**数据类型和运算符作业**

1. **填空题**
2. Java语言规定标识符由（ 字母、数字、下划线、美元符、中文(jdk1.8之后) ）组成，并且第一个字符不能是（ 数字 ）。
3. Java中整型变量有（byte、short、int、long ）四种，不同类型的整数变量在内存中分配的字节数不同，数值范围也不同。对于int型变量，内存分配（ 4 ）个字节。
4. 在Java中浮点型变量有（ double ）和（ float ）两种，对于float型变量，内存分配（ 4 ）个字节，尾数可精确到7位有效数字，对于double型变量，内存分配（ 8 ）个字节。
5. char c=’a’;System.out.println(c+1);运行结果为:（ 98 ）。
6. （ && ）是短路与运算符，如果左侧表达式的计算结果是false，右侧表达式将不再进行计算。
7. 下面的语句是声明一个变量并赋值：boolean b1=5!=8； b1的值是（ true ）。
8. 哪一个运算符运算效率最高：（ 运算符 ），所以最有效率的方法算出2乘以8等于多少的语句是（ 2 << 3 ）。
9. 基本数据类型的类型转换中，要将double类型的变量3.14159赋给为整数类型变量n的语句是（ int n=(int)3.14159; ）。
10. 八进制整数012表示十进制的（ 10 ），十六进制数0x3C表示十进制的（ 60 ）。
11. 一个十进制整数转换成八进制数后是1234，则它转为十六进制数后是（ 29c ）。
12. **选择题**

|  |  |  |
| --- | --- | --- |
| **1.** | **在Java中，以下错误的变量名是（ D ）。（选择一项）** | |
|  |  |  |
|  | **A** | constant |
|  | **B.** | flag |
|  | **C.** | a\_b |
|  | **D.** | final 这个是关键字 |

|  |  |  |
| --- | --- | --- |
| **2.** | **以下选项中属于合法的Java标识符的是（ CD ）。（选择两项）** | |
|  |  |  |
|  | **A.** | public 这个是关键字 |
|  | **B.** | 3num 不能数字打头 |
|  | **C.** | name |
|  | **D.** | \_age |

|  |  |  |
| --- | --- | --- |
| **3.** | **在Java中，byte数据类型的取值范围是（ A ）。（选择一项）** | |
|  |  |  |
|  | **A** | -128 ~ 127 |
|  | **B.** | -228 ~128 |
|  | **C.** | -255 ~ 256 |
|  | **D.** | -255 ~ 255 |

|  |  |  |
| --- | --- | --- |
| **4.** | **下面的代码段中，执行之后i和j的值是（ C ）。（选择一项）** | |
|  | **int** i=1;**int** j;  j=i++; //先人后己， 先赋值再++ | |
|  |  |  |
|  | **A** | 1，1 |
|  | **B.** | 1，2 |
|  | **C.** | 2，1 |
|  | **D.** | 2，2 |

|  |  |  |
| --- | --- | --- |
| **5.** | **下面Java代码的执行结果是（ A ）。（选择一项）** | |
|  | **public class** Test {  **public static void** main(String args[]) {  System.***out***.println(100 % 3); //余数1  System.***out***.println(100%3.0); //余数1.0  }  } | |
|  |  |  |
|  | **A** | 1 1.0 |
|  | **B.** | 1 1 |
|  | **C.** | 1.0 1.0 |
|  | **D.** | 1. 33.3 |

|  |  |  |
| --- | --- | --- |
| **6.** | 下面的赋值语句中错误的是**（ A ）。（选择一项）** | |
|  |  |  |
|  | **A** | **float** f = 11.1; //11.1是double类型，想要赋值给float应该要在尾部加f |
|  | **B.** | **double** d = 5.3E12; //16进制的数 |
|  | **C.** | **double** d = 3.14159; |
|  | **D.** | **double** d = 3.14D; //16进制的数 |

