数据与数据运算

任何程序设计语言，都是由语言规范和一系列开发库组成的。如标准C，除了语言规范外，还有很多函数库；MS Visual C++更是提供了庞大的API和MFC。

Java语言也不例外，也是由Java语言规范和Java开发包组成的。

学习任何程序设计语言，都是要从这两方面着手，尤其是要能够熟练地使用后者。

主要内容

1、Java语言基本元素
2、Java数据类型
3、Java语言结构
4、Java控制语句
5、Java类定义规范
6、Java数组

语言规范

7、Java开发类库组成 → 开发类包

1、Java语言基本元素

- 标识符（Identifier）
- 保留字（Reserved Word ）

1 Java语言基本元素—标识符

- 程序员对程序中的各个元素加以命名时使用的命名记号称为标识符（identifier）包括：类名、变量名、常量名、方法名、…
- Java语言中，标识符是以字母，下划线(_),美元符($)开始的一个字符序列，后面可以跟字母，下划线，美元符，数字。
- 合法的标识符

  identifier　userName　User_Name

  _sys_value　$change
- 非法的标识符

  2mail　room#　class

1 Java语言基本元素—保留字

- 具有专门的意义和用途，不能当作一般的标识符使用，这些标识符称为保留字(**reserved word**)。

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

1 Java语言基本元素—保留字:基本分类

- 数据和返回值类型：**int, void, return……**
- 包/类/接口：**package, class, interface**
- 访问控制：**public, private, protected**
- 分支、循环及循环控制：**if, switch, break,while,for**
- 例外处理：**throw, try, finally**
- 保留词（无含义但不能使用）：**goto, const**

1 Java语言基本元素—保留字:注意事项

- 在Java中，true、false和null都是小写的。区别于C++中大写的TRUE、FALSE和NULL。
- 所有的数据类型所占用的字节数都是固定的，并且和实现无关的，因此在Java中没有sizeof操作符。
- 不必死记这些关键词，当理解每个关键词的含义后，自然就记住了所有的关键词。

2 Java数据类型

- 常量（**Constant**）
- 变量（**Variable**）
- 数据类型（**Data Type**）
  - 基本数据类型（**Primary Data Types**）
  - 复合数据类型（**Composite Data Types**）

2 Java数据类型-常量

Java中的常量值是用文字串表示的，它区分为不同的类型，如整型常量123，实型常量1.23，字符常量‘a’，布尔常量true、false以及字符串常量"This is a constant string."。

与C、 C++不同，Java中不能通过#define命令把一个标识符定义为常量，而是用关键字final来定义，其定义格式为：

final Type varName = value [, varName [=value] …];

- **final int GLOBAL_ACCOUNT=100;**
- **final double PI=3.14159**

2 Java数据类型-变量

- 程序中的基本存储单元，其定义包括变量名、变量类型和作用域几个部分，定义格式为：
  - **Type varName [= value][{, varName [=value]}];**
  - **int n = 3, n1 = 4;**
- **Java中变量的缺省初值(成员变量)都是确定的：**
  - 布尔变量的初值为：**false**
  - 整数变量的初值为：**0**
  - 浮点数变量的初值为：**0.0**
  - 字符型变量的初值为：**'\u0000'**
  - 引用（复合）变量的初值为：**null**

在方法实现中定义的变量必须显式的初始化。

2 Java数据类型-变量的作用域

- 变量必须先定义，后使用。
- 按变量的位置分，变量可分为局部变量和成员变量。
- 作用域：变量的有效范围。
- 局部变量的作用域是从它定义的位置至它所在的块语句的结尾处。
- 在定义局部变量时，如果没有指定初始化表达式，那么局部变量没有值。
- 成员变量的作用域是整个类。
- 在一定的作用域内，变量名必须唯一。
- 方法参数的作用域就是参数所在的这个方法体内。

