**第一章 初识JAVA**

1.1java发展史

Java于1995年5月sun公司推出，由詹姆斯·高斯林（java之父）和其小组共同开发，最开始被称为Oak，后来改名为java。Java 在一定程度上继承了多数C语言的语法，但又有一定的改变，java既小巧又安全，而且可以移植，具有跨平台性，可以解决Internet语言问题。

1.2java环境搭建

下载JDK。JDK是用于开发java应用程序的开发包，它提供了编译、运行java程序所需的各种工具和资源，由jdk（快捷指令组）、jre、jvm组成。可在Oracle（甲骨文）网站下载。

配置环境变量。我的电脑—属性—高级系统设置—环境变量—（win10）找到path点击编辑，再新建一个通道，将安装的JDK下的bin的路径粘贴此处——新建java\_home，将JDK安装目录粘贴。

检查环境变量是否配置成功。Win+r键，打开程序搜索命令行，输入cmd——再输入java -version 查看是否提示JDK版本号。也可以再输入java命令验证。

1.3程序执行过程

java文件→cmd→javac 文件路径 文件名→生成.class字节码文件→java class文件路径 文件名

eclipse程序过程：

编译时自动转化为class文件，存在bin文件夹中，运行时在jvn虚拟机中运行，并输出结果。

要将eclipse中的文件拿出来直接运行是，需要删除pakege，才能从目录文件中删除。

导入外来java程序文件，将文件复制到储存空间，iport导入打开。

**第二章 数据类型和运算符**

2.1变量的命名

在java中，标识符用来为程序中的常量，变量，方法，类，接口和包命名。

标识符命名规则：

①：标识符由字母，数字，下划线(\_)，美元符($)组成

②：标识符的首字母有字母，下划线，美元符组成，不能以数字开头

③：标识符的命名不能与关键字，布尔值（false、true）和null相同

④：标识符区分大小写，没有长度限制，坚持见名知意原则

命名采用驼峰法（多单词组成时第一个单词首字母不大写第二个开始首字母大写）

备注：使用eclipse创建文件时文件名遵从该规则，但首字母也需大写。

2.2java中的注释

①：单行注释 // 只能书写在一行的注释，当只有一行内容需要注释时，一般使用单行注释

②：/\*......\*/ 多行注释 ， 当有多行内容需要注释时，一般使用多行注释

③：/\*\*.......\*/ 文档注释 类似多行注释，但会生成一份HTML格式的API帮助文档。

2.3数据类型

基本数据类型分为：

整数型：（byte,short,int,long)

小数型（float，double）

字符型：（char）

基本数据类型：布尔型（boolean）

以上为八大基本类型，数据类型分为八大基本类型和引用类型，引用类型如String，Scanner等。

2.4 java中的基本数据类型取值范围

|  |  |  |  |
| --- | --- | --- | --- |
| 基本类型 | 大小 | 实例 | 取值范围 |
| Boolean | 1字节8位 | true | true,false |
| byte | 1字节8位有符号整数 | -12 | -128~+127 |
| short | 2字节16位有符号整数 | 100 | -32768~+32767 |
| int | 4字节32位有符号整数 | 12 | -2^31~+2^31-1 |
| long | 8字节64位有符号整数 | 10000 | -2^63~+2^63-1 |
| char | 2字节16位Unicode字符 | ‘a’ | 0~65535 |
| float | 4字节32位浮点数 | 3.4F | -3.4E38~3.4E38 |
| double | 8字节64位浮点数 | -2.4e3D | -1.7E308~1.7E308 |

数据类型转换时，需要遵从由低到高的转换规则：

Byte short

Int long float double

Char

高的类型可以直接使用低的数据类型的数据： float a=2;

