java_2

内容：
标识符、注释、关键字、数据类型、常量和变量、方法,运算符

Java注释

1.类注释使用：

```
/**
 *
 * @author djy
 *
 */
```

2.单独注释代码使用

1).**/多行注释**

2).//单行注释

Java的数据类型

Java语言是**强类型**语言，意思就是每个变量和每个表达式都有一个在编译时就确定的类型，也就是说，所有的变量必须先声明，后使用，变量必须显式申明类型。

程序概述

软件是什么？其实软件就是用来**处理数据**的程序。可以把软件分为两个部分，一个部分是“数据”；另一个部分是“数据处理的逻辑”。

程序 = 数据结构 +算法

如某东用户是数据，商品也是数据。那么什么是数据处理逻辑呢？购买商品的流程就是数据处理逻辑[解决一类或是一个事情就叫方法]。

程序开发离不开数据，程序就是用来处理数据。

通常数据是会发生变化的，而数据的处理逻辑是不发生变化的。如商品的数量种类是会变化的，用户的数量也是会变化的，但购买的流程逻辑通常不会变化。

Java 中有两大数据类型:

```
- 内置数据类型
    Java语言提供了八种基本类型
- 引用数据类型
    对象、数组都是引用数据类型等。
    自定义类引用如：
    User user= new User("djy")。
```

数据类型

在生活中数据是有类型的，例如人的姓名是**字符串类型**，人的年龄是正**整数类型**。Java中数据也是有类型的（任何数据必然会有类型），Java中数据的类型,如：

```
String name="djy";
String age="35";
.......
```

基本数据类型[有**8种基本类型**]包括boolean类型和数值类型。数值类型有整数类型和浮点类型。基本数据类型是Java语言中**内置**的类型，有时候也把它们分为**整数**类型、**小数**类型、**字符**类型、**布尔**类型。有时候也把char称为**字符**型，实际上字符型也是一种整数型。基本类型是最简单、最基础的类型。

引用数据类型是强大的数据类型，它是基于基本数据类型创建的。

基本类型	引用类型
int（使用最多）	Integer
double（使用最多）	Double
boolean（使用最多）	Boolean
float	Float
char	Character
long	Long
short	Short
byte	Byte

byte是“字节”类型，代表一个8位二进制，也就是一个字节。

字符型char：表示一个字符，如('a'，'A'，'0'，'中')

整数常量默认是int类型，小数常量默认是double类型。

布尔boolean类型，只有两个值true与false，用于表示逻辑上的“真”或“假”，主要用来流程的控制。

```
double>float
long>int>short>byte;
```

引用类型包括类、接口和数组类型，还有一种特殊的null类型。所谓引用数据类型就是对一个对象的引用，对象包括实例和数组两种。

常量

Java中的数据量分为常量和变量。

常量就是**不变**的数据量，如100数字。

```
1). 整数类型：
      如数字22...
2). 小数类型
      如1.0、-5.23、6.66等
3) 布尔类型(boolean)
      true、false
4).字符类型(char)
    如'z'，'H'，'中'
    字符必须使用单引号''包裹，并且其中只能且仅能包含一个字符。
5). 字符串类型
    字符串String类型是一种引用类型
        如“编写Java”，“0123”，“”，“null”
    字符串必须使用英文双引号""包裹，可以包含0~N个字符
```

如System.out.println(“Hello World!”)的圆括号中放置的内容就是一个字符串，也可以把圆括号中的字符串换成其它的字符串，不只是可以换成其他字符串，还可以把圆括号中的字符串换成其他类型的常量，如：

```
/*
* 常量
*/
public class Const {
    public static void main(String[] args) {
        //整数类型
        System.out.println(100);
        System.out.println(12345);
        //小数类型
        System.out.println(3.15);
        //字符类型
        System.out.println('z');
        System.out.println('H');
        System.out.println('中');
        //布尔类型
        System.out.println(true);
        System.out.println(false);
        //字符串类型
        System.out.println("编写Java代码");
        System.out.println("0123");//数字加了双引号则为字符串
    }
}
```

Java标识符

Java所有的组成部分都需要定义**名字**（名称）。**类名**、**变量名**以及**方法名**都被称为标识符。

*：尽可能使用英文单词或是英文缩写。

Java标识符注意命名规范：

- 所有的标识符都应该以字母（A-Z或者a-z），美元符（$）、或者下划线（_）开始
- 首字符之后可以是字母（A-Z或者a-z），美元符（$）、下划线（_）或数字的任何字符组合
- 关键字不能用作标识符
- Java 关键字，不能用于常量、变量、和任何标识符的名称

