第三章

1、Java语言定义了哪几种基本数据类型？   
答：8种基本数据类型。Byte（字节型）, short（短整型）, int（整型）, long（长整型）, float（单精度浮点型）, double（双精度浮点型）, char（字符型）

2、表示整数类型数据的关键字有哪几个？他们各占用几个字节？   
答：byte, short, int, long分别占1, 2, 4, 8个字节

3、单精度浮点float和双精度浮点double的区别是什么？   
答：单精度浮点数的数据位是32位，双精度浮点数的数据位是64位，double的精度是float的两倍

4、字符型常量与字符串常量的主要区别是什么？   
答：字符串常量是用一对单引号括起来的单个字符，字符串常量是用双引号括起来的一串若干个字符（可以是0个）

5、简述Java语言对定义标识符的规定有哪些。   
答：标识符可以由字母、数字和下划线、美元符号等组合而成，标识符必须以字母、下划线或美元符号开头，不能以数字开头

6、Java语言采用何种编码方案？有何特点？   
答：Unicode字符集编码方案，便于西文字符和中文字符的处理

10、写出由键盘输入数据的两种基本格式。   
答：在1.5版本之前，Java用BufferedReader来读取输入数据，在1.5版本之后，Java用

Scanner来读取输入数据

java   
import java.io.\*; //将[java.io](http://java.io/)包里的所有接口或类都导入到javaBean中  
public class Buffer { //公共类缓冲区  
public static void main(String[] args) throws IOException {

//告诉编译器 我这个方法中间可能有些地方 要抛出异常  
String s;   
BufferedReader buf = new BufferedReader(new InputStreamReader(System.in));

//构造一个字符流的缓存，里面存放在控制台输入的字节转换后成的字符。  
str = buf.readLine(); //从这个缓存中读取一行的内容  
}   
}

java   
import java.util.\*; //导入完整的实用工具（Utility）库  
public class Scan {   
public static void main(String[] args) {

/\*

public static void main(String[] args)

public:权限修饰符，权限最大。

static:随着MianDemo类的加载而加载，消失而消失。

void:  没有返回值

main: 函数名，jvm识别的特殊函数名

(String[] args):定义了一个字符串数组参数

\*/  
Scanner sc = new Scanner(System.in); //构造一个Scanner对象实例  
double num = sc.nextDouble();   
}   
}

15、逻辑运算符中“逻辑与、逻辑或”和“简洁与、简洁或”的区别是什么？   
答： 非简洁运算在必须计算完左右两个表达式之后，才取结果值；而简洁运算可能只计算左边的表达式而不计算右边的表达式，即对于 &&，只要左边的表达式为 false，就不计算右边的表达式，则整个表达式为 false；对于 ||，只要左边表达式为 true，就不计算右边表达式，则整个表达式为 true。

18、写出下列表达式的值，设x=3, y = 17, yn = true。

答：(1) x + y \* x--

3 + 17 \* 3； 54

(2) -x \* y + y

-3 \* 17 + 17； -34

(3) x < y && yn

3 < 17 && true; true

(4) x > y || !yn

3 > 17 || false; false

(5) y != ++x ? x : y

17 != 4 ? 4 : 17; 4

(6) y++ / --x

17 / 2; 8

第四章

3、编写一个Java应用程序，输出1~100之间所有既可以被3整除，又可被7整除的数。

**package** test;

**public** **class** Exercise {

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

System.***out***.print("输出1~100之间所有既可以被3整除，又可被7整除的数:");

**for** (**int** i = 1; i <= 100; i++) {

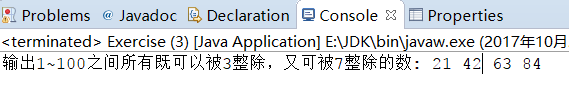
**if** (i % 3 == 0 && i % 7 == 0)//简洁与（均为true，结果才为true）

System.***out***.print(" "+i);

}

}

}



4、编写一个Java应用程序，在键盘上输入数n，计算并输出1!+2!+…+n!的结果

**package** test;

**import** java.util.\*;//导入完整的实用工具（Utility）库

**public** **class** Exercise {

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

System.***out***.print("请输入n: ");

Scanner in = **new** Scanner(System.***in***);//新创建一个输入的Scanner对象，然后赋值给in

**int** n = in.nextInt();

**long** sum = 0, temp = 1;

**for** (**int** i = 1; i <= n; i++) {

temp \*= i;//(temp=temp\*i)

