第一天总结(Java 入门及开发环境搭建)：

1. 常用的 Dos 命令
   打开控制台命令
      win+r 回车
   d: 回车　转到 D 盘
   cd　改变当前目录
   cd.. 转到根目录
   cd\ 传到上一级目录
   dir 显示磁盘目录命令
   md 建立子目录
   rd 删除子目录命令
   tree 显示磁盘目录结构命令
   copy 文件复制命令
2. Java 语言的特点
   A: java 语言的源码是开放的。
   B: java 语言是跨平台，具有可移植性，与平台无关的编程语言。
   C: java 语言可以对内存垃圾自动收集。
   D: java 语言编写的程序虽然是”一次编译，到处运行”，但是必须要 java 运行环境。
   E: java 语言是跨平台的，但是 jvm 不是跨平台的，不同的操作系统需要不同的 jvm。
3.JDK 和 JRE　JVM
   JDK ： java 语言开发工具包　JRE:java 语言运行环境 JVM:java 语言虚拟机
    功能范围：JDK>JRE>JVM
4.java 语言的最基本的单位是类，用 class 表示
5.java 语言的执行过程
   A: 开发源程序(.java 文件)　Demo.java
   B: 通过 javac 命令编译(.class)　javac Demo.java
   C: 通过 java 命令执行　java Demo
6.环境变量
 (1)path 环境变量的作用
    让 javac 和 java 命令可以在任意的目录下使用。
 (2)path 环境变量的配置(掌握)
       A:只修改 path
          D:\develop\Java\jdk1.7.0_45\bin;以前的 path
       B:先建立一个 JAVA_HOME,后修改 path
          新建：JAVA_HOME 值是 D:\develop\Java\jdk1.7.0_45
          修改：%JAVA_HOME%\bin;以前的 path　(常用)
  (3)claspath 环境变量的作用
    让指定的 class 文件在任意目录下都可以被访问。
  在最左边配置一.;就可以了。

第二天总结(数据、数值类型和运算符)：

1.常量：在程序运行的过程中值不会改的量，而且可以被直接输出

分类：

A:字面值常量

1.整数常量 20 -89

2.实数常量 4.5 6.7 -4.9

3.字符串常量 “123” “changjiang”

4.布尔常量 true false

5.空常量

B：之定义常量

2.java 语言中的整数常量的表示

A：二进制 由 1 和 0 组成，以 0b 开头

B: 八进制 由 0-7 组成。以 0 开头。

C: 十进制 由 0-9 组成。默认就是十进制

D: 十六进制 由 0-9，A-F(不区分大小写)组成，以 0x 开头。

3.变量

在程序运行的过程中，其值在指定范围内发生改变的量。

变量定义的格式： 数据类型 变量名 = 初始化值；

或者： 数据类型 变量名；

变量名 = 初始化值；

4.数据类型

(1)基本数据类型(4 类 8 种)

A:整型

byte 1 字节 8 位

short 2 字节 16 位

int 4 字节 32 位

long 8 字节 64 位

B:浮点型

float 4 字节 32 位

double 8 字节 64 位

C:布尔型

true false

D：字符型

char 2 字符 16 位

注意：

整型数的默认类型是 int，浮点型数默认的类型是 double，

是 long 类型的话加后缀 l 或 L, float 类型需要加 F 或者 f 后缀。

类型转换

1.小到大(隐式转换)

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

2.大到小(显示转换)

格式：(转换后的数据类型)数据;

3. 面试题：

byte b1 = 3;

byte b2 = 4;

byte b3 = b1 + b2;//编译错误，byte　short　char 三种类型会默认转换为 int 类型。

byte b4 = 3 + 4;

第三天总结(运算符&程序流程控制语句)

5.运算符

(1)算术运算符

+,-,*,/,%,++,--

+：正号，加法，字符串连接符。

```
System.out.println("5+5="+5+5);//5+5=55
System.out.println(5+5+"=5+5");//10=5+5
```

%：取得余数

左边如果大于右边，结果是余数。

左边如果小于右边，结果是左边。

左边如果等于右边，结果是 0。

注意：在取余时正负号跟左边一致。

++/--：

++ 其实相当于把数据+1

单独使用：

在数据的前后，结果一致。

参与操作使用：

如果在数据的后边，数据先操作，在++/--

如果在数据的前边，数据先++/--，在操作。

(2)赋值运算符

=,+=,-=,*=,/=,%=

int a = 10;

把 10 赋值给 int 类型的变量 a。

a += 20;

把左边和右边的和赋值给左边。

注意事项：

a = a + 20;

a += 20;