|  |  |  |
| --- | --- | --- |
| **7.** | **在Java中，下面（ AB ）语句能正确通过编译。（选择两项）** | |
|  |  |  |
|  | **A** | System.*out*.println(1+1); |
|  | **B.** | **char** i =2+'2';  System.*out*.println(i); |
|  | **C.** | String s="on"+'one'; //’one’能用单引号吗？ |
|  | **D.** | int b=255.0; //一个浮点型数能赋值给int吗？ |

|  |  |  |
| --- | --- | --- |
| **8.** | **以下Java运算符中优先级别最低的两个选项是（ AB ）。（选择二项）** | |
|  |  |  |
|  | **A** | 赋值运算符= |
|  | **B.** | 条件运算符 ?= //这里是指三目运算符 |
|  | **C.** | 逻辑运算符| |
|  | **D.** | 算术运算符+ |

|  |  |  |
| --- | --- | --- |
| **9.** | **有以下方法的定义，请选择该方法的返回类型（ D ）。（选择一项）** | |
|  | method(**byte** x, **double** y) {  **return** (**short**)x/y\*2; //运算过程中自动往大的类型转  } | |
|  |  |  |
|  | **A** | byte |
|  | **B.** | short |
|  | **C.** | int |
|  | **D.** | double |

|  |  |  |
| --- | --- | --- |
| **10.** | **关于以下Java程序中错误行的说明正确的是（ B ）。（选择一项）** | |
|  | **public class** Test2 {  p**ublic static void** main(String[] args) {  **short** s1=1;//1  s1=s1+1;//2 这里的1是int类型，int和short运算结果会变为int类型  s1+=1;//3 这里用的+=运算符，这种运算符会自动处理数据类型  System.*out*.println(s1);  }  } | |
|  |  |  |
|  | **A** | 1行错误 |
|  | **B.** | 2行错误 |
|  | **C.** | 3行错误 |
|  | **D.** | 1行，2行，3行都错误 |

|  |  |  |
| --- | --- | --- |
| **11.** | **如下写法哪些是不对的（ BD ）（选择二项）** | |
|  |  |  |
|  | **A** | byte b = 30; |
|  | **B.** | byte c = 500; //超出了byte的取值范围-128~127 |
|  | **C.** | long d = 2343223; |
|  | **D.** | float f = 3.14; //要在3.14尾部加f |

1. **判断题**
2. Java中变量名不区分大小写，number和NuMbEr代表同一个变量。**（ × ）**
3. 在编译Java源程序时，计算机根本不会去识别各个变量名的具体含义，因此命名规范对编写Java程序而言不是必要的，而且有可能会降低编写Java源程序的效率。**（ × ）**
4. Java语言中不同数据类型的长度是固定的，不随机器硬件不同而改变。**（ × ）**
5. char类型变量可以存储一个Unicode字符，在内存中占2个字节，该字符可以是一个汉字。**（ × ） //如果utf8编码就占3个字节，如果是gbk编码就占2个字节**
6. 运算符||和&&既是逻辑运算符，也是位运算符。根据两侧操作数的类型判断是逻辑运算符还是位运算符。**（ × ） 位运算的 或| 与&**
7. ^ 是异或位运算符，运算规则是如果两个操作数相同，结果是0，否则结果是1。**（ √ ）**
8. 赋值运算符优先级别低于条件运算符，条件运算符优先级别低于算术运算符。**（ √ ）**
9. 赋值和条件运算符是运算级别最低的两种运算符，都具有自右向左的结合性。**（ √ ）**
10. 整型常量的默认类型是int类型，浮点常量的默认类型是float类型。  
    **（ × ）**
11. 00101010 & 00010111语句的执行结果为00111111**（ × ）**
12. **简答题**
    1. Java是一种强类型语言，说明Java的数据类型分类。

