**第二章笔记**

**关键字**

1. JAVA中的关键字有：public static void int class，关键字必须是小写，没必要去记。
2. goto 和 const是保留字，在java中无意义，变量和类不能以此命名。
3. main System Static 这三个都不是关键字。

**标识符**

1. 标识符用于给变量、类和方法命名。
2. 标识符的命名规则：①由字母、数字、下划线、$组成②不能以数字开头③不能是关键字。
3. JAVA中区分大小写。
4. 标识符要有意义。
5. 标识符的命名规范：类名大写开头，如 class Hello;变量和方法小写开头，如 int ageOfPerson，但是命名规划不是强制的，只是为了让人能够一眼看懂，便于理解程序。

**变量**

1. 变量就是内存的名字。
2. 变量是有类型的，int型变量只能存储整数数据，double可以存储双精度浮点型数据，string只能存储字符串
3. 变量的定义：变量类型 变量名 = 变量的值，如int price=9。
4. 在一个{}内一个变量只能定义一次，即使类型不同。
5. 局部变量在使用前必须初始化，否则会报错。

**数据类型**

1. 在JAVA中数据类型分为整数（byte short int long）和小数（float double）。
2. 整型默认为int，小数默认为double。
3. 变量尽量选择小范围的数据类型，以节省空间。
4. 声明时值不能超出该类型的范围。
5. 声明long型变量时要在末尾加大写的“L”，如long l2=3333333333333333L。

**整数类型转换**

1. 隐式类型转换： byte b=3；

int i=b；

1. 错误的： int i=3；

byte b=i；

1. 显式类型转换：int i=3；

byte b=（byte）i；

1. 小给大，没问题；大给小，需转换。

**小数类型**

1. double d=3.14d; double d=3.14D; double d=3.14; double d=3;均可。

2. 但是float f=3.14; 不可，因为JAVA中小数默认为double类型。

除非float f=3.14f（F）；加个f或者F。

3. float to double没问题；double to float需转换。

4. int to double 没问题；double to int 需转换。

5. 小给大，没问题；大给小，需转换。

**小数类型计算问题**

1. (int)5F/8F和(int)(5F/8F)不同。

2. 5/8 5.0/8 5f/8f 是不同的。

**常用数据类型**

1. 布尔类型：boolean: true/false。
2. 字符串：String
3. 转义字符：回车：\n 斜杠\ : \\ 引号：\’ 或\”

**字符串与整数、小数的转换**

1. int to string : Integer.toString();
2. String to int : Integer.parseInt();
3. double to string : Double.toString();
4. String to double : Double.parseDouble ();

**基本运算符**

1. + - \* / %
2. 自增 ++
3. 自减 –
4. + 可用于字符串的拼接:”abc”+”def” 或者 “abc”+5得到的也是字符串“abc5”。
5. “abc”+5+6 得到”abc56”

5+6+“abc” 得到”11abc”

“abc”+(5+6) 得到”abc11”

**赋值运算符**

1. x++等价于x=x+1等价于x+=1；
2. \*=,/=,+=,-=
3. 赋值运算符也是一个值，等于赋值之后的变量的值。

**Scanner**

1. Scanner sc= new Scanner(System.in); //使用Scanner前必须写的

System.out.println(“请输入年龄”);

int age=sc.nextInt();//等待输入整型 //这是用于输入一个整数的语句

System.out.println(“请输入名字”);

String s=sc.nextLine();等待输入字符串 //这是一个用于输入字符串的语句

1. 当遇到Scanner无法解析的情况时，一是由于编译器版本太旧，至少得是1.7版；二是由于未导入Java.until.Scanner这个包，将鼠标放在Scanner上面即可选择导入

**基本类型的赋值**

public class Test2

{

public static void main(String[] args)

{

int i=10;

int j=i;

i=20;

System.out.println(j); //输出结果为10而不是20，因为在赋值的时候就把值//拷贝过去了

}

}

**交换两个变量**

int i=10,j=20,temp; //交换两个变量的时候需要创造一个第三方变量用于暂时存储

temp=i; //i=10,j=20,temp=10

i=j; //i=20,j=20,temp=10

j=temp; //j=10,i=20,temp=10

**比较运算符运算优先级**

1. 比较运算符用于进行值的真假性判断，结果是Boolean(不同于C语言，JAVA中的真假用true和false二不是1和0)，有==,>=,<=,!=,>,<