2 Java数据类型-数据类型

基本类型	数值类型	整型（byte、short、int、long）
		实型（float、double）
	字符型（char）	
	布尔型（boolean）	
复合类型	数组	
	类（class）	
	接口（interface）	

2 Java数据类型-基本数据类型

- 所有基本类型所占的位数都是确定的，并不因操作系统的不同而不同。
- 所有基本类型的关键词都是小写的。

数据类型	所占位数	数的范围
char	16	0 ～65535
byte	8	-2^{7}～$2^{7}-1$
short	16	-2^{15}～$2^{15}-1$
int	32	-2^{31}～$2^{31}-1$
long	64	-2^{63}～$2^{63}-1$
float	32	$3.4e^{-038}$ ～$3.4e^{+038}$
double	64	$1.7e^{-308}$ ～$1.7e^{+308}$

2 Java数据类型-布尔类型(boolean)

- 布尔型数据只有两个值**true**和**false**，且它们不对应于任何整数值

  布尔型变量的定义如：

  boolean b = true;

- 布尔型数据只能参与逻辑关系运算：

  && || == != !

- 示例：

```
boolean b1;
boolean b2 = true;
b1 = !b2;
boolean b = (b1 && b2) != false;
```

2 Java数据类型-字符类型(char)

- 字符型数据代表16位的Unicode字符
- 字符常量是用单引号括起来的一个字符
  - ‘a’ ‘B’ ‘\n’ ‘\u0030’ '\u6211'(我)
- 字符型数据的取值范围为
  - 0~65535 或者说 \u0000~\uFFFF
  - \u0000为缺省值
- 示例
  - char c1;　　\\ 缺省值为0
  - char c2 = '0';　　\\ 赋初值为字符‘0’
  - char c3 = 32;　　\\ 用整数赋初值为空格

2 Java数据类型-字符类型(char)

- 特殊字符的常量表示法：
  - 反斜线（Backslash） ‘\\’
  - 退格（Backspace） ‘\b’
  - 回车（Carriage return） ‘\r’
  - 进纸符（Form feed） ‘\f’
  - 制表符（Form feed） ‘\t’
  - 换行（New line） ‘\n’
  - 单引号（Single quote） ‘\’’
  - Unicode字符 ‘\uHHHH’

2 Java数据类型-整数类型(byte,short, int,long)

- 整型常量

1. 十进制整数

如123，－456，0

2. 八进制整数

以0开头，如0123表示十进制数83，－011表示十进制数－9。

3. 十六进制整数

以0x或0X开头，如0x123表示十进制数291，－0X12表示十进制数－18。

- 整型变量(每一个整型常数默认为int类型，占有32位)

  类型为byte、short、int或long

  byte在机器中占8位，short占16位，int占32位，long占64位。整型变量的定义如：

  int x=123;　//指定变量x为int型，且赋初值为123

  byte b = 8;

  short s = 10;

  long y = 123L; 或 long z = 123l;

- 64位长整数以l或L结尾： 12l, -343L, 0xffffffffL
  - 1L << 32 等于 4294967296L
  - 1 << 32 等于 0
  - 没有以l或L结尾的数字，根据其实际值所属范围，可以被用作byte，short，或int型整数
  - 以l或L结尾的数字，无论其实际值所属范围怎样，都被用作long型整数

- 示例
  - byte b1=0,byte b2 = 127;　　\\ 赋初值
  - byte b3=(byte)(b1+b2);　　\\+运算会提升类型为int
  - short s1 = (short)(b1 + b2); \\ 强制类型转换
  - short s2 = (short)(b1 + 123) \\ 强制类型转换
  - int n = b1 + b2;　　\\ 不需要强制类型转换
  - long l1 = 2343;　　\\ 不需要强制类型转换
  - long l2 = 4294967296L;　\\ 必须用L或l结尾
  - long l3 = 65536*63356; \\ 乘法运算越界，l3为0
  - long l4 = 65536L*63356; \\ l4为4294967296L

