Java简介dos简介

#一、基本语法

语句与代码块

1. JAVA是以;为语句的分隔符号
2. JAVA允许代码元素之间出现空白
3. 一对大括号所包含的语句是一个语句块

注释方法

注释是代码必要内容，便于回看代码

1. //注释一行
2. / .... /注释多行
3. / * .... /文档注释 文档注释的保存

```
javadoc -d D:\codeJAVA\JAVAproject\temp -author -version comment.java
```

在终端使用javadoc +"-d D:\codeJAVA\JAVAproject\temp"+"-author -version"+"comment.java"
javadoc+-d 输出文档位置+输出标签信息+对应代码位置

标识符命名规则与规范

1. JAVA中对标识符严格区分大小写，长度没有限定
2. 标识符不能是JAVA中的关键字和保留字
3. 数字不能作为标识符的第一个字符
4. 标识符只能是字母数字与$和_
5. 对于常量一般使用大写字母
6. 对于变量名方法名可以使用驼峰命名法

转义字符

1. \t 制表符实现对齐功能
2. \n 换行符
3. \ \ :输出一个\
4. \ " : 输出一个"
5. \ ' :输出一个‘
6. \r:输出回车

#二、变量与运算

变量是程序的基本组成部分，变量由类型，变量名称和值组成
定义一个变量则在内存中申请一个空间，而变量则是指向该空间，赋值一个变量则是将该值放在变量指向的空间中。

基本数据类型

```
1. 逻辑型 boolean[1]:
 boolean truth=true;
 不可以用0或者非零的数代替
 注意：boolean类型默认值为false
2. 文本型 char[2] 和 String:
 char mychar="Q";
 String greeting="Good morning!\n";//String不是基本类型
 而是一个类，有两个类可以表示字符串String和StringBuffer
 字符常量通常使用单引号括起来的单个字符，可以直接赋值一个整数，会根据编码输出字符，每一个数字对应unicode编码都是字符
3. 整型 byte[1] short[2] int[4] long[8]: 默认int
 如果一个数超过了计算机的表达范围则称为溢出，对于整形的最大  值加一则上溢变为最小值，最小值减一则下溢变为最大值
 **从大内存整型数赋值到小内存整型变量可能会有损失，执行的时候会报错**
 byte mybyte=100;
 short myshort=20000;
 int myint=1000000;
 long mylong=10000000000L //long类型需要在后面加上L或l
4. 浮点型 double[8] float[4] 默认double
 float myfloat=3.14f //float类型需要在后面加上F或者f
 double mydouble=3.1415926;
 double类型数赋值到float变量可能会有损失，执行的时候会报错
 科学计数法：  小数+e/E+指数
 double精度更高
 浮点数计算的时候得到的是近似值，当计算的时候对小数的结果进  行相等判断的时候要谨慎，应该是两个数差值绝对值在某个精度范  围内进行判断
```

System.out.println输出的时候+的作用，如果均为数值型则做加法运算，如果从左到右遇到字符串则进行拼接运算

基本数据类型转换

char->int->long->float->double
byte->short->int->long->float->double
赋值的时候先判断范围，符合范围则判断类型
低精度能向高精度转换，byte和short不能自动转为char，需要先转换为int类型，这两个类型

参与运算后自动转换为int类型
参与运算后表达式自动转换为操作数中最大的类型

强制类型转换

强制类型转换一般是高精度转化为低精度的过程
小数取整只保留整数，并且超出范围会溢出
强转步骤是在需要强转的数据前面加上（强转类型）
char类型可以保存int整数，但是不能保存int变量值，需要强转

变量必须先声明再使用
JAVA是强类型语言，必须指定数据类型

常量介绍

定义常量的时候需要加上final

```
final int a=10;
```

常量在赋值给低精度类型变量的时候，只要常量的范围不超过该类型的范围时候，不会报错

```
public class final01{
    public static void main(String[] args){
    final int A=10;
    byte b=A;
    System.out.println(A);
    }
}
```

基本数据类型与字符串的转换

基本数据类型转换字符串

```
int n1=100;
double d1=4.5;
String s1=n1+"";
String s2=d1+"";
```

字符串转换基本数据

```
String s1="100";
int n1=Integer.parseInt(s1);
String s2="4.5"
```

```
double d1=Double.parseDouble(s2);
```

字符串中取出字符

```
String s3=123+"";
char a=s3.charAt(0);
```

复合数据类型

对于日期相关的三个变量day month year进行封装，使用class创建一个日期类

```
class MyDate{
int day;
int month;
int year;
}
MyDate a;
a.day=30;
a.month=12;
a.year=2025;
```