引用类型  
 基本类型（8个）

* 1. i++和++i的异同之处  
     i++ 先人后己  
     ++i 先己后人
  2. 运算符||和|的异同之处  
     || 和 | 都代表“或”

||是逻辑运算符，具有短路模式   
|是位运算符，不具备短路模式

* 1. Java中基本数据类型转换的规则  
     小转大（自动转换），运算过程中，自动往大类型转！！  
     大转小（强制转换，需要我们编写代码手动转换）
  2. 编程时，为什么需要注释? 多行注释能不能嵌套使用?  
     注释就是对代码的解释，对代码的说明，提高代码的可读性。  
     多行注释不能嵌套使用
  3. 标识符的作用是什么?这个标识符合法吗? int aaa@bbb = 33;标识符能不能使用汉字开头?为什么?  
     标识符的作用是起一个名字？方便后面使用？
  4. 不合法，特殊字符只可以使用$和\_
  5. 能用汉字开头，因为Java支持汉字的字符集，但不推荐这么做
  6. java中有没有goto语句?有没有goto关键字?  
     java中有goto关键字，但是已经过时了（已经不允许使用了）！为什么要放弃？因为这个goto太随性了，代码的可读性及差

goto也是流程控制的一种（想去哪就去哪，随意定制）

* 1. byte,short,int,long类型，分别占用几个字节?表示范围多大?如果想表示我国的GDP总额，使用哪个类型较好?  
     GDP总额这个数很大，使用long类型较好
  2. float,double分别占用几个字节?如何用科学计数法表示3.14?  
     float占4个字节，double占8个字节

3.14E0

* 1. 整型常量默认是什么类型?浮点常量默认是什么类型?  
     整型数据默认是int

浮点型数据默认是double

* 1. 浮点数能用于比较吗?下面可能打印什么结果:

|  |
| --- |
| float d1 = 423432423f;  float d2 = d1+1;  if(d1==d2 ){  System.out.println("d1==d2");  }else{  System.out.println("d1!=d2");  } |

可以比较，会打印d1！=d2

* 1. 字符型变量几个字节? 这种写法对吗? char c = "d";  
     不一定吧，取决于编码了

不对，java中双引号代表String类型，char要用单引号

* 1. 布尔型变量占用空间是一位还是一个字节?  
     八分之一，因为只有1 or 0 一个字节是8位，所以是1。虽然实际使用是1个bit，但也不一定。直接占用最小单位1个字节也有可能，或者干脆用int来存，4个字节？
  2. 这种写法好不好? if(b==true)  
     不好，因为直接写if（b）就可以了，也都能看明白
  3. 常量的声明使用哪个关键字?

final

* 1. 常量的命名规范是?解释一下，驼峰原则?  
     常量命名全部大写

驼峰原则，首字母小写每个单词首字母大写，起起伏伏就像驼峰

* 1. 类名的命名规则是? 方法名、变量名的命名规则是否一致? 常量的命名规则是?  
     首字母大写，其余的遵循驼峰原则

方法名，变量名规则是一样的

* 1. 引用类型是占用几个字节?  
     4个
  2. 算术运算符中类型提升是怎么回事? a+b返回什么类型? int a=3; long b=3;  
     自动类型转换，因为如果高的转为低的可能会造成精度损失，所以往高转

返回long类型

* 1. 与或非分别是什么含义? 短路是怎么回事? &,|会发生短路吗?  
     与：所有都真，才真 101 $ 011 = 001

或：1个真，就真 101 | 011 = 111  
非：取反 ! 101 = 010

短路就是如果第一个运算已经能知道整个表达式的结果了，后面就没必要计算了

比如与，所有都真才真，只要出现一个假就不用算了，肯定是假

不会，&& ||才是短路或和与

* 1. 5\*4最快的运算方式是?  
     用位运算符来算
  2. 什么情况下，加号会变成字符串连接符?  
     参与运算的其中一方为字符串时，+会自动变为字符串连接操作，但不建议这么做。因为String是Final定义的，在方法区中只有一份，效率极低，推荐使用Stringbuilder或StringBuffer来拼接大量字符串
  3. int能否自动转换成byte,short,char? 是否有一定条件才能转换?  
     不能，向下转型需要强转，而且如果内容大于目标类型，会损失精度
  4. 自动类型转换中，容量小和容量大指的是什么意思?  
     比如a类型只能存放1-10，b类型可以存放1-100