结果是等价的，理解不是等价的。

因为+=这种运算符，内含了强制类型转换功能。面试题
比如：

short s = 2;
short ss = 2+s//编译出错，默认转换为 int 类型。
s+=3;//不报错，隐含了强制类型转换功能
等价于
s = (short)(s+3)

(3) 关系运算符
==,!=,>,>=,<,<=

特点：关系运算符的结果都是 boolean 类型。
请千万注意：== 不要写成 =

(4) 逻辑运算符
&,|,!,^,&&,||

&:有 false 则 false
|:有 true 则 true
!:true 变 false,false 变 true
^:相同 false,不同 true

&&:有 false 则 false
||:有 true 则 true
笔试中常见：
&&和&的区别是：如果左边有 false 了，右边将不再执行。
||和|的区别是：如果左边有 true 了，右边将不再执行。

开发中常用：
&&,||,!(s+3);

(5) 位运算符
符号表示：^
注意：一个数据对同一个数据^两次，结果还是数据本身。
举例：a ^ b ^ b = a
面试题：交换两个数据(不用第三方变量)
a=a^b;
b=a^b;
a=a^b;

(6) 三元运算符
格式：
条件表达式?表达式 1:表达式 2(三元运算符)

执行流程：
根据条件表达式返回的是 true 还是 false，决定结果是什么。
如果是 true,就把表达式 1 作为结果。

如果是 false,就把表达式 2 作为结果。

面试题：

用三元运算符比较出三个数中的最大值

int result = (a>b)?(a>c?a:c):(b>c?b:c);

6.if 语句(掌握)

(1)用于做判断使用的。

常见于对某个范围进行判断，或者几个变量进行判断，还有就是 boolean 表达式的判断。

(2)格式：

A:第一种格式

```
if(条件表达式)
{
    语句体;
}
```

执行流程：

如果条件表达式为 true，就执行语句体；

否则，什么都不执行。

B:第二种格式

```
if(条件表达式)
{
    语句体 1;
}
else
{
    语句体 2;
}
```

执行流程：

如果条件表达式为 true，就执行语句体 1；

否则，就执行语句体 2；

特殊：

可以和条件运算(三元运算)在某些情况下进行替换。

一般是在赋值的情况下可以。

举例：

获取两个数中的最大值。

```
if(a>b){
   return b;              <==========>      a>b?a:b;
}
else
return b;
```

C:第三种格式

```
if(条件表达式 1)
```

{
语句体 1;
}
else if(条件表达式 2)
{
语句体 2;
}
…
else
{
语句体 n;
}

执行流程：
如果条件表达式 1 为 true，就执行语句体 1；
如果条件表达式 2 为 true，就执行语句体 2；
…
否则，就执行语句体 n；

D:注意事项
a:什么时候时候哪一种 if 语句。
第一种格式在判断条件为一种情况下使用。
第二种格式在判断条件为两种情况下使用。
第三种格式在判断条件为多种情况下使用。
b:每一种 if 语句其实都是一个整体，如果有地方执行了，
其他的就不执行了。
c:如果 if 或者 else 里面控制的语句体是一条语句，是可以省略大括号的，
但是，如果是控制多条语句，就必须写上大括号。
建议：永远写上大括号。
d:大括号和分号一般不同时出现。举例说明：if (a>20);{}

7：switch 语句(掌握)
(1)用于做选择使用的。一般用于几个常量的判断。
switch 会把几个常量值直接加载到内存，在判断的时候，效率要比 if 高。
所以，针对几个常量的判断，一般选择 switch 语句。
(2)switch 语句的格式：
switch(表达式)
{
case 值 1:
语句体 1;
break;
case 值 2:
语句体 2;
break;
case 值 3:
语句体 3;

```
        break;
    ...
    default:
        语句体 n;
        break;
}
```

A:针对格式的解释

switch:表示这里使用的是 switch 语句，后面跟的是选项。

常量表达式支持的类型：byte,short,int,char

JDK5 以后可以是枚举(以后讲)

JDK7 以后可以是 String 字符串(后面讲)

case:表示这里就是选项的值，它后面的值将来和常量表达式的值进行匹配。

case 后面的值是不能够重复的。

break:

switch 语句执行到这里，就结束了。

default:

当所有的 case 和表达式都不匹配的时候，就执行 default 中的语句。

它相当于 if 语句的 else。一般不建议省略。

B:执行流程

进入 switch 语句后，就会根据表达式的值去找对应的 case 值。

如果最终没有找到，那么，就执行 default 的内容。

C:注意事项：

a:default 整体可以省略吗?