2 Java数据类型-浮点数类型，实型(float，double)

- 实型常量

1. 十进制数形式

由数字和/或小数点组成

2. 科学计数法形式

如：123e3或123E3，其中e或E之前必须有数字，且e或E后面的指数必须为整数。

- **32位浮点数形式： 0.23f，1.23E-4f，.18F**

n实型变量(实数的默认类型是double型,占64位)

类型为float或double，float在机器中占32位，double占64位。实型变量的定义如：

- double d1 = 127.0;\\ 赋初值为127
- double d2 = 127;　\\ 赋初值为127
- float f1 = 127.0f;　\\ 必须在数字后加f或F
- float f2 = 4.0e38f; \\ 错误！32位浮点数不能超过3.40282346638528 86e38
- float f3 = (float)d1; \\必须强制类型转换

2 Java数据类型-使用举例

```
public class Assign
{
    public  static  void  main (String args[ ] )
   {
      int  x ,  y ;
      byte  b = 6;
      float  z = 1.234f ;          //1.234默认为double型
      double  w = 1.234 ;
      boolean  flag = true ;
      char  c ;
      c = 'A' ;
      x = 12 ;
      y = 300;
      ......
   }
}
```

2 Java数据类型-数据类型转换

- 自动类型转换

整型、实型、字符型数据可以混合运算。运算中，不同类型的数据先转化为同一类型，然后进行运算，转换从低级到高级：

低--->高

byte,short,char—> int —> long—> float —> double

操作数1类型	操作数2类型	转换后的类型
byte、short、char	int	int
byte、short、char、int	long	long
byte、short、char、int、long	float	float
byte、short、char、int、long、float	double	double

当单目运算的操作数类型是byte、short或char时，自动转换为int类型;否则保持原有类型，不进行转换。

- 数据类型转换必须满足如下规则：

  不能对boolean类型进行类型转换。

  不能把对象类型转换成不相关类的对象。

  在把容量大的类型转换为容量小的类型时必须使用强制类型转换。宽整型强制转换为窄整型(n位)时，只保留最低的n位。

转换过程中可能导致溢出或损失精度

- int i = 8; byte b=(byte)i;
- (byte)255 == -1 (byte)0x5634 == 0x34

- 浮点数到整数的转换是通过舍弃小数得到，非四舍五入
  - (int)23.7 == 23   (int)-45.89f == -45

2 Java数据类型-复合(引用)数据类型

- 在**Java**中，引用（**reference**）指向一个对象在内存中的位置，本质上它是一种带有很强的完整性和安全性限制的指针。
- 当你声明某个类、接口或数组类型的变量时，那个变量的值总是某个对象的引用或者是**null**。
  - 指针就是简单的地址而已，引用除了表示地址而外，还象被引用的数据对象的缩影，还提供其他信息。
  - 指针可以有++、--运算，引用不可以运算。

- 引用型变量只支持有限的逻辑判断：
  - 相等判断（是否同一个对象的引用）： == !=
    - theObject == null
    - otherObject != theObject
  - 类型判断（是否是某个类的实例）： instanceof
    - theObject instanceof ClassName
    - "" instanceof String

    例：Car car=new Car();

    if (car instanceof Car) result=1;

    else result=0;

3 Java语言的结构

- 运算符（**Operator**）
- 表达式（**Expression**）
- 注释（**Comment**）
- 语句（**Statement**）
- 代码段（**Code Block**）
- 作用域（**Scope**）

（1）运算符

- 算术运算符： +，－，*，/，%，++，－－
- 关系运算符： >，<，>=，<=，==，!=
- 布尔逻辑运算符：!，&&，||，&，|
- 位运算符： >>，<<，>>>，&，|，^，~
- 赋值运算符： =，及其扩展赋值运算符如+=，－=，*=，/=等。
- 条件运算符： ？：

- 其它
  - 分量运算符 .，
  - 下标运算符 []，
  - 实例运算符 instanceof，
  - 内存分配运算符 new，
  - 强制类型转换运算符 (类型)，
  - 方法调用运算符 ()
  - …