虚拟机中输入的整数类型默认为int类型，所以输入long类型数据时需后加L（大写防认错）： long a=24141214L；

虚拟机中输入的小数类型默认为doouble类型，所以输入float类型数据时需后加f： float a=15.42f

2.5 常量

常量指的是在程序运行中值不能改变的量。

比如布尔常量 只有 true和false

字符常量，输入法中所有的字符，但部分字符无法直接输出，需使用转义字符，比如：

\n 转义字符常量：表示换行

\b 转义字符常量：表示按backspace键

\t 转义字符常量：表示按TAB键

\\ 特殊字符常量：表示反斜杠

\’ 特殊字符常量：表示单引号

\” 特殊字符常量：表示双引号

2.6变量

与常量对应的就是变量，变量是在程序运行时值可以改变的量

“变量类型”可以从数据类型中选择

“变量名”是定义的名称变量，要遵循标识符命名原则

中括号 中的内容为初始值，是可选项

2.7常用运算符

（1）赋值运算符

= int a=5；把5赋值给a a=5

（2）算数运算符

+ - \* / %(取模) ++(自增1) --(自减1)

对于除法运算符"/",如果两个操作数为整数，结果会舍弃小数部分而保留整数部分。

如果操作数中有一个是浮点数，将会自动类型转换，结果会保留小数部分。

（3）：复合算术运算符

+= -= \*= /= %= 例如 a+=1 a-=1 //a=a+1 a=a-1

（4）逻辑运算符

逻辑运算的结果为boolean类型

\* &&：两个操作数都为true，结果为true。其他情况都为false

\* ||: 两个操作数都为false，结果false。其他情况都为true

\* ! :操作数位true，结果为false。 操作数为false，结果为true

（5）条件运算符

又称三目运算符/三元运算符

格式: 条件?表达式1:表达式2

首先判断条件，如果为true，则返回表达式1的值。如果为false，则返回表达式2的值。

例：int a;

a= 5<7?5:7

首先判断5<7是否成立（true或false）成立则取表达式1的值 此处是 5 不成立取表达式2的值 此处是7

2.8系统输出

打印：

\* System.out.println() 打印并换行。

\* System.out.print() 打印，但不换行

\* System.out.print() 使用\n转义符， 让其能换行打印

\* 使用\t，让光标移动到下一个水平制表位置。

**第三章 选择结构和 循环结构**

3.1 流程控制结构

3.1.1顺序结构

指的是程序从上到下依次执行每条语句的结构

3.1.2选择结构

根据条件的判断来选择执行不同的代码，选择结构可以分为单分支结构，双分支结构 和多分支结构

单分支：

\*if(判断条件){

\*代码  
\*}

双分支：

if的双分支结构：

\* if(条件){

\*代码1

\*}else{

\*代码2

\*}

当条件成立，执行代码块1；如果条件不成立，则执行代码块2

注意:else子句不能单独使用，它是if语句的一部分。

多分支：

\* if(条件1){

\* 代码块1

\* }else if(条件2){

\* 代码块2

\* }else if(条件3){

\* 代码块3;

\* }else if(....){

\* .....

\* }else{

\* 代码块n

\* }

首先对if语句的条件1判断，为true，则执行代码块1。之后分支结构结束。若为false，则判断条件2，为true则执行代码块2。之后分支结束。依次如此，最后，如果前面的条件都不成立，则执行else的代码块。

分支嵌套：

If(){

If（）{

}

Switch控制语句

Switch(表达式){

case 常量 1:

语句;

break;(注：非必输项)

case 常量 2:

语句;

break;

…

default:(注：非必选项)

语句;

}

注意事项

1. 如果每个case执行完后都需要跳出来，则不要忘记写break。
2. case后面的常量值不能重复。
3. default的位置顺序可以变动，但要符合逻辑顺序
4. Switch结构只能处理等值判断条件。针对于判断条件处在一个值得区间，则使用if结构或者三目运算符