可以，但是不建议。

b:default 的位置可以放到前面吗?

可以，但是不建议。

c:break 可以省略吗?

可以，但是不建议。

default 在最后，break 是可以省略的。

case 后面的 break 可以省略，但是结果可能有问题。

d:switch 语句什么时候结束呢?

就是遇到 break 或者执行到程序的末尾

面试题：

```
int x=2,y=3;
switch(x)
{
default://3
    y++;
case 3://1  4
    y++;
case 4://2  5
```

switch 语句什么时候结束呢?

就是遇到 break 或者执行到程序的末尾

```
        y++;
    }
    System.out.println("y="+y);//6
```

8.Scanner 的使用

Scanner 的使用分为三步

(1)导包　import java.util.Scanner;

(2) 创建对象,封装键盘录入

```
Scanner sc = new Scanner(System.in);
```

(3) C:调用方法,获取数据

```
int number = sc.nextInt();
```

第四天总结(循环):

1.循环的组成

A:循环体,就是要做的事情。

B:初始化条件。一般定义的是一个初始变量

C:判断条件。用于控制循环的结束。

D:控制条件。用于控制变量的变化。一般都是一个++/--操作。

2.循环的分类:

A: for

```
for(初始化条件;判断条件;控制条件)
{
    循环体;
}
```

执行流程:

a:先执行初始化条件(只执行一次);

b:执行判断条件(只能是 boolean 类型的表达式)

c:根据判断条件的返回值:

true:执行循环体。

false:就结束循环。

d:最后执行控制条件。返回到 b 继续。

B: while

```
初始化条件;
while(判断条件)
{
    循环体;
    控制条件;
}
```

执行流程:

a:先执行初始化条件;

b:执行判断条件
c:根据判断条件的返回值：
true:执行循环体。
false:就结束循环。
d:最后执行控制条件。返回到 b 继续。

C: do...while(了解)

初始化条件;
do{
循环体;
控制条件;
}while(判断条件);

执行流程：
a:先执行初始化条件;
b:执行循环体和控制条件;
c:执行判断条件
d:根据返回值
true:返回 b。
false:就结束循环。

注意：
a:一般使用 for 循环或者 while 循环。而且这两种循环是可以等价转换的。
b:do...while 循环至少执行一次循环体。

3.案例：(理解)

A:正三角形理论

内循环的判断条件：y<=x

```
for(int x=0; x<5; x++)
{
    for(int y=0; y<=x; y++)
    {
        System.out.print("*");
    }
    System.out.println();
}
```

B:倒三角形理论

内循环的初始化条件：y=x

```
for(int x=0; x<5; x++)
{
    for(int y=x; y<5; y++)
    {
        System.out.print("*");
```

```
            }
            System.out.println();
        }
```

C:九九乘法表(扩展)

4.break 和 continue(掌握)

(1)有些时候，我们需要对循环进行一些控制终止,这个时候，就出现了两个关键字：

break 和 continue

(2)特点：

A:它们都必须在循环中(break 还可以在 switch 中。)。

一般在循环的判断中。

B:如果单独使用 break 和 continue，后面是不能有语句的，因为执行不到它后面的语句。

(3)区别：

A:break 结束当前循环。

B:continue 结束本次循环，进入下一次循环。

(4)如何退出嵌套循环：(了解)

用带标签的循环。

格式:

```
标签名：for(){
            for(){
                if()
                {
                    break 标签名;
                }
            }
        }
```

5：应用场景(理解)

(1)变量：发现有一个数据是变化的时候，就要用变量。

(2)if 语句：如果是一个范围的判断，boolean 类型的表达式的判断，几个数据的判断。

(3)switch 语句：几个数据的判断。一般这种情况，优先选择 switch。

(4)for 语句：如果次数或者范围特别明确。(水仙花)

(5)while 语句：如果次数或者范围不明确。(珠穆朗玛峰)

第五天总结(函数&&数组)

1：函数(掌握)

(1)定义在类中，有特定功能的一段小程序。

(2)函数的格式：

```
修饰符 返回值类型 函数名(形参类型 形式参数 1,形参类型 形式参数 2...)
{
    函数体;
    reutrn 返回值;
```

}

A:修饰符 public static
B:返回值类型 程序最终结果的数据类型
C:函数名 其实就是函数的名称,方便我们调用。
D:参数
形参类型　　数据类型
形式参数　　就是接收实际参数的变量
实际参数　　就是实际参与操作的变量(常量)
E:函数体 就是按照正常的逻辑完成功能的代码。
F:返回值 就是程序的最终结果
G:reutrn 返回值　哪里调用程序，return 就把结果返回到哪里。