注：所有标识符（变量名，类名，方法名等），使用**英文单词**或是英文单词**缩写**（**关键字**除外）几乎不会出现问题

- 标识符是大小写敏感
- 合法标识符举例：age、$salary、_value、__1_value
- 非法标识符举例：123abc、-abc

最好的命令法则是，**驼峰式**命名，如：UserList，addUser,saveUser

*：类名称大驼峰命名首字母大写,如HospitalInfo,类里的变量名称则是小驼峰命名hospitalNname

变量定义

声明变量定义语法格式：

```
数据类型  变量名  =  数据值；
int       num    =  100;//赋值100
```

int是**数据类型**，指定num名称的变量只能存储100整数。

*：变量定义后可以不赋值，使用时再赋值，不赋值无法使用(若使用，则没有任何的输出)

```
public static void main(String[] args) {
    int num;
    num = 20; //为num赋一个数字值为20
    System.out.println(num);//读取num变量中的值，再输出到控制台
}

//变量使用时有作用域的限制。
public static void main(String[] args) {
    int num = 20;
    {
       int y = 20;
    }
    System.out.println(num);//读取num变量中值，再输出到控制台
    System.out.println(y);//读取y变量中的值失败，失败原因是找不到y变量（作用域范围），因为超出了y
变量作用范围，所以不能使用y变量
}
```

```
//x变量不可以重复定义（变量名称必须是唯一）
public static void main(String[] args){
      int x = 10;
      double x = 5.5;//类型转换失败，编译失败，变量重复定义
 }
```

*：变量名称定义**唯一**，必须先定义后使用。

数据类型转换

不同类型的变量假如在一起运算也是可行的，但要先进行类型转换再运算。

转换的过程中，数据遵循一个原则：

```
范围小的数据类型值（如byte），可以直接转换为范围大的数据类型值（如int）
范围大的数据类型值（如int），不可以直接转换为范围小的数据类型值（如byte）
```

各种数据类型按照数据范围从小到大依次列出：

```
byte -> short -> int -> long -> float -> double
```

关于数据类型转换有：

1) **自动类型转换**

```
表示范围小的数据类型转换成范围大的数据类型，这种方式称为自动类型转换
自动类型转换格式：
   范围大的数据类型 变量 = 范围小的数据类型值；
如：
        double d = 1000;
   或
        int i = 100;
        double d2 = i;
```

2)**强制类型转换**

```
表示范围大的数据类型转换成范围小的数据类型，这种方式称为强制类型转换
强制类型转换格式：
    范围小的数据类型  变量 =（范围小的数据类型）范围大的数据类型值;
如：
    int  i = (int)6.12;    //i的值为6
或
    double  d = 5.12;
    int  i2 = (int)d;      //i2的值为5
```

面向过程(方法)

Java方法是语句的集合，主是为执行一个功能。

方法是解决一类问题的步骤的有序组合
方法包含于类或对象中
方法在程序中被创建，在其他地方被引用(必须先创建对象,使用new关键字)

方法优点

1．使程序变得更简短而清晰
2．有利于程序维护
3．可以提高程序开发的效率
4．提高了代码的重用性（当开发变得多，代码功能增多时，如果有有重复功能的代码，则学会重构代码或是抽取公共的代码方法，以便于更多的地方调用）

方法命名规则

如果类名： 首字母必须大写

方法名：第一个单词首字母小写，方法的名字的第一个单词应以小写字母作为开头，后面的单词则用大写字母开头写，如：addUser，updateUser ，deleteUser , selectUser , findUser , getUserById

方法定义

定义一个方法伪代码语法：

```
修饰符 返回值类型 方法名(参数类型 参数名){
    ...
    方法体
    ...
    return 返回值;
}
```

方法包含一个方法头和一个方法体。下面是一个方法的所有部分：

1).修饰符：修饰符，这是可选的，告诉编译器如何调用该方法。定义了该方法的访问类型。
2)返回值类型：方法可能会返回值。returnValueType 是方法返回值的数据类型。有些方法执行所需的操作，但没有返回值。在这种情况下，returnValueType 是关键字void（无返回值）。
3).方法名：是方法的实际名称。方法名和参数表共同构成方法签名。
4).参数类型：参数像是一个占位符。当方法被调用时，传递值给参数。这个值被称为实参或变量。参数列表是指方法的参数类型、顺序和参数的个数。参数是可选的，方法可以不包含任何参数。
5).方法体：方法体包含具体的语句，定义该方法的功能。

```
public static int outNum(int num1, int num2) {//static如果不写查看方式调用
   int result = num1 + num2;
   return result;
}
```