例子：

public class Test3

{

public static void main(String[] args)

{

System.out.println(3==5); //输出false

int i=9;

System.out.println(i>=3); //输出true

System.out.println(i==3); //输出false

}

}

1. 也可以把比较的结果赋值给Boolean类型的变量，比如

public static void main(String[] args)

{

boolean flag;

int a=3,b=4;

flag =a==b;

System.out.println(flag); //输出false

}

1. 判断字符串是否相等不要用==而用equals

public static void main(String[] args)

{

String s1="abc",s2="def";

System.out.println(s1==s2); //输出结果为false

System.out.println(s1.equals(s2)); //输出结果为false

}

1. 不必要记住优先级，只需要记住()优先级最高，用它来写出可读性高的程序，而不是来考倒别人。

**逻辑运算符与或非**

1. ==与=是不同的，比如

**public** **static** **void** main(String[] args)

{

**int** a=3,b=4;

System.*out*.println(a==b); //输出为Boolean值false

System.*out*.println(a=b); //输出为数值4

}

结果不同的原因在于“==”用于判断真假，它的值为Boolean类型，而“=”赋值符也是一个值，等价于它右边的值

1. int a=3,b=4;

boolean b1=(a=b); //有语法错误，因为不能将整型赋值给Boolean型变量

1. boolean b1=false;

boolean b2=true;

boolean b3=(b1=b2);

System.out.println(b3); //输出true

1. 逻辑运算符对布尔类型进行运算，有&&，||，！ 。 其中&&是二元运算符，只有当左右都为真时才为真；||也为二元运算符，当其中一个为真时就为真；！为一元运算符，取反，真变假，假变真

**短路运算**

1. 使用 | | 时，如果左边为true，那么不执行右边的语句。

2. 使用 &&时，如果左边为false，那么不执行右边的语句。

3. 使用！时无短路。

**三元运算符**

1. 条件表达式？表达式1：表达式2。如果条件表达式为true，则执行表达式1，否则执行表达式2。

**IF**

1. if(条件表达式){表达式}，如果条件表达式为真，则执行大括号中的语句，否则不执行。
2. 字符串长度 变量名.length()
3. if的语句后面加上return可以在执行完后跳过后面的语句
4. 常见问题：

* if的括号中不能使用“=”，而应使用“==”
* JAVA中if的条件表达式只接受布尔值，不像C语言只接受数值。
* 即使只有一条执行语句，也要使用大括号,否则容易出问题。
* if的括号后面加个;表示执行满足条件则执行空的语句

**ＩＦ　ＥＬＳＥ**

1. ｉｆ（比较表达式）

｛比较表达式为true时执行这里｝

　ｅｌｓｅ

｛比较表达式为false时执行这里｝

２．ｉｆ（条件表达式1）｛｝

ｅｌｓｅ　ｉｆ（条件表达式2）｛｝

ｅｌｓｅ　ｉｆ（条件表达式3｛｝

．．．．．．．

ｅｌｓｅ｛｝//最后最好加上else

当满足其中任何一个条件后就结束，不再判断接下来的条件。

**if的嵌套**

* 1. 在“if” “if else” “if elseif else”三个语句中都可以无限嵌套新的任意的这三个语句
  2. 尽量避免过多的

**switch case**

1. switch（表达式）

{

* 1. case 值1:语句1
  2. break;
  3. case值2: 语句2
  4. break;
  5. ···········
  6. default:
  7. break;

}

1. case后面只能是确定的值
2. switch括号中的表达式只能是byte，short，int，char类型,JDK1.5后可以是枚举类型，JDK1.7后还可以是String类型
3. break意味着switch的结束，default相当于if else if中的else
4. 一般每个case后面都要加个break，否则不会结束，而是继续执行下面case的语句
5. fall through：当多个case的代码一样时，可以合并写，在他们的最后加上break
6. 当进行当个离散值判断的时候，switch可以代替if，switch能做的if都能做，反正不一定