(3)函数的特点：
A:函数与函数之间是平级关系。不能在函数中定义函数。
B:运行特点　函数只有被调用才执行。

(4)案例：
有明确返回值的例子：

A:求两个数据的和
B:求两个数据的最大值
C:比较两个数是否相等

void 类型例子：

A:nn 乘法表
B:根据给定的行和列输出一个*组成的长方形

(5)函数的调用
A:有明确返回值
a:单独调用　一般没有意义。
b:输出调用　但是如果想拿结果继续操作，就有问题了。所以，不好。
c:赋值调用　推荐方式。

B:void 类型
单独调用

(6)函数重载
A:函数名相同,参数列表不同(个数不同,对应的类型不同)。
与返回值类型无关。

B:举例：
```
public static int sum(int a,int b){...}
public static int sum(int a,int b,int c){...}
public static int sum(float a,float b){...}
```

2：数组(掌握)

(1)数组是存储同一种类型的多个元素的容器。
(2)好处：数组中的元素会被自动从 0 开始编号，方便我们获取。
(3)格式：

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

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

C:int[] arr = new int[]{1,2,3};

D:int[] arr = {1,2,3};

推荐 A 和 D。

(4)Java 内存图：

A:栈 存储局部变量使用。

使用完毕，立马消失。

B:堆 所有 new 出来的都在堆里面。

a:每一个实体都有地址值

b:每一个实体内的内容都有默认值

整数：0

浮点数：0.0

字符：'\u0000'

布尔：false

c:在垃圾回收器空闲的时候被回收。

C:方法区

D:本地方法区

E:寄存器

(5)操作：

数组的索引。

举例：System.out.println(arr[2]);

数组的长度。

数组名.length

A:数组的遍历

```
class Test
 {
public static void main(String[] args)
 {
int [] arr ={1,2,3,4,5,6,7,8};
for (int x=0;x<arr.length ;x++ )
    {
    System.out.print(arr[x]+" ");
  }
System.out.println();
}
```

```
    }
```

B:数组获取最值

```
class Test
 {
    public static void main(String[] args)
 {
int [] arr ={13,42,33,41,56,46,77,48};
 int max=arr[0];
   for (int x=1;x<arr.length ;x++ )
                {
   if (arr[x]>max)
      {
   max = arr[x];
  }
                }
System.out.println(max);
  }
}
```

C:数组的查找

```
package it.cast;

/*
                数组的查找:
 */

import java.util.Scanner;

public class Test {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("请输入你要查找的数据：");
        int num = sc.nextInt();
        int[] arr = { 13, 42, 33, 41, 56, 46, 77, 48 };
        boolean b = false;
        int y = 0;
        for (int x = 0; x < arr.length; x++) {
            if (arr[x] == num) {
                y = x + 1;
                b = true;
            }

        }
```

```
            if (b == true) {
                System.out.println("你要查找的数据" + num + "在数组中第" + y + "个位置");
            } else if (b == false) {
                System.out.println("你要查找的数据" + num + "不在数组中");
            }
        }
}
```

(6)二维数组：

格式：

A:int[][] arr = new int[3][2];//定义一个长度为 3 的二位数组，每个一维数组的长度为 2

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

C:int[][] arr = {{1,2,3},{4,5},{6,7,8,9}};

遍历：

```
/*
    二维数组遍历操作
*/

class ArrayTestDemo1
{
    public static void main(String[] args)
    {
        int [][] arr ={{11,12},{21,22,23},{31,32,33,34}};
        for (int x=0;x<arr.length ;x++ )
        {
            for (int y=0;y<arr[x].length;y++)
            {
                System.out.println(arr[x][y]+" ");
            }
        }
    }
}
```

应用：遍历求和。

```
/*
    二维数组的遍历求和。
*/

class ArrayTestDemo
{
    public static void main(String[] args)
    {
```

```
        int [][] arr = new int[3][];
        arr[0] = new int[]{11,12};
        arr[1] = new int[]{21,22,23};
        arr[2] = new int[]{31,32,33,34};
        int sum = 0;
        for (int x=0;x<arr.length ;x++ )
        {
            int groupSum = 0;
            for (int y=0;y<arr[x].length;y++)
            {
                groupSum = groupSum + arr[x][y];
            }
            sum = sum +    groupSum;
        }
        System.out.println("二维数组元素和为 sum="+sum);
    }
}
```