方法调用

一般使用main方法（main也是个方法）调用前面的outNum方法，执行输出打印查看。

Java 支持两种调用方法的方式，根据方法是否**返回值**来选择，当程序调用一个方法时，程序的控制权交给了被调用的方法。当被调用方法的返回语句执行或者到达方法体闭括号时候交还控制权给程序。当方法返回一个值的时候，方法调用通常被当做一个值。

关键字

Java关键字是电脑语言里事先定义（专门被电脑在某特殊处使用过的），有特别意义的标识符，有时又叫保留字，还有特别意义的变量。Java的关键字对Java的编译器有特殊的意义，他们用来表示一种数据类型，或者表示程序的结构等，**关键字不能**用作变量名、方法名、类名、包名和参数。

关键字**	**含义**
class	声明一个类
public	一种访问控制方式：共用模式
static	表明具有静态属性
void	声明当前成员方法没有返回值
.....	.....

Java修饰符定义

```
- 访问控制修饰符 ： public（公共的） ， protected（受保护）， private（私有的）...
    使用最多：
        public：定义公共类或是方法使用
        private：定义属性名称使用最多
- 非访问控制修饰符 ： final（最终的）， static（静态的）...
     final，static一般两个结合使用，用在不可变且固定的属性值
     也可使用在类或是类方法中
```

```
public class DB {

    public static String DBUSERNAME="root";
    public static final String PASSWORD="root123";

    public static void main(String[] args) {
        System.out.println(DBUSERNAME + PASSWORD);
    }

}
```

运算符

运算符是一种特殊的符号，用以表示数据的运算、赋值和比较等。Java语言使用运算符将一个或多个操作数连缀成执行性语句，用以实现特定操作功能。

Java语言中的运算符可以分为如下几种：

```
算术运算符
赋值运算符
比较运算符
逻辑运算符
类型相关运算符
```

算术运算符

运算符是用来计算数据的符号。数据可以是常量，也可以是变量。被运算符操作的数我们称为操作数。

算术运算符[+加，-减,*乘, /除,%取余,++自增，--自减]最常见的操作就是将操作数参与数学计算

使用算术运算符时：

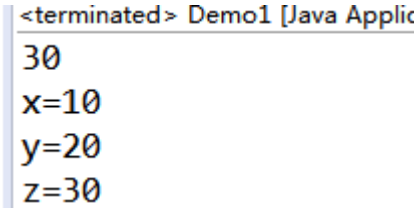
```
a).加法运算符在连接字符串时要注意，只有直接与字符串相加才会转成字符串
b). 除法“/”当两边为整数时，取整数部分，舍余数。当其中一边为浮点型时，按正常规则相除
c). “%”为整除取余符号，小数取余没有意义结果符号与被取余符号相同
d). 整数做被除数，0不能做除数，否则报错（使用用此方法来做异常测试）
```

代码:

```
/*
 * 算术运算符
 */
public class OperatorDemo1 {
    public static void main(String[] args) {
        /*
         * 常量使用算数运算符
         */
        System.out.println(10+20);

        /*
         * 变量使用算数运算符
         */
        int x = 10;
        int y = 20;
        //"+"作为加法运算使用
        int z = x + y;
        //"+"作为连接字符串使用
        System.out.println("x="+x);
        System.out.println("y="+y);
        System.out.println("z="+z);
    }
}
```

运行结果：



算数运算符++、--的使用

算数运算符不会改变参与计算的变量值。而是在原有变量值不变的情况下，计算出新的值。但是有些操作符会改变参与计算的变量的值，比如++，--。

看一段代码：

```
int a = 3;
int b = 3;
a++;//后加
b--;//后减
System.out.println(a);
System.out.println(b);
```

输出结果：a值为4，b值为2；

```
说明a的原有值发生改变时，在原有值的基础上自增1；b的原有值也发生改变，在原有值的基础上自减1：
    ++运算符，在原有值的基础上自增1
    --运算符，在原有值的基础上自减1
```

再看一段代码：

```
int a = 3;
int b = 3;
++a;//前加
--b;//前减
System.out.println(a);
System.out.println(b);
```