由于数据类型的长度是确定的，所以没有长度运算符sizeof。

（2）表达式

表达式是由操作数和运算符按一定的语法形式组成的符号序列。

- 一个常量或一个变量名字是最简单的表达式，其值即该常量或变量的值；
- 表达式的值还可以用作其他运算的操作数，形成更复杂的表达式。

例：x　　num1+num2　　a*(b+c)+d

　　3.14　　x<=(y+z)　　x&&y||z

1　%可以操作于浮点数：7.6%2.9=1.8

2　++、--不进行运算符的提升，运算结构的类型与变量的类型相同

3　==、 !=可用于任何类型的比较。布尔、数值、引用和浮点类型

4　&&、||为快速逻辑与、快速逻辑或。运算符左边表达式即可确定与、或结果，则右边表达式将不被计算。

5(1)用int变量a,b,c表示三个线段的长，它们能构成一个三角形的条件是：任意两边之和大于第三边。该条件的Java布尔表达式为：

(a + b) > c && (b + c) > a && (a + c) > b

或 !((a + b) <= c || (b + c) <= a || (a + c) <= b)

(2)用int变量y，表示年号y是闰年：

y%400==0 || y%4==0 && y%100!=0

(3)用int变量age存放年龄，boolean变量sex存放性别（true为男），表示20岁与25岁之间的女性：

age>=20 && age <= 25 && !sex

&&、||运算符举例

```
class Example0212
{
    int x = 0, y = 100;
    public void method()
    {   boolean test1 = x!=0 && y/x>10;
        System.out.println("test1=" + test1);
        boolean test2 = x==0 || y++>100;
        System.out.println("test2=" +test2 + " y=" + y);
  }
}
```

```
class TestExample0212
{
    public static void main(String args[])
    {     Example0212 o = new Example0212();
          o.method();
   }
}
```

```
Java TestExample0212
test1=false
test2=true   y=100
```

（3）运算符的优先次序

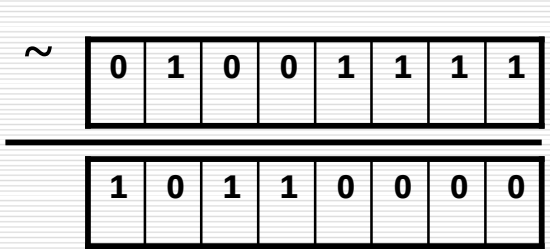
1）	. , [] , ()	9）	&
2）	++ , -- , ! , ~ , instanceof	10）	^
3）	new (type)	11）	|
4）	* , / , %	12）	&&
5）	+ , -	13）	||
6）	>> , >>> , <<	14）	?:
7）	> , < , >= , <=	15）	= , += , -= , *= , /= , %= , ^=
8）	== , !=	16）	&= , |= , <<= , >>= , >>>=

