第二章 Java 基础

讲授思路

- 标识符、关键字
- 数据类型
- 运算符和表达式
- 流程控制

标识符

- 标识符概念：
  - Java语言中，对于变量，常量，函数，语句块也有名字，我们统统称之为Java标识符。
  - 标识符是用来给类、对象、方法、变量、接口和自定义数据类型命名的。
- 举例：
  - class Student {};
  - int identifier ;
  - String userName ;

标识符

- 标识符组成：
  - Java标识符由数字，字母和下划线(_)，美元符号（$）组成，只能以字符、“_”或“$”开头。
  - 标识符是大小写敏感的并且未规定最大长度。
  - 标识符不能是Java关键字或保留字。
- 举例：
  - 下面的标识符是合法的：
    - myName，My_name，Points，$points,_sys_ta
  - 下面的标识符是非法的：
    - #name，25name，class，&time，if

标识符

- Java语言中的命名约定：
  - 类和接口名。每个字的首字母大写，含有大小写。
    - 例如，MyClass，HelloWorld，Time等。
  - 方法名。首字母小写，其余的首字母大写，含大小写。尽量少用下划线。
    - 例如，myName，setTime等。
  - 常量名。基本数据类型的常量名使用全部大写字母，字与字之间用下划线分隔。对象常量可以大小写混写。
    - 例如，SIZE_NAME。
  - 变量名。可大小写混写，首字符小写，不用下划线，少用美元符号。给变量命名时尽量做到见名知义。

关键字

- 关键字是已被Java占用的标识符，有专门的意义和用途，在Java编程语言中使用的关键字如下：

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

数据类型

- 数据类型是程序设计语言描述事物、对象的方法。Java数据类型分为基本类型和引用类型两大类。
- 基本类型
  - 整数类型 byte, short, int, long
  - 浮点类型 double, float
  - 字符类型 char
  - 布尔类型 boolean
- 引用类型
  - 类、接口、数组、枚举

数据类型

- 整型类型：用于表示没有小数部分的数值，它允许是负数。Java提供了4种整型，具体内容如表所示：

类型	存储需求	取值范围	默认值
byte	1字节	-128~127	0
short	2字节	-2^{15}~$2^{15}-1$	0
int	4字节	-2^{31}~$2^{31}-1$	0
long	8字节	-2^{63}~$2^{63}-1$	0

数据类型

- 浮点类型：用于表示有小数部分的数值。Java有2种浮点类型，具体内容如表所示：

类型	存储需求	取值范围	默认值
float	4字节	-3.40292347E+38 ~3.40292347E+38 F	0.0
double	8字节	-1.79769313486231570E+308~ 1.79769313486231570E+308	0.0

```
float fNum1 = 10;          //编译通过（int转float）
float fNum2 = 10.0;          //编译不通过（10.0认为是
double，不能转float）
float fNum3 = 10.0f;         //推荐写法
float fNum4 = (float) 10.0;//类型强转，会发生截断
double dNum1 = 10.0;
double dNum2 = 10.0d;
```

数据类型

- 字符类型：用于表示单个字符。通常用来表示字符常量。
  - 占用2个字节
  - 采用unicode编码，字符的存储范围在\u0000~\uFFFF
  - 编码中的第一个字节仍与 ASCII 兼容
- 布尔类型：有false和true两个值,用来判定逻辑条件。
  - 整数值和布尔值之间不能进行相互转换

```
int n = 10;
if(n){          // 编译错误
System.out.println(n);
}
```

变量的定义

- 变量是在程序运行过程中其值可以被改变的量。变量包括变量名、变量值两部分。
- 变量的定义
  - 变量的声明
  - 变量的初始化

变量的定义

- 变量声明的格式如下：
  - 数据类型 变量名1[,变量名2，...];
- 举例：

```
//声明一个存放整型且名是stuNo的变量
int stuNo;
//声明浮点型变量x,y.
float x,y;
```

```
float x;
float y;
```

变量的定义

- 变量的初始化：声明一个变量后，通过赋值语句对变量进行显示的初始化。
- 举例：

```
//变量声明
double salary;
//变量初始化
salary = 5000.0;
```

```
//声明一个double类型的变量并赋值为50.0
double salary=50.0;
```

```
int x =10 , y = 20 , z = 30 ;
int x , y = 20, z ;
```

常量的定义

- 常量是在程序运行过程中其值始终保持不变的量。
- Java使用关键字final来定义常量。
- 常量定义的语法格式
  - final 数据类型 常量名称 = 值；
- 举例：

```
final int NUM = 12;
final float PI = 3.14159;
```

```
final int AGE = 10;
int final NUM = 20;      //编译错误(注意位置)
```