3.1.3循环结构。 循环结构

3.2循环

While

while(循环条件，结果是boolean){

循环体

}

先判断循环条件，如果为ture，则执行循环体。

如果为false，结束循环

do\_while

while(循环条件，结果是boolean){

循环体

}

先判断循环条件，如果为ture，则执行循环体。

如果为false，结束循环

While与do\_while的区别在于，do\_while不管循环条件是否成立，至少有一次执行循环体。

For

for(表达式1;表达式2;表达式3){

循环体

}

首先执行表达1，一般进行变量的初始化。

然后执行表达式2，判断循环条件。

如果为true，则走循环体。

循环体执行完毕后，执行表达式3，改变循环变量的值。

次执行表达式2，判断循环条件.....后续依次如此。

三种循环结构的使用：

①：循环次数确定的情况下，通常使用for循环

②：循环次数不确定的请款古关系啊，通常使用while 和do while

break

break的使用

break语句只能出现在switch和循环语句中。 表示结束switch，或循环

continue

continue的作用：跳过本次循环，而进行下一次循环 。

continue只能出现在循环语句中。

return

结束当前的方法

JVM内存分区

Jvm内存分为：栈，堆，方法区

栈：存放文件名，

堆：存放方法名的内容，单个数为数据，多个数为方法区地址，

方法区：在数组堆中对应地址存放内容

**第四章 数组**

4.1．一维数组

4.1.1数组定义

数组是用来储存一组相同类型数据的数据结构。当数组初始化完毕后，java为数组在内存中，分配一段连续的空间，其在内存开辟的空间也将随之固定下来。此时数组长度就不在发生变化。即使数组没有保存任何数据，数组所占的空间依然存在(默认初始值)。

4.1.2jvm内存分区

Jvm内存分为：栈，堆，方法区

栈：存放文件名，

堆：存放方法名的内容，单个数为数据，多个数为方法区地址，

方法区：在数组堆中对应地址存放内容

4.1.3建立数组

方法1：

int[] a= {1.2.3};

数组类型 数组名 数组数值

方法2：

int[] a=new int[i];

具有i个元素的数组。

方法3：

int[] a=new int[]{1,2,3};

4.1.4获取数组元素

int[] a=new int[4];

0 0 0 0

下标0 下标1 下标2 下标3

赋值：

a[0]=1；

a[1]=2；

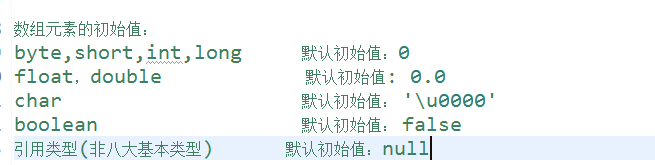
a[2]=3；

a[3]=4；

1 2 3 4

下标0 下标1 下标2 下标3

4.1.5数组默认初始值



4.1.6数组遍历

For遍历：

for(i=0;i<arr.length;i++){

System.out.println(arr[i])

}

增强型for：

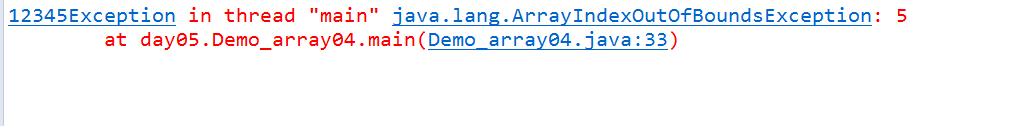
for(int num:arr){

System.out.println(num)

}

4.1.7

1、数组下标越界异常：java.lang.ArrayIndexOutOfBoundsException



2、声明数组并初始化的时候，没有固定数组长度。

|  |
| --- |
|  |

3、

|  |
| --- |
|  |

4.2二维数组

4.2.1二维数组定义

二维数组实际是一个一维数组，他的每一个元素又是一个一维数组

4.2.2二维数组与内存

|  |
| --- |
|  |
|  |

4.2.3建立二维数组

方法1：

int[][] arr=new int[3][5];

方法2：

int[][] arr;

arr=new int[3][5];

方法3：

int[][] arr=new int[3][];

arr[0]=new int[2];

arr[1]=new int[4];

arr[2]=new int[8];

方法4：

int[][] arr={{2,3},{2,41,312}{42，12，12，23}};

方法5：

int[][] arr=new int[][]{{2,3},{2,41,312}{42，12，12，23}};