基本类型变量与引用类型变量

1. 基本类型变量包含单个值
2. 引用类型变量的变量值是指向内存空间的地址，所指的地址中保存着另外的变量

```
//基本类型变量
int a;
a=12;
//引用类型变量
MyDate today;
today=new Date();
```

3. 引用赋值

```
MyDate a,b;
a=new MyDate();
b=a;
```

使用上述的操作a和b就同时指向new开辟的数据空间

变量作用域

变量在同一个作用域不能重名

1. 局部变量在声明他的代码块{}中有效
2. 类成员变量
3. 方法参数的作用域就是其所在的方法
4. 异常处理器参数

```
public class MyClass {
    private int privateVar;      // 只能在当前类中访问
    int defaultVar;              // 默认访问权限，只能在同一包内访问
    protected int protectedVar; // 可以在子类中访问
    public int publicVar;       // 可以在任何地方访问

    public MyClass() {
        // 构造函数中可以访问所有成员变量
        privateVar = 1;
        defaultVar = 2;
        protectedVar = 3;
        publicVar = 4;
    }

    public void printValues() {
        // 该方法中可以访问所有成员变量
        System.out.println(privateVar);
        System.out.println(defaultVar);
        System.out.println(protectedVar);
        System.out.println(publicVar);
    }
}
```

#三、运算符与进制转换

基本的算数运算符

名称	符号	细节描述
加	+	
减	-	
乘	*	

名称	符号	细节描述
除	/	整数除以整数的时候会自动取整
取余	%	取模的本质是a%b=a-a/b*b
自加	++	独立存在则是自增1，前后位置完全等价；表达式使用，前++是先自增后赋值，后++是先复制后自增
自减	--	

赋值运算符复合形式，a+=b，a-=b，a*=b，a/=b等价于a=a+b,a=a-b,a=ab,a=a/b
并且运算过程也可以发生类型转换

关系运算符

关系运算符结果都是boolean型，true或者false

运算符名称	符号
==	相等于
!=	不等于
>	大于
<	小于
>=	大于等于
<=	小于等于

逻辑运算符

运算符名称	符号	介绍
逻辑与	&	a与b均为真时，结果为真，否则为假
逻辑或	|	a与b有一个为真时结果为真，否则为假
逻辑异或	^	a与b不同的时候结果为真，否则为假
取反	!	颠倒真假
短路与	&&	a&&b当a为假的时候直接返回假，跳过b的运算
短路或	||	a||b当a为真的时候直接返回真，跳过b的运算

短路与和**短路或**比**逻辑与**和**逻辑或**的效率更高

三元运算符:条件表达式？表达式1:表达式2，条件表达式为真时候运行表达式1，当条件表达式为假的时候运行表达式2
表达式1和表达式2的运算结果要可以赋值给接收的变量，仍然要考虑精度，如果两个表达式

返回的精度不一致，则最后返回的时候按照最大精度返回。三元运算符是一个整体
三元运算符等价于if ... else ... 语句

运算符优先等级:

优先级	符号
1	（）
2	++，--
3	*，/，%
4	+，-
5	==，!=
6	&
7	^
8	|
9	&&
10	||
11	... ? ... : ...
12	=,*=,/=,+=,-=

有括号先算括号内的，优先级相同则从左往右运算
优先级排序：

1. (),{}等
2. 自加自减
3. 算数运算符
4. 位移运算符
5. 比较运算符
6. 逻辑运算符
7. 三元运算符
8. 赋值运算符

接收输入

使用Scanner类,首先需要导入所在类，导入操作也是一个语句需要加分号

```
/*
import java.util.Scanner;
public class input{
public static void main(String[] args){
```

```
Scanner myScanner=new Scanner(System.in);
System.out.println("请输入名字：");
String name=myScanner.next();
System.out.println("请输入年龄：");
int age=myScanner.nextInt();
System.out.println("请输入薪水：");
double salary=myScanner.nextDouble();
System.out.println("信息如下："+"\n"+"名字"+"\t"+"年龄"+"\t"+"薪
水"+"\n"+name+"\t"+age+"\t"+salary);
}
}
```

进制介绍

进制类型	进制范围	进制表示方法
二进制	0，1	int n1=0b1010;//开头为0b
八进制	0-7	int n2=01010;//开头为0
十进制	0-9	int n3=1010;//正常写法
十六进制	0-9和A(10)-F(15)	int n4=0x10101;//开头为0x

十进制转化为二进制，八进制，十六进制，对目标数不断除以对应进制数，最后将余数反过来按顺序排列就是对应的进制数
最后得到的进制数应该是八的倍数

二进制转化为八进制，三位一组；二进制转化为十六进制，四位一组
八进制转化为二进制，一个数分解为三位二进制；十六进制转化为二进制，一个数分解为四位二进制

位移运算符

原码，反码与补码：对于二进制最高位是符号位，0表示正数，1表示负数，正数的原码，反码与补码均一样，负数的反码等于符号位不变其他位取反，补码等于反码加一，**计算机运算的时候都是用补码运算的**