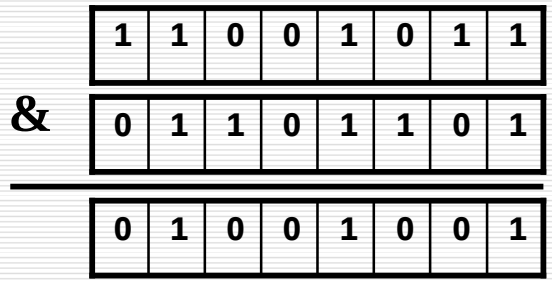
位运算符

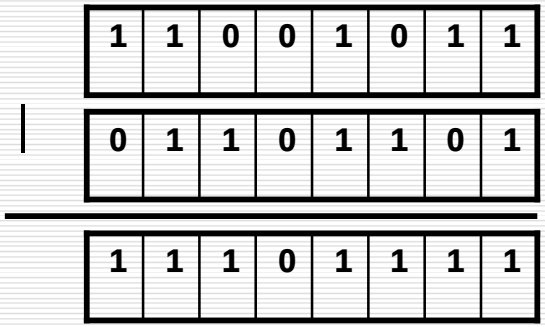
位运算符功能

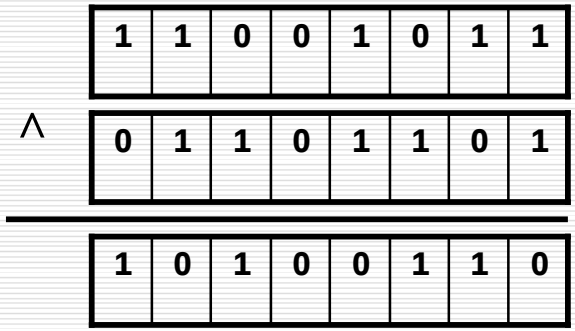
~ 取反　　&按位与　　|按位或　　^按位异或

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
- 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

移位运算符应用举例

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

负数=正数取反+1

表达式的计算次序

- 从左到右，并尊重运算符优先级和结合性的原则进行。
- 结合性:确定同级运算符的运算顺序。运算符有左结合性和右结合性两种。左结合性指的是从左向右使用运算符。例如二元算术运算符具有左结合性，计算a + b - c时；而右结合性是从右向左使用运算符。例如，赋值运算符具有右结合性，计算a = b = c时。

  int i=3, j=i++, k=i+i++;

  i,j,k各是多少？

3 Java语言的结构—注释

- 举例：
  - // 单行注释，简单的解释语句含义.
  - /* 多行注释，用来说明更多的内容，包括算法等.
    ……
    */
  - /** Java文档注释，可以通过javadoc生
    * 成类和接口的HTML格式的帮助文档.
    * 这种注释有其特殊的格式（参见相关文档）
    */

3 Java语言的结构—语句和代码段

- 一个由分号（;）结尾的单一命令是一条语句（Statement），一条语句一般是一行代码，但也可以占多行代码。

```
int a = 1; // 变量定义及初始化语句
```

- 用大括号（{…… }）围起来的多条语句构成一个代码段（Code block）；同时代码段也大括号前的一些修饰性描述：

```
class Aclass {
  for (int i=0; i<=1; i++) {
     ……
  }
}
```

3 Java语言的结构—作用域

- 作用域决定了变量可使用的范围
  - 全局变量（**Global variables**）：变量可以在整个类中被访问；
  - 局部变量（ **Local variables** ）：变量只能在定义其的代码段中被访问。
- 作用域规则：在一个代码段中定义的变量只能在该代码段或者该代码段的子代码段中可见。
- 使用局部变量比使用全局变量更安全。

```
class Scoping
{    int x = 0;
     void method1()
     {    int y;
          y = x;  // OK. method1可以访问y.
     }
     void method2()
     {    int z = 1;
          z = y;  // Error. y 在method2的作用域之外定义
     }
}
```

包装类（wrapper class）

Java语言中专门提供了所谓的包装类（wrapper class）。这些类将基本类型包装成类。

使用包装类的方法与其他类一样，定义对象的引用、用new运算符创建对象，用方法来对对象进行操作。

基本类型	包装类
byte	Byte
short	Short
int	Integer
long	Long
char	Character
float	Float
double	Double
boolean	Boolean
void	Void

例:包装类Integer类常用方法的使用

```
2001 的二进制是:
11111010001
2001 的八进制是: 3721
2001 的十六进制是: 7d1
```

```
class UseWrapper {
  public static void main(String args[]) {
    int num = 2001;
    System.out.println(num + " 的二进制是: " +
    Integer.toBinaryString(num)); //转换十进制数为二进制数
    System.out.println(num + " 的八进制是: " +
    Integer.toOctalString(num)); //转换十进制数为八进制数
    System.out.println(num + " 的十六进制是: " +
    Integer.toHexString(num));   //转换十进制数为十六进制数
```

```
    Integer iobj = Integer.valueOf("123");
    Integer iobj1 = new Integer(234);
    Integer iobj2 = new Integer("234");
    int i = iobj.intValue();
    System.out.println("iobj = " + iobj);
    System.out.println("i = " + i);
    System.out.println("iobj1==iobj2 ? " + iobj1.equals(iobj2));
}
}
```

```
iobj = 123
i = 123
iobj1==iobj2 ? true
```