4.2.4 Arrays类

JDK中专门用于操作数组的工具类，位于java.until包中。

Arrays.equals(array1,array2) 比较两个数组是否相等。只有当两个数组的长度相等， 对应位置的元素值也一一相等时，该方法返回true，否则返回false。

Arrays.sort(array) 对数组array进行升序排列

Arrays.toString(array) 将一个数组array转换为字符串

Arrays.fill(array,val) 把数值array的所有元素都赋值为val。

Arrays.copyOf(array,length) 把数组复制成一个长度为length的新数组，返回类型 与array数组类型一致

补充点：

System.arraycopy(源数组,源数组的起始位置,目标数组,目标数组的起始位置,要复制的数组元素个数);

**第五章 方法的运用**

1 方法

方法，函数，过程:

用于封装一段特定的逻辑功能。

方法可以在程序中反复调用

方法可以减少代码重复，便于程序维护

例如：

public static int sum (int num1,double num2){ 方法体 }

[修饰符] [返回类型] [方法名] [参数列表]

修饰符： 暂时不讲

返回类型: 如果方法有返回值，则必须匹配相应的类型。 如果没有返回值，则用void

方法名:给一段逻辑功能代码取个 名字。

参数列表:外界传入值到方法中，供使用(非必输)

方法的四种定义

1、无返回值，无参数列表

|  |
| --- |
|  |

2、无返回值，但有参数列表

|  |
| --- |
|  |

有返回值，但无参数列表

|  |
| --- |
|  |

有返回值，有参数列表

|  |
| --- |
|  |

**第六章 linux系统基础指令**

Java开发后，会部署在Linux上面。

所以对linux操作的一些命令要有些了解。

下面是我开发中经常用到的一些命令：

cd ：切换 cd huhua

cd .. :返回上级目录

pwd:查看路径

ls:查看当前目录下的所有文件

find:文件查找 find -name hello.txt

grep:内容查找 grep -lr 我们 \*

view:仅查看文件内容 view hello.txt

mkdir:创建文件夹 mkdir huhua

vi:创建查看并修改 创建文件 加后缀 vi study.xml

:wq :保存并退出

:q :不保存退出

tar :打包和压缩

tar -cvf 加包名 建包 tar -cfv huhua2018.tar

tar -rvf 添加文件 tar -rfv (你的文件路径+文件名)

tar -xvf 解压（注意目录） tar -cfv huhua2018.tar

ps -ef |grep java 查看含有"java"的活动进程

kill -9 pid 杀死进程

i : 插入

cp：复制文件

mv:移动文件

rm:删除文件

rm -r: 删除文件夹

上传

lcd 本地目录

cd 服务器目录

put 上传文件名

**第七章 Svn**

7.1SVN

7.1.1SVN定义

SVN是Subversion的简称，是一个开放源代码的版本控制系统，它采用了分支管理系统，它的设计目标就是取代CVS。SVN就是用于多个人共同开发同一个项目，共用资源的目的。

7.1.2运行方式

SVN服务器是由一个固定服务器充当云盘，用户可以自行传输及更改自己的文件。

7.1.3工作流程

（1）下载并安装svn，

（2）创建一个本地仓库，

（3）连接主机

（4）编号文件后放入自己想传送的文件夹，右键，tortoiseSVN，add自己的文件，commit并输入备注

7.1.4恢复版本

右键，showlog，选中之前的版本即可。

7.2 git

7.2.1 git

Git是一个开源的分布式版本控制系统，可以有效、高速地处理从很小到非常大的项目版本管理。

7.2.2 运行方式

Git是一个本地与网络服务器之间的信息传输功能软件。

7.2.3工作流程

（1）下载并安装svn，（首先建立用户：git config user.name “用户名” git config –global user.email “邮箱” ）

（2）创建一个本地仓库，并使用指令建立一个本地的 .git版本库，上传并备注自己的文件。（切换到仓库目录： cd 目标目录

建立本地仓库： git init ）

（3）连接github云仓库，（链接仓库：git remote add origin 仓库地址（网站） ）

（4）上传文件 （上传文件： git push -u origin master

）