A类型就比b类型小，如果一个b类型变量是20，转换成a类型就会损失精度

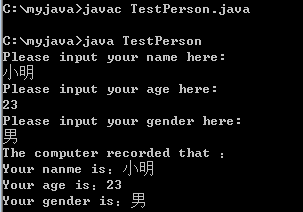
* 1. 布尔类型能否自动转换为int? 如果不能，简述理由。  
     不能，因为boolean存放的是true和false，不能转成任何形式
  2. 下面两种写法，哪个较好：

a. 70L\*60\*24\*365\*70\*20

b. 70\*60\*24\*365\*70\*20L

b，因为后缀一眼就能认出来这是个long，比较符合习惯吧

1. **编码题**
   1. 输入自己的名字，年龄和性别，分别用不同的变量接收，并将输入的信息做输出。

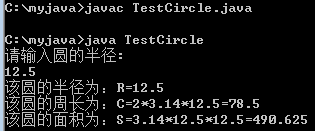


文本

描述已自动生成

* 1. 输入圆形半径，求圆形的周长和圆形的面积,并将结果输出。

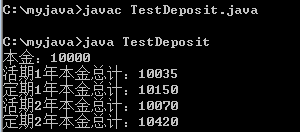
图片包含 图形用户界面

描述已自动生成

* 1. 银行利率表如下表所示，请计算存款10000元，活期1年、活期2年，定期1年，定期2年后的本息合计。



结果如下图所示。（结果四舍五入，不保留小数位。使用Math.round(double d)实现）



文本

描述已自动生成

* 1. 某个公司采用公用电话传递数据，数据是四位的整数，在传递过程中是加密的，加密规则如下：每位数字都加上5,然后用和除以10的余数代替该数字，再将第一位和第四位交换，第二位和第三位交换。结果如图所示。



文本

描述已自动生成

1. **可选题**
   1. 网上查询资料，了解如下内容
      1. ASCII、ISO8859-1、GB2312、GBK、GB18030、BIG5、UNICODE、ANSI等字符集的特点

ASCII：使用7或8位来表示128或256种可能，涵盖大小写字母，数字，标点和一些常用符号

ISO8859-1：向下兼容ASCII，支持部分欧洲文字

GB2312：包含大约7000多简体汉字

GBK：在GB2312上扩展，支持20000个简繁汉字和一些符号

GB18030：继续在GBK基础上扩展，增加一些少数民族文字

UNICODE：各国都推出自己的字符集，所有UNICODE将他们常用的统一起来

ANSI：最早传真机使用，供最早的英文，符号来使用

* + 1. 常用进制（二进制、八进制、十六进制、十进制）的特点及其转换

二进制逻辑型强，适合电路进行开关来表示，剩下的进制各有各的用处，用于表示不同的事务各有优缺点

进制的转换就是除幂操作，不断除以进制数的次方，然后取余

* 1. 一个java源文件可以写多个class吗?编译后，会不会生成多个class文件?

可以，会，内部类

* 1. 中文乱码是怎么造成的?

字符集，编码方式不统一。比如UTF8使用3个字节来表示1个汉字，使用1个字节的字符集，就会吧1个汉字整体，拆成3块，再对应自己的字典显示出来。那必然会乱码

* 1. unicode字符集是几个字节表示一个字符?为什么需要utf-8?

2个字节，utf8是3个字节，字数多了

* 1. 查资料，将java关键字的含义查出（除了strictfp之外）。

Goto特殊保留字，goto就是B语言那个意思，不好用，容易死循环。

Const特殊保留字，没有含义

Abstract抽象声明

Assert已经过时不推荐使用，断言的意思，判断表达式，如果是真就没事，如果是假就抛Error，停止程序

Byte，shrot，int，long，float，double，char，Boolean八大基本变量类型

Break，continue，循环控制语句，一个打断，一个跳过此次

Case：switch用来判断执行

Catch，try，finally，异常的捕获与执行

Public private default protected，访问权限修饰符

Class类声明修饰符

Do while，for循环使用

If else判断使用

Extend，implements继承和实现

Import导包

Instancof判断是否为指定类的实例对象

Interface接口声明

New新建对象

Native不提供Java的实现，由别的程序实现

Null空

Package包

Return返回

Static静态声明

Strictfp精确浮点，使浮点运算更精确

Super父类引用

synchronized同步

this本类

throw，throws抛异常，一个具体的抛，一个往外抛

transient序列化的时候不序列化此对象，好像是？不太记得了

void无返回值

volatile一个稍微不那么严谨的同步

* 1. 关系运算符中，能不能这么写："1<a<3"?

