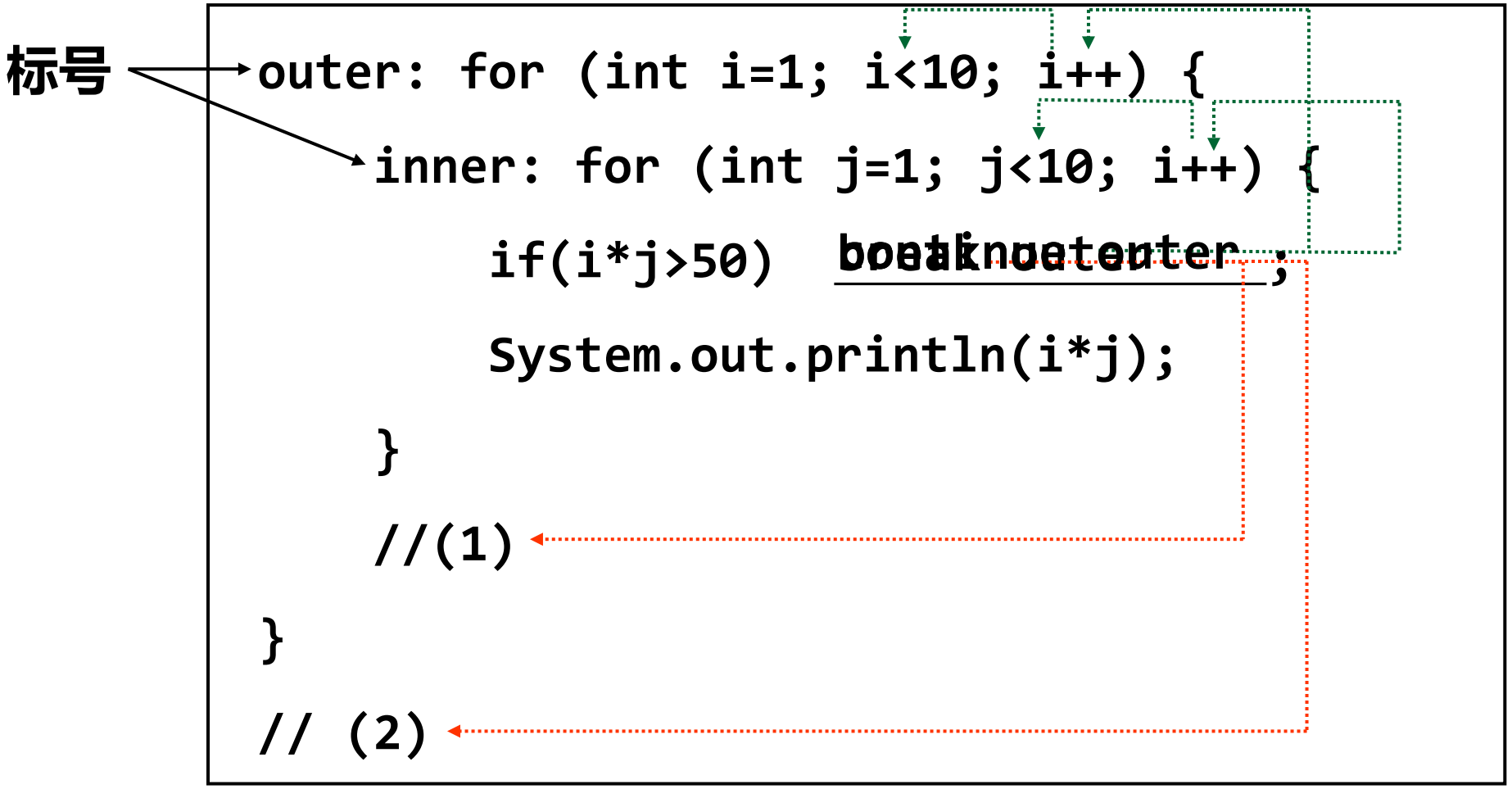
7.4 break 与 continue

- break;
  跳出包含它的最内层循环, 或跳出包含它的最内层switch结构。
- break 标号;
  跳出标号指明的循环。

- continue;
  结束包含它的最内层循环的当前迭代，程序控制转到循环体末尾。
- continue 标号;
  结束标号指明的循环的当前迭代。

7.4 break 与 continue

break和continue语句使用示例：



示例：显示素数

题目要求：显示前50个素数，输出时每行10个素数。

示例参考代码如下，参见：examples/lecture02/primenumbers

```
package primenumbers;

public class PrimeNumber {
    public static void main(String[] args) {
        final int NUMBER_OF_PRIMES = 50; // 需要显示的素数的个数
        final int NUMBER_OF_PRIMES_PER_LINE = 10; // 每行显示的素数的个数

        int count = 0; // 已经输出的素数的个数
        int number = 2; // 检测素数的起点

        System.out.printf("前%d个素数的列表：\n", NUMBER_OF_PRIMES);

        while (count < NUMBER_OF_PRIMES) {
            boolean isPrime = true; // 标记当前数是否是素数

            for (int divisor = 2; divisor <= (int) Math.sqrt(number); divisor++) {
                if (number % divisor == 0) {
                    isPrime = false;
                    break;
                }
            }

            if (isPrime) {
                count++;

                System.out.print(number + " ");
                if (count % NUMBER_OF_PRIMES_PER_LINE == 0) {
                    System.out.println();
                }
            }

            number++; // 准备好下一个要判断的整数
        }
    }
}
```

示例显示了Java语言的运算符、表达式和语句的基本语法，同时展示了一些与C语言不同的编程习惯。

课后工作

- 结合教材内容，复习课堂讲授内容
- 完成作业网站：练习1、练习2