输入输出初步

- System.out.println:在程序中可以将常量、变量或表达式的值输出到屏幕。
- 参数可以是char,byte,int,boolean，float,double,String,char[]（字符数组）和Object（对象）类型的，各种类型的数据转换成相应的字符串类型输出。
- print和println方法的区别
- 用“+”运算符连接多个输出项为一项进行输出。

设有 int i = 10,j = 20;

System.out.println("i=" + i + " j=" + j);

// 输出为i=10 j=20

输入

System.in.read():

可用于从键盘输入整数值在0-255之间的byte值（单个字符）。

不论从键盘输入何种类型的数据，Java从键盘接收数据都是以字符的形式进行，再根据实际数据的需要进行类型转换。

例：从键盘输入一个字符，并输出它在Unicode字符集中的前一字符和后一字符

```
import java.io.*;  // 引入java.io包
class CharDemo {
   public static void main(String args[]) throws Exception {
      char ch,ch1,ch2;
      System.out.println("enter a char:");
      ch=(char)System.in.read();  //read返回int型,即返回字符的ASC码
      ch1=(char)(ch-1);
      ch2=(char)(ch+1);
      System.out.println(" : " + ch1);
      System.out.println(" : " + ch2);
   }
}
```

从键盘输入一个整数和实数，并输出它们的和

```
import java.io.*;  // 引入java.io包
class InputDemo {
public static void main(String args[]) throws Exception {
    //用标准输入System.in创建一个 BufferedReader
   BufferedReader br =
   new BufferedReader(new InputStreamReader(System.in));
   System.out.print("请输入一个整数： ");
   String str = br.readLine();      // 输入字符行存入字符串
   int i = Integer.parseInt(str);   // 转换字符串为整型数据
```

```
    System.out.print("请输入一个实数： ");
    str = br.readLine();
    float f = Float.parseFloat(str);  //转换字符串为实型数据
    System.out.print("它们的和是： "+(i+f));
}
}
```

包装类的类型转换

1 Integer通过parseInt等方法将字符串转换为数值，同样不同的包装类可将字符串转为相应基本类型

public static int parseInt(String s)