输出结果:a值为4，b值为2；

前++，后++：

```
        int c = 9;
        System.out.println(++c);
        System.out.println(c++);
        System.out.println(c);
        System.out.println(--c);
```

赋值运算符

赋值运算符就是为变量赋值的符号，如下：

运算符	运算规则	范例	结果
=	赋值	int a=2	2
+=	加后赋值	int a=2，a+=2（等同于a=a+2）	4
-=	减后赋值	int a=2，a-=2(等同于a=a-2)	0

注意：如+=这样形式的赋值运算符，会将结果自动强转成等号左边的数据类型

赋值运算符:

```
/*
* 赋值运算符
* +=, -=, *=, /=, %= :
* 上面的运算符作用：将等号左右两边计算，会将结果自动强转成等号左边的数据类型,再赋值给等号左边的
* 注意：赋值运算符左边必须是变量
*/
public class NumDemo2 {
    public static void main(String[] args) {
        int x = 10;
        x += 20;// 等同 x=x+20;
       System.out.println(x);
   }
}
```

比较运算符

比较运算符，又叫关系运算符，它是用来判断两个操作数的大小关系及是否相等关系的，结果是布尔值true或者false。

运算符	运算规则	范例	结果
==	相等于（开发使用最多）	4==3	False
!=	不等于（开发使用最多）	4!=3	True
<	小于	4<3	False
>	大于	4>3	True
<=	小于等于	4<=3	False
>=	大于等于	4>=3	True

注意一下：

> 赋值运算符的 = 符号与比较运算符的 == 符号是有区别的，如下：
>
> 赋值运算符的 = 符号，是用来将 = 符号右边的值，赋值给 = 符号左边的变量
>
> 比较运算符的 == 符号，是用来判断 == 符号 左右变量的值是否相等的（两值相比较逻辑判定if使用最多）

代码编写：

```
int a = 3;
int b = 4;

System.out.println( a=b );
System.out.println( a==b );

//代码输出的结果:第一个值为4，第二个值为false。

//如果是字符比较则不同
String loginName = "张三";
String loginName2 ="李四" ;

System.out.println( loginName.equals(loginName2));//使用equals比较才正确
//System.out.println( loginName=loginName2 );
System.out.println( loginName==loginName2 );

//== 与 equals区别
String s1 = new String("张三");
System.out.println(s1.hashCode());
String s2 = new String("张三");
System.out.println(s2.hashCode());
String s3 = "张三";
System.out.println(s3.hashCode());

System.out.println(s1==s2);//false,不是同一个对象
System.out.println(s1==s3);//false,不是同一个对象
System.out.println(s1.equals(s2));//true,equals比较是值
System.out.println(s1.equals(s3));//true,equals比较是值

String num1 = new String("111");
int num2 = 111;
```

```
System.out.println(num1.equals(num2));//false,不是同一个类型对象，一个String ， 一个int
```

*：== 与 equals 的区别

逻辑运算符

逻辑运算符，它是用于布尔值进行运算的，运算的最终结果为布尔值true或false。

运算符	运算规则	范例	结果
!	非 (开发中使用多)	!true	Flase
&&	与 (开发中使用多)	false&&true	False
||	或 (开发中使用多)	false||true	True

代码编写：

```
boolean b = 100>10;//输出：true
boolean b2 = false;

System.out.println(b&&b2); //打印结果为 false
System.out.println(b||b2); //打印结果为 true
System.out.println(!b2); //打印结果为 true，!取反
System.out.println(b && 100>10); //打印结果为 true
```

运算符的结果说明：

```
&&（与，且）：
        参与运算的两边数据，有false，则运算结果为false（两边一致方可）
||（或）：
        参与运算的两边数据，有true，则运算结果为true
!（逻辑非）：
        参与运算的数据，原先是true则变成false，原先是false则变成true（取反）
```

三元（或三目）运算符

伪代码格式：

```
（条件表达式）？表达式1：表达式2；
```

三元运算符运算规则：

```
先判断条件表达式的值，若为true，运算结果为表达式1；若为false，运算结果为表达式2
```

三元运算符的使用：

方式一：

```
    System.out.println(3>2?"正确":"错误");
```

方式二：

```
int a = 3;
int b = 4;
System.out.println(a==b?"相等" : "不相等");


String loginName = "张三";
String loginName2 ="李四" ;
System.out.println(loginName.equals(loginName2)?true:false);

System.out.println("".equals(loginName)?"空值":loginName);
```