不能，分开写，用或或与连接

* 1. 这两个表达式分别返回什么结果? ((1<3)?"a":"b")+3+4， x=-2; x > 0 ? 1 : (x == 0 ? 0 : -1)

第一个返回字符串“b34”

第二个返回-1

* 1. 4&5，4|5的结果分别是多少? 4&&5这个操作可行吗?为什么?

4&5 = 100 & 101 = 100 = 4

4|5 = 100 | 101 = 101 = 5

不可以，&&只是逻辑运算符，没有位运算符的含义

* 1. long能自动转换成int吗？long能自动转换成float吗?

不可以，高精度往低精度转换，都要手动强制转换

* 1. 使用Scanner接收键盘输入，是否一定要加import java.util.\*; ?

是的，不然Scanner类在哪定义的，Jvm不知道

* 1. 强制类型转换中，是否可能发生数据丢失?

是的

* 1. 利用”异或”运算的性质,对几个字符进行加密并输出密文,然后再解密。加密算法是：密钥是字符’8’，明文的每个字符和密钥进行异或运算，得到密文。密钥和密文的每个字符再次进行异或运算，重新得到明文。结果如图所示。



图形用户界面, 文本

描述已自动生成

* 1. 翻译如下文章(英文对于我们软件开发人员来说很重要，所以我们一开始就要加大对英文的训练)：

|  |
| --- |
| I originally approached Java as "just another  programming language," which in many senses it is.  But as time passed and I studied it more deeply, I began to see that the  我最初接触Java时，它在各方面都让我感觉只是一个不同的编程语言。但随着时间的流逝，随着我学习的深入，在当时，我开始发现  fundamental intent of this language was different from other languages I had  seen up to that point.  这个语言和别的我见过的语言是不同的  Programming is about managing complexity: the complexity of the problem  编程是处理复杂问题的，正如你想解决一个复杂问题，  you want to solve, laid upon the complexity of the machine in which it is solved.  依赖于你解决问题的机器复不复杂  Because of this complexity, most of our programming projects fail.  正因为这个复杂性，你的很多项目失败了  And yet, of all the programming languages of which I am aware, almost none have gone all out and decided that their main design goal would be to conquer the complexity of developing and maintaining programs.  然而，所有我知道的编程语言中，几乎没有gone all out and decided（不知道怎么翻译）他们主要的设计目的是为了解决复杂的开发维护问题  Of course, many language design decisions were made with complexity in mind, but at some point there were always other issues that were considered essential to be added into the mix. Inevitably, those other issues are what cause programmers to eventually "hit the wall" with that language.  当然，很多语言被设计用来，但在某些情况下，总有一些奇怪的问题considered essential to be added into the mix（不懂），最终，这些问题造成了程序员在使用这些编程语言时想“撞墙”  For example, C++ had to be backwards-compatible with C (to allow easy migration for C programmers), as well as efficient.  举个例子，C++必须兼容C（使一些C语言工程师更易转型），同时也保证了效率  Those are both very useful goals and account for much of the success of C++,  这些使得C++设计的很好，变得很成功useful goals and account不知道  but they also expose extra complexity that prevents some projects from being finished (certainly, you can blame programmers and management,  但也暴露了额外的一些复杂性？（不是很理解）阻碍了一些程序的正常执行（当然你可以责备程序员或者经理  but if a language can help by catching your mistakes, why shouldn't it?).  但如果，一个编程语言可以帮助你发现错误，为什么不用它呢？）  As another example, Visual BASIC (VB) was tied to BASIC, which wasn't really designed to be an extensible language, so all the extensions piled upon VB have produced some truly unmaintainable syntax.  另一个例子，VB语言，与B语言强关联，他并没有想设计成一个扩展性的语言。所以，当所有的扩展堆积到VB上时，VB的语法变得不可维护。  Perl is backwards-compatible with awk, sed, grep, and other Unix tools it was meant to replace, and as a result it is often accused of producing "write-only code"(that is, after a while you can't read it).  Perl是向后兼容awk、sed、grep和其他一些Unix工具的。