符号	名称	解释
~	按位取反	1变0,0变1；对补码运算
&	按位与	两位全为1，结果为1，否则为0
|	按位或	两位存在1，结果为1，否则为0
^	按位异或	两位一个为0一个为1，结果为1，否则为0

符号	名称	解释
>>	算数右移	低位溢出，符号位不变，并用符号位补溢出的高位,本质连续除以n个2
<<	算数左移	符号位不变，低位补0，本质连续乘以n个2
>>>	逻辑右移（无符号右移）	低位溢出，高位补0

```
int a=1>>2; //00000001->00000000
int b=1<<2; //00000001->00000100
```

#四、流程控制

分支控制if ... else ...

单分支if语句
if (condition) { 代码块; }，条件为true则运行代码块，否则跳过
双分支语句if else语句
if (condition) { 代码块1; } else { 代码块2; }，条件为真则运行代码块1，否则运行代码块2
多分支语句if ... else if ... ...else语句
if (condition1) { 代码块1; } else if (condition2){ 代码块2; }...else{代码块n}，条件为真则运行代码块1，否则向后继续选择判断。
嵌套语句if(condition){if...else...}else{if...else...}语句

```
import java.util.Scanner;
public class run{
    public static void main(String[] args){
        Scanner myScanner=new Scanner(System.in);
        System.out.println("请输入年份：");
        int age=myScanner.nextInt();
        if ((age % 4 == 0 && age % 100 != 0) || age % 400 == 0){
            System.out.println(age+"是闰年");
        }else{
            System.out.println(age+"不是闰年");
        }

    }
}
```

分支控制switch

switch(表达式){case 常量1: 语句块1;break; case 常量2: 语句块2; break; ...default: 语句n; }
break表示跳出当前switch语句，如果都没有匹配上那就执行default

case子句中必须是常量不能是变量
switch的表达式返回值可以是byte，short，int，long，char,String，enum[枚举]不可以是float和double

```
import java.util.Scanner;
public class switch01{
    public static void main(String[] args){
    System.out.println("请输入字母a到g：");
    Scanner myScanner=new Scanner(System.in);
    char chr=myScanner.next().charAt(0);
    switch(chr){
    case 'a':System.out.println("星期一");break;
    case 'b':System.out.println("星期二");break;
    case 'c':System.out.println("星期三");break;
    case 'd':System.out.println("星期四");break;
    case 'e':System.out.println("星期五");break;
    case 'f':System.out.println("星期六");break;
    case 'g':System.out.println("星期日");break;
    default:System.out.println("输入错误");
    }
}
}
```

循环控制for

for(循环变量初始化; 循环条件; 循环变量迭代){
循环操作;
}
循环操作只有一个语句的时候可以省略大括号
循环变量只能在循环体中使用，如果想在外面使用那么就需要在外面声明

```
public class for01{
    public static void main(String[] args){
    for (int i = 0; i<10; i++){
    System.out.println("hello!");
    }
    }
}
```

增强for循环: for(数据类型 元素名 : 数组名)

```
int[] numbers={1,2,3,4,5,6,7,8,9,10};
for (int element : numbers){
System.out.println(element);
}
```

循环控制while

循环变量初始化;
while(condition){
循环操作;
循环变量迭代;
}
while循环先判断后执行
循环变量迭代使得程序有一个出口，当目标变量不满足条件的时候退出；或者在while循环中使用if加break退出循环

```
public class while02{
    public static void main(String[] args){
    int i = 1;
    while (i<=100){
    if (i%3==0)
    System.out.println(i);
    i++;
    }
    }
}
```

循环控制do ... while

do{
循环体;
循环变量迭代;
}while(循环条件0);
do...while循环先执行后判断，一定会执行一次。并且最后有一个分号;

```
public class dowhile01{
    public static void main(String[] args){
    int i=1;
    int sum=0;
    do{
    if(i%5==0&&i%3!=0){
    System.out.println(i);
```

```
    sum++;
    }
    i++;
    }while(i<=200);
    System.out.println("sum="+sum);
    }
}
```

多重循环

顾名思义就是多个循环语句进行嵌套
空心金字塔

```
import java.util.Scanner;
public class stars{
    public static void main(String[] args){
    System.out.println("请输入金字塔高度1-50");
    Scanner myScanner=new Scanner(System.in);
    int num=myScanner.nextInt();
    for(int i=1;i<=num;i++){
    for(int k=num-1-i;k>=0;k--){
    System.out.print(" ");//使用空格构建外轮廓所需要的缩进
    }
    for(int j=1;j<=2*i-1;j++){
    if(j==1||j==2*i-1||i==num)
    System.out.print("*");//使用*构造空心轮廓
    else
    System.out.print(" ");//使用空格填补空心
    }System.out.println();//回车转换下一层
    }
    }
}
```