- 常量被赋值之后，就不能再改变了。
- 习惯上，常量名使用大写定义。

运算符和表达式

- 运算符
- 表达式
- 运算符优先级
- 类型转换

运算符

- Java中支持的运算符

算数运算符	+ - * / %
自增运算符、自减运算符	++ --
关系运算符	> >= < <= == !=
逻辑运算符	&& || !
三元运算符	?:
赋值运算符	= += -= *= %=
位运算符	& | ^
字符串连接运算符	+
instanceof比较	instanceof检查对象是否是某种类型

运算符（自增、自减运算符）

- 自增运算符、自减运算符
  - 在Java中，借鉴了C和C++的实现方式，也使用了自增、自减运算符：n++将变量n的当前值加1；n--将n的值减1。
- 举例：

```
int  n = 12;
int  m = 12;
n++;
m --;
```

- 它的操作数不能是数值。例如，4++是一条非法的语句。

运算符（自增、自减运算符）

- 这两个运算符有两种形式
  - “后缀”形式：n++，m --;
  - “前缀”形式：++n ，--m;
- 举例：

```
int m = 7;
int n = 7;
int a = m ++;
int b = ++ n;
```

```
int n = 10;
System.out.println(n++);
System.out.println(++n);
System.out.println((n++)+1);
System.out.println(n);
//System.out.println((++n)++);  //编译错误
```

```
10
12
13
13
```

运算符（关系运算符）

- 关系运算符
  - 使用两个等号 ==检测是否相等。例如，3 == 7的值为false。
  - 使用 != 检测是否不相等。例如，3 != 7的值为true。
  - 经常使用的运算符还有 <、>、<= 和 >=。

运算符（逻辑运算符）

- 逻辑运算符
  - &&表示逻辑“与”，例如 x&&y。
  - || 表示逻辑“或”，例如 x || y。
  - !表示逻辑“非”，例如 !x。
  - &&和 || 是按照“短路”方式求值的。如果第一个操作数已经能够确定表达式的值，第二个操作数就不必计算了。
- 举例：

```
x!=0 && ( 1/x > (x+y) )
```

运算符

- 三元运算符
  - 表达式：condition ? expression 1 : expresson2
  - 条件condition为真时计算第1个表达式，否则计算第2个表达式
- 举例：返回x和y中较小的那个值。

```
x < y ? x : y
```

表达式

- 表达式概念
  - 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
- 表达式的类型和值
  - 表达式中操作数进行运算得到的结果称为表达式的值
  - 表达式值的数据类型即为表达式的类型
- 表达式的运算顺序
  - 应按照运算符的优先级由高到低执行

运算符优先级

运算符	结合性
! ~ ++ -- + - (一元运算)	从右向左
* / %	从左向右
+ -	从左向右
<< >>	从左向右
< <= > >= instanceof	从左向右
== !=	从左向右
&	从左向右
^	从左向右
|	从左向右
&&	从左向右
||	从左向右
? :	从右向左
= += -= *= /= %=	从右向左

