Java 基础学习笔记

一、java 与 C 语言的区别

由于在大学时只学过 C 语言，现在开始学 java 语言，总感觉 C 语言在脑海中“先入为主”的观念对 java 语言的学习有一定的干扰（虽然这两个语言在一些方面有很高的相似性和相同性）。下面对 java 基础阶段遇到的问题做一些初步的总结。

1. main 函数的区别

C 语言中 main 函数可以有返回值(一般为 int 类型)，也可以没有返回值(一般为 void 关键字)，例如：

```
int   main(…)      //有返回值
void   main(…);   //无返回值
```

C 语言中的 main 函数可以有参数(第一个参数是：int argc，表示命令行中参数的个数，第二个参数是 char * argv[]指向字符串的指针数组)，也可以没有参数。

例如：

```
int   main(int argc   char * argv[])          //有返回值，有参数
void   main( void )                           //无返回值，无参数
```

java 中 main 函数没有返回值（为 void 关键字），有一个参数（String[] args），java 的 main 函数（方法）必须是如下定义，这是 java 的规范。

public static void main (String[ ] args)

初学时，经常把 C 语言的习惯带到 java 中，

例如 public static int main(String[] args)…编译时就会报错。

2. 条件表达式的区别

在 C 语言中表示真/假的变量一直是由 int 类型来表示，C99 新添加了_Bool 类型。一个布尔变量只有 1(真)或 0(假)两种情况。并且 C 语言把任何非零的值都认为是真，零就是假。如果编译器不支持 C99 标准可以使用 int 来代替_Bool，所以 C 语言中 if 和 while 的条件判断语句的值可以是_Bool 型的真(1)和假(0),也可以是 int 类型的 0 和非零。（本段参照《C Primer Plus 第五版中文版》的 124 页）。

例如：

```
int a=1, b=2;
if(a<b){          //条件表达式的值为布尔类型的：true
    printf(“a 比 b 小”);
}

int i = 3;
while(i){         //条件表达式的值为 int 类型的：3
    printf(“a 比 b 小”);
}
```

在 java 中表示真假的变量只能是 boolean 类型，且值只能是 true 或 false，不能是 int 或其他类型。

例如：

```
boolean a = true;
while(a){
    System.out.println(“这是死循环语句”);
}
```

注意：在 java 中真/假不能用 int 类型的非零和零表示。

3. 输出语句 print 的区别

在 C 语言中，打印语句 printf（）函数的每个参数之间是用逗号“，”隔离，而在 java 中 System.out.println()函数只能用连接符“+”号来隔离不同类型的参数。

例如：printf("%d, %o", a, a);　　//C 语言输出语句

System.out.println(12+'a'+"hello");　//java 输出语句

二、java 中数据类型和数据类型转换

1、java 中有基本数据类型和引用数据类型

基本数据类型：4 类 8 种

整数类型：

byte	1
short	2
int	4
long	8

浮点型：

float	4
double	8

字符型：（无符号类型，范围：0---65535）

char	2

布尔型：

boolean 1 或 0

注意：整数默认使用 int 类型，浮点数默认使用的 double 类型。

例如：

byte a=0,b=3; byte c =a+b;　//错误，因为 byte 类型参与计算时默认是 int 类型

改为：byte c = (byte)(a+b);　//需加强制类型转换

float f = 1.1　//错误，因为浮点数默认使用的 double 类型

改为：float　f = 1.1f 或 float　f = 1.1F　//需加上显式类型声明

引用数据类型：

类 class

接口 interface

数组 [ ]

2、强制类型转换：

大的数据类型向小的数据类型转换。需要程序猿手动完成。

格式：目标类型 变量名=(目标类型)(被转换的数据);

例如：　byte　b;

byte　b1 = 3;

byte　b2 = 4;

b = (byte)(b1+b2);//byte 类型参与运算时自动转换为 int 类型。

3、隐式类型转换：

小的数据类型向大的数据类型转换时，编译器自动完成。

例如：　int i = 4;

short a = 3;

i+a 的结果是 int 型的 7，因为 short 型的 a 和 int 型的 i 参与计算时，a 自

动先被转换成 int 类型，然后在参与运算。

数据类型的转换顺序：

byte、short、char → int → long → float → double

注意：byte,short,char 相互之间不转换，他们参与运算时首先转换为 int 类型，然后在参与运算。

三、运算符的区别

1、 逻辑运算符 && 和 & 的区别?

1.1、双与“&&”也叫“短路与”，当运算符的左边表达式为假时右边不执行，整个表达式为假，该运算符执行速度快，效率高，例如：

```
int x = 1,y = 1;
if(x++==2 && ++y==2) {
    x =7;
}
System.out.println("x="+x+",y="+y);
```

结果：x = 2, y = 1;

1.2、单与“&”不管运算符的左边表达式是否为假，都执行右边表达式，该运算符执行效率低。例如：

```
int x = 1,y = 1;
if(x++==2 & ++y==2) {
    x =7;
}
System.out.println("x="+x+",y="+y);
```

结果：x=2, y=2

单与“&”和双与“&&”的逻辑运算的执行结果一直，但实际项目中推荐使用双与“&&”逻辑运算。

2、 == 与 = 的区别

==是判断运算符，=是赋值运算符

四、控制跳转语句 break、continue 和 return 的区别

break: 结束当前循环，程序执行循环后面的代码

continue:结束本次循环，继续下一次循环

return: 结束当前方法，return 后面的语句不执行

五、方法重载

在同一个类中，方法名相同、方法的参数个数或者参数类型不同，叫做方法重载

注意：方法重载是在同一个类中，与方法的返回值类型无关，只看方法名和参数列表。在调用时，JVM 虚拟机通过参数列表的不同来区分同名方法。

例如：下列哪个选项与 show 不是函数重载？

```
class Demo {
    void show(int a,int b,float c){}
}
```

A.void show(int a,float c,int b){}

B,void show(int x,int y,float z){}

C.int show(int a,float c,int b){return a;}
D.int show(int a,float c){return a;}
答案 B，因为 B 选项的参数个数和参数类型和示例中的一致，不符合方法重载的定义。

六、System.out.println()函数的输出组合方式

下列程序执行的结果是：

```
public class Test()
{
    public static void main(String[] args)
    {
        System.out.println(“”+’a’+1);
    }
}
```

结果是：a1

System.out.println()函数的输出组合方式有：

数字加字符：结果输出是数字：1+'a'=98
字符加数字：结果输出是数字：‘a’+1=98
字符加字符串：结果输出是字符串：‘a’+“bcd”=abcd
字符串加字符：结果输出是字符串：“bcd”+‘a’=bcda
字符串加数字：结果输出是字符串：“bcd”+1=bcd1
数字加字符串：结果输出“是”字符串：1+ “bcd” =1bcd

七、s += 1 和 s=s+1 的区别

1、java 中“+=”运算符隐藏了强制类型转换，例如：

```
short   s = 23;
s += 23;          //本句相当于 s = (short)( s+23 );
```

2、s=s+1 需要加强制类型转换，例如

```
short   s = 23;
s = s + 23 ;      //错误，因为在 java 中 short 类型的变量 S 在参与运算时先被转换成 int 类型，然后在参与运算，此处的结果是 int 类型，需要加上强制类型转换，即 s = (short)(s + 23).
```

本文是我在学习 java 基础时，对 java 基础部分容易混淆的知识点，做一些简要的总结，如有遗漏或错误，欢迎大家批评指正，本人愿和大家一起努力学习，共同进步。后续将会对面向对象作出一些总结。