他的初衷是取而代之，结果是它经常告知“只读代码”（就是这样，一阵子以后，你不可以阅读代码）  On the other hand, C++, VB, Perl, and other languages like Smalltalk had some of their design efforts focused on the issue of complexity and as a result are remarkably successful in solving certain types of problems.  另一方面，C++，VB，perl，和一些其他的语言，像是Smalltalk等。在复杂度的设计上下了一些功夫。结果是在处理特定问题时可圈可点  What has impressed me most as I have come to understand Java is that  somewhere in the mix of Sun's design objectives, it seems that there was a goal of reducing complexity for the programmer.  真正令我印象深刻的是，我开始理解Sun公司的设计目标，他们为了减轻程序员事物的复杂性而设计。  As if to say, "We care about reducing the time and difficulty of producing robust code." In the early days, this goal resulted in code that didn't run very fast (although this has improved over time), but it has indeed produced amazing reductions in development time—half or less of the time that it takes to create an equivalent C++ program. This result alone can save incredible amounts of time and money, but Java doesn't stop there.  好像是说“我们关心降低大型代码的难度和时间”。在早一些时候，这种追求导致了程序运行缓慢（尽管它后来提升了很多），但它确实降低了很多的开发时间—几乎是写一个等效C++程序的一半时间。可以节约了不可想象的时间和金钱。但java远不止于此  It goes on to wrap many of the complex tasks that have become important, such as multithreading and network programming, in language features or libraries that can at times make those tasks easy.  它更进一步，将很多复杂、重要的任务做了包装。比如多线程和网络编程。java程序功能和依赖能更简单的解决那些问题  And finally, it tackles some really big complexity problems: cross-platform programs, dynamic code changes, and even security, each of which can fit on your complexity spectrum anywhere from "impediment" to "show-stopper."  最终，他铲除了程序复杂的这座大山：跨平台程序，动态代码变动，甚至安全性。这些每一个问题，能让你complexity spectrum anywhere from "impediment" to "show-stopper."（大概意思应该是劝退）  So despite the performance problems that we've seen, the promise of Java is tremendous: It can make us significantly more productive programmers.  所以尽管我们发现了很多性能上的问题。Java的承诺仍然是巨大的：它能极大提升我们的生产力  In all ways—creating the programs, working in teams, building user interfaces to communicate with the user, running the programs on different types of machines, and easily writing programs that communicate across the Internet—Java increases the communication bandwidth between people. I think that the results of the communication revolution may not be seen from the effects of moving large quantities of bits around. We shall see the true revolution because we will all communicate with each other more easily: one-on-one, but also in groups and as a planet.  从各方面看—编写一个程序，与同事的合作，搭建接口与用户的交流，在不同类型的设备上运行程序，和便捷的代码编写，让互联网能互相交流。Java提高了与人交流的宽度。我认为通信革命的结果不单是数据量的增多。我们应该看到真正的革命，就是我们与人交流变得更加的便捷。一对一，也包括小组之间，星体之间（吹牛逼）。  I've heard it suggested that the next revolution is the formation of a kind of global mind that results from enough people and enough interconnectedness. Java may or may not be the tool that foments that revolution, but at least the possibility has made me feel like I'm doing something meaningful by attempting to teach the language.  我曾听说，下次革命是formation of a kind of global mind that results from enough people and enough interconnectedness.不懂Java或许可以，或许不能成为下次革命的（饲料？）但至少，我有责任去传授这一个编程语言 |