类型转换

- 基本数据类型之间可以进行相互转换。
  - 隐式转换
    - 当进行类型加宽转换时可以自动实现，被称为隐式转换。
    - 举例 ：

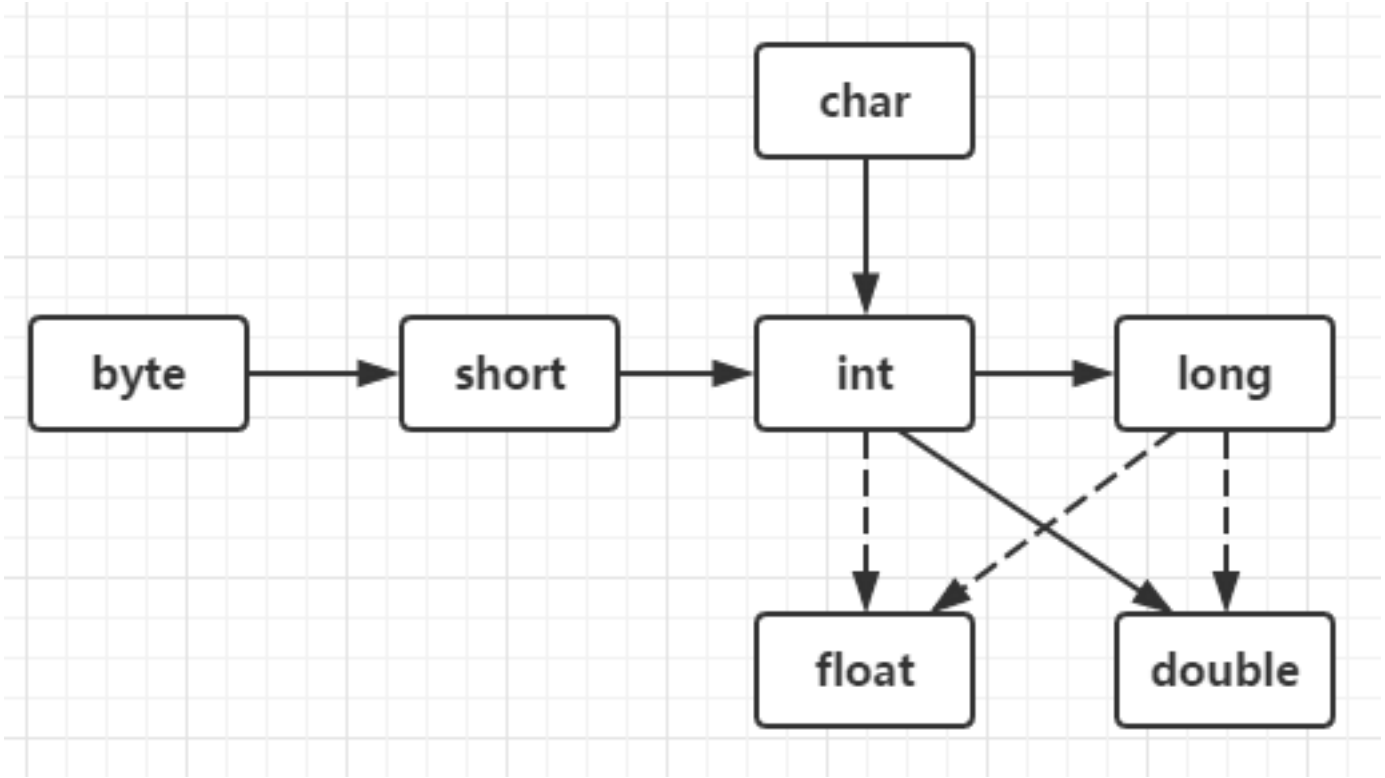
```
int n = 5 ;
double  f = n ;
```

  - 强制转换
    - 当进行类型收缩转换时转换必须进行显式转换，被称为强制转换。
    - 举例：

```
double  f  = 5.5 ;
int n = ( int ) f ;
```

类型转换

- 数据类型按容量大小排序
  - byte→ short→ (char)→ int→long→float→ double。
  - byte，short, char之间不会相互转换，他们三者在计算时首先转换为 int 类型。

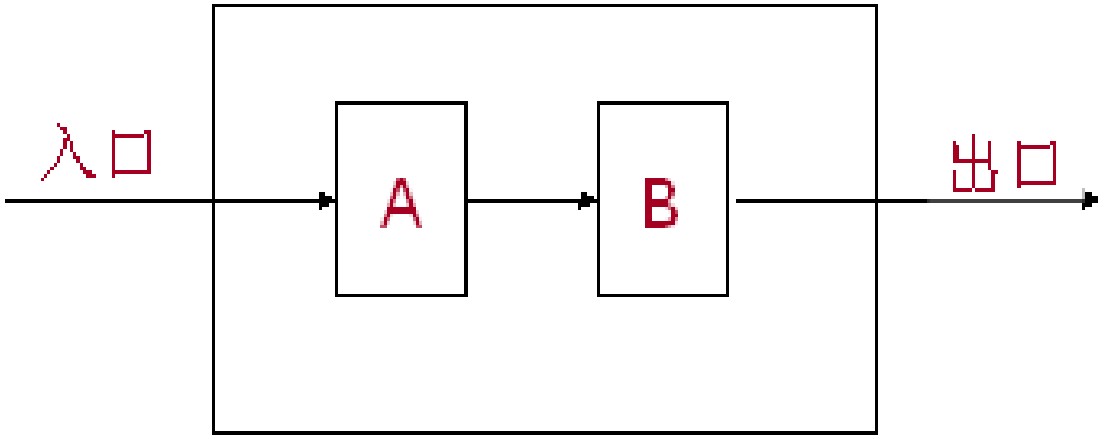


讲授思路

- 顺序流程
- 分支流程
- 循环流程

顺序流程

- 顺序流程是按照语句顺序依次执行一系列语句(或语句块)。顺序流程是最基本的控制流程。
- 顺序流程的示意图：



分支流程

- 条件语句使部分程序可根据某些表达式的值被有选择的执行。
  - 条件分支
    - if...else
    - switch...case
    - ...