跳转控制语句break

执行break会结束语句块的执行，一般配合条件判断语句使用在switch语句或者循环中

```
public class break01{
    public static void main(String[] args){
    int num;
    int i=0;
    while(true){
    num=(int)(Math.random()*100);
```

```
    i++;
    System.out.println(num);
    if(num==50)
    break;
    }
    System.out.println("i="+i);
    }
}
```

在多重嵌套循环中break语句可以使用标签指定需要终止的是那一层的代码块
label1:{ .....
label2:{
label3{
break label2;
}
}
}
没有指定则默认退出当前最近循环体
最好不用，因为容易混乱

```
import java.util.Scanner;
public class break02 {
    public static void main(String[] args) {
        Scanner myScanner = new Scanner(System.in);
        String user = "";
        String passcode = "";
        int i = 3;  // 初始设置3次机会

        // 进行3次密码和用户名的输入
        for (; i > 0; i--) {
            System.out.println("请输入用户名：");
            user = myScanner.next();  // 获取用户名输入
            System.out.println("请输入密码：");
            passcode = myScanner.next();  // 获取密码输入

            // 检查用户名和密码是否正确
            if (user.equals("丁真") && passcode.equals("666")) {
                System.out.println("登录成功！");
                break;  // 密码正确，退出循环
            }

            // 如果用户名密码不对，输出剩余机会
            if (i > 1) {
```

```
                System.out.println("你的账号或者密码有误\n你还有" + (i - 1) +
"次机会! \n");
            }
        }

        // 当机会用完后，输出失败信息
        if (i == 0) {
            System.out.println("登录失败!");
        }

        myScanner.close();  // 关闭Scanner
    }
}
```

跳转控制语句continue

continue结束本次循环，继续进行下一次循环
continue与break一样也可以使用标签，通过指定标签跳转指定循环体

跳转控制语句return

return使用在方法，表示跳出所在的方法。如果return
写在main方法中则会直接退出程序

#五、数组与排序查找

数组基本介绍

数组可以存放多个同一类型的数据，数组也是一种数据类型，是引用类型，数组就是一组数据，数组索引从0开始编号，
数组索引格式： 数组名[索引数]
数组长度： 数组名.length
数组定义格式：

1. 静态初始化: 数据类型[] 数组名={数组元素};
2. 动态初始化: 数据类型 数组名[]=new 数组类型[大小];
3. 动态初始化: 数据类型 数组名[]; 数组名=new 数组类型[大小]; //先声明再分配
   注意事项：
4. 数组是多个相同类型数据的组合
5. 数组创建后有默认值
6. 数组下标最大值是数组长度减一，因为数组是从0开始编号
7. 数组属于引用类型，数组型数据是对象

```
public class array01{
    public static void main(String[] args){
    char[] chars=new char[26];
    for(int i=0;i<chars.length;i++){
    chars[i]=(char)('A'+i);
    }
    for (int j=0;j<chars.length;j++){
    System.out.print(chars[j]+" ");
    }
    }
}
```

基础类型数据赋值这个值就是具体的值，并且相互不影响

```
int a1=10;int a2=a1; a1+=1; -->a1=11;a2=10;
```

数组在默认情况下是引用传递，赋的值是地址

```
int[] arr1={1,2,3};int[] arr2=arr1;arr2[0]=10;
arr1=arr2->{10,2,3};arr1和arr2均改变
```

数组的拷贝

```
int[] arr1={1,2,3};
int[] arr2=new int[arr1.length];
for(int i = 0;i < arr1.length; i++){
arr2[i]=arr1[i];
}
```

多维数组

静态初始化

```
int[][] arr={{0,0,0,0,0,0},{0,0,0,0,0,0}{0,0,0,0,0,0,0}};
```

动态初始化
二维数组其实是由多个一维数组组成的，这些一维数组可以长度相同也可以不同
创建规则数组

```
int a[][];a=new int[4][4];
```

创建不规则数组

```
int a[][]=new int[2][]; a[0]=new int[10];a[1]=new int[5];
```

arr.length在二维数组中表示每二维数组的行数，组成二维数组的一维数组数量，arr[i].length为二维数组第i行的元素数，组成二维数组中第i个一维数组的元素数

数组的复制

由于数组之间的赋值是引用赋值，所以不能实现数组数据的复制
要实现数组的复制可以使用java中的arraycopy的方法

```

```