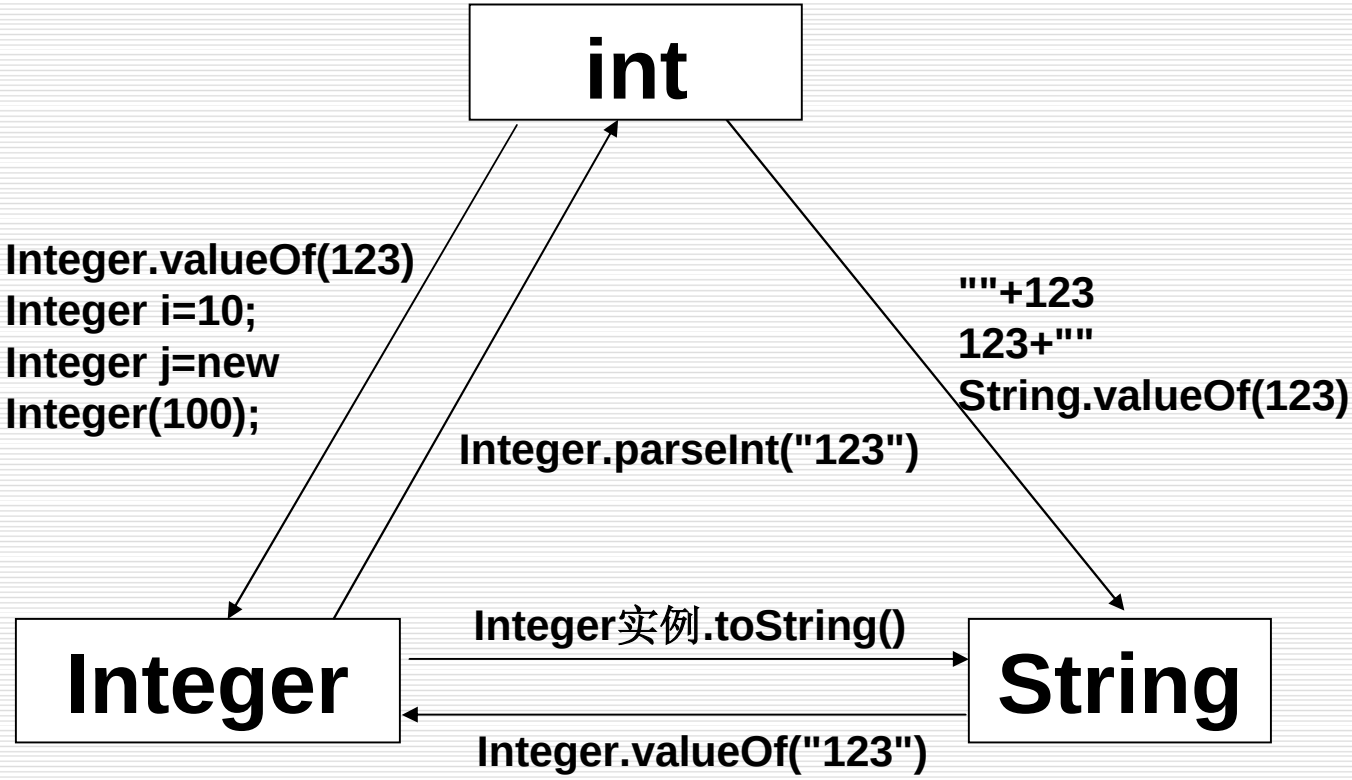
public static float parseFloat(String s)

public static double parseDouble(String s)

public static long parseLong(String s)

public static byte parseByte(String s)

类型转换:基本类型，包装类，字符串



parseInt等方法举例

```
String s="12.34";
float    y1=Float.parseFloat(s);
double y2=Double.parseDouble(s);
int       y3=Integer.parseInt(“1234”);
long     y4=Long.parseLong(“12345678”);
byte     y5=Byte.parseByte(“123”);
```

2 通过toString方法将数值转换为字符串

```
float  x1=12.34f;
double x2=123.4;
int  x3=1234;
long x4=12345678;
byte x5=123;
String ss1=Float.toString(x1);
String ss2=Double.toString(x2);
String ss3=Integer.toString(x3);
String ss4=Long.toString(x4);
String ss5=Byte.toString(x1);
```

3 通过valueOf方法将字符串转换为包装类型

int i=Integer.valueOf(str).intValue()

Integer.valueOf(str)：返回一个整型(Integer)对象，该对象含有字符串str表示的整数值。

对象变量.intValue()：返回整型对象的整数值。

同理，有：

String s1="3.1415926";

float y1=Float.valueOf(s1).floatValue();

返回Float类型

4 通过String类的valueOf方法将数值转换为字符串

```
float  x1=12.34f;
double x2=123.4;
int    x3=1234;
long   x4=12345678;
String s1=String.valueOf(x1);
String s2=String.valueOf(x2);
String s3=String.valueOf(x3);
String s4=String.valueOf(x4);
```

5 包装类总结

以Integer为例，其他包装类相似。

Integer常用静态方法：

```
Integer.parseInt("1234");
Integer.toString(1234);
```

Integer常用实例方法：

```
Integer it=new Integer("1234");
Integer it=new Integer(1234);
int i1=it.intValue();
```

下课！ 

Thank you!