条件分支流程 if……else

- 基本语法：
  - if (条件表达式) {
    - //条件为真时执行
    - 语句块1
  - } else {
    - //条件为假时执行
    - 语句块2
  - }

条件分支流程 if……else

- 情况一：
  - if (条件表达式) {
  - //条件为真时执行
  - 语句块1
  - }　　--->无else的情况
- 情况二：
  - if (条件表达式1) {
  - 语句块1
  - } else if (条件表达式2) {
  - 语句块2
  - } else if (条件表达式n-1){
  - 语句块n-1
  - } else {
  - 语句块n
  - }--->重复地交替出现if...else if ...的情况

条件分支流程 if……else

- 课堂练习：有两个整型变量a，b，请在控制台上输出a与b中值较大的那个数。

```
int a = 10;
int b = 5;
if ( a > b ) {
        System.out.println("最大值=" + a);
} else {
        System.out.println("最大值=" + b);
}
```

条件分支流程 switch……case

- switch语句是多分支的条件语句。
  - 嵌套的if语句可以处理多分支条件，但没有switch语句直观。
- 基本语法：
  - switch (表达式) {
    - case constant1 :
      - 语句;
      - break;
    - case constant2 :
      - 语句;
      - break;
    - default:
      - 语句;
      - break;
  - }

条件分支流程 switch……case

- 课堂练习:输入学生成绩等级，输出对应的分数范围。成绩划分为A、B、C、D四等。其中A等表示90分及以上，B等表示70分到90分之间，C等表示60分到70分之间，D等表示不及格。

条件分支流程 switch……case

```
switch ( grade )
{
        case 'A'  :
                System.out.println( "你的成绩在90分以上" ) ;
                break ;
        case 'B'  :
                System.out.println( "你的成绩在75-90分之间" ) ;
                break;
        case 'C'  :
                System.out.println( "你的成绩在60-75分之间" ) ;
                break;
        case 'D'  :
                System.out.println( "你的成绩在60分以下" ) ;
                break;
        default :
                System.out.println( "成绩等级错误" ) ;
}
```

循环流程

- 循环语句是根据条件，要求程序反复执行某些操作。
- Java中的循环包括：
  - for循环
  - while循环
  - do……while循环
  - 增强型for循环
- 完整的循环语句一般包括四部分内容：
  - 初始化部分：设置循环的初始条件
  - 迭代部分：用来更新循环控制条件
  - 终止部分：退出循环的条件判断
  - 循环体部分：被反复执行的代码

循环流程- for

- 当循环变量在指定范围内变化时，重复执行循环体，直到循环变量超出了指定的范围时退出。
- 基本语法：
  - for（ 初始化表达式; 终止条件表达式; 更新表达式) {
    - 循环体
  - }
- 注意：
  - 初始化表达式中的变量必须声明并赋值。
  - 终止条件表达式的值必须为布尔值。

循环流程- for

- 课堂练习：计算任意给定的两个数之间的所有数之和。

```
public class Calculator{
    int sum( int begin, int end ) {
            int sum = 0;
            for ( int i = begin; i <= end; i++ ) {
                sum += i ;
            }
            return sum ;
     }
    public static void main(String[] args) {
            Calculator cal = new Calculator() ;
            int sum = cal.sum( 1, 100 ) ;
            System.out.println( "1~100的和是：" + sum ) ;
     }
}
```

循环流程- while

- 当条件满足时进入，重复执行循环体，直到条件不满足时退出。
- 基本语法：
  - while(循环条件表达式) {
    - 循环体
  - }

循环流程- while

- 课堂练习：已知 1×2+2×3+3×4+…+n×（n+1）＜1000求n的最大值。

循环流程- while

```
public class Calculator{
    int computeN( int totalNum ) {
            int  s = 0 ;
            int  n = 1 ;
            while  ( s < totalNum ) {
            s += n * ( n + 1 ) ;
            n = n + 1 ;
            }
            return n – 2 ;
   }
   public static void main ( String[] args ) {
           Calculator  cal = new Calculator() ;
            int n= cal.computeN( 1000 ) ;
           System.out.println( “n的最大值是：” + n ) ;
    }
}
```

循环流程- do……while

- 无条件进入，执行一次循环体后判断是否满足条件，当条件满足时重复执行循环体，直到条件不满足时退出.
- 基本语法：
  - do {
    - 循环体
  - } while (循环条件表达式);

循环流程- do……while

- 课堂练习：已知 1×2+2×3+3×4+…+n×（n+1）＜1000求n的最大值。

循环流程- do……while

```
public class Calculator{
    int computeN( int totalNum ) {
            int s = 0 ;
            int n = 1 ;
            do {
            s += n * ( n + 1 ) ;
            n = n + 1 ;
            } while ( s < totalNum )
            return n – 2 ;
   }
   public static void main( String[]  args ) {
    Calculator  cal = new Calculator() ;
     int n= cal.computeN( 1000 ) ;
           System.out.println( “n的最大值是：" + n ) ;
    }
}
```

循环流程-增强型for

- Java提供了一个更为简洁的循环语句，用于数组或者集合的遍历。
- 基本语法：
  - for (类型 变量名: 数组或集合) {
    - 循环体
  - }

总结

- 注释、标识符、关键字
- 数据类型
- 变量和常量的定义
- 流程控制

Thank You