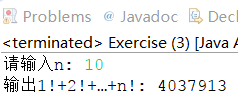
sum+=temp;//(sum=sum+temp)

}

System.***out***.print("输出1!+2!+…+n!: "+sum);

}

}



5、在键盘上输入数n，编程计算sum = 1 - (1 / 2!) + (1/ 3!) -…(-1)^n-1(1/n!)。

**package** test;

**import** java.util.\*;//导入完整的实用工具（Utility）库

**public** **class** Exercise {

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

System.***out***.print("请输入n: ");

Scanner in = **new** Scanner(System.***in***);//新创建一个输入的Scanner对象，然后赋值给in

**int** n = in.nextInt();

**double** sum = 0;

**int** sign = 1, temp = 1;

**for** (**int** i = 1; i <= n; i++) {

temp \*= i;//(temp=temp\*i)

sum += sign \* (1.0/temp);//(sum=sum+sign \* (1.0 / temp))

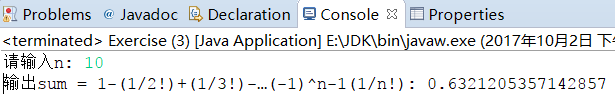
sign = -sign;

}

System.***out***.println("输出sum = 1-(1/2!)+(1/3!)-…(-1)^n-1(1/n!): "+sum);

}

}



10、设有一长为3000m的绳子，每天减去一半，问需几天时间，绳子的长度会短于5m。

**package** test;

**public** **class** Exercise {

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

**int** day = 0;

**double** len = 3000;

**while** (len >= 5) {

len /= 2;

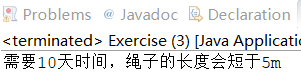
day++;

}

System.***out***.println("需要"+day+"天时间，绳子的长度会短于5m");

}

}



11、编程输出如下的数字图案：

1 3 6 10 15   
2 5 9 14   
4 8 13   
7 12   
11

**package** test;

**public** **class** Exercise {

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

**int**[][] matrix = **new** **int**[5][5];

**int** k = 1;

**for** (**int** i = 0; i < 5; i++) {

**for** (**int** j = 0; j <= i; j++)

matrix[i - j][j] = k++;

}

**for** (**int** i = 0; i < 5; i++) {

**for** (**int** j = 0; j < 5 - i; j++)

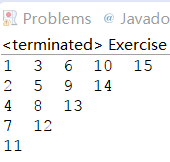
System.***out***.print(matrix[i][j] + " ");

System.***out***.println();

}

}

}



第五章

4、找出4 x 5矩阵中值最小和最大元素，并分别输出其值及所在的行号和列号。

**package** test;

**public** **class** Exercise {

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

**int**[][] matrix;

matrix = **new** **int**[][]{{12,23,34,15,7},{23,14,61,45,78},{3,12,43,54,65},{34,56,87,54,23}};

**int** minRow = 0, minCol = 0, maxRow = 0, maxCol = 0;

**for** (**int** i = 0; i < 4; i++) {

**for** (**int** j = 0; j < 5; j++) {

**if** (matrix[minRow][minCol] > matrix[i][j]) {

minRow = i;//最小行号

minCol = j;//最小列号

}

**if** (matrix[maxRow][maxCol] < matrix[i][j]) {

maxRow = i;

maxCol = j;

}

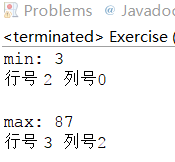
}

}

System.***out***.println("min: " + matrix[minRow][minCol] + "\n行号 " + minRow + " 列号" + minCol + "\n\n" + "max: " + matrix[maxRow][maxCol] + "\n行号 " + maxRow + " 列号" + maxCol);

}

}



5、产生0~ 100之间的8个随机整数，并用冒泡排序将其升序排序后输出（冒泡排序算法：每次进行相邻两数的比较，若次序不对，则交换两数的次序）。

**package** test;

**import** java.util.\*;

**public** **class** Exercise {

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

System.***out***.println("冒泡排序算法：");

**int**[] num = **new** **int**[8];

Random random = **new** Random();

**for** (**int** i = 0; i < 8; i++) {

num[i] = random.nextInt() % 100;

}

**for** (**int** i = 0; i < 7; i++) {

**for** (**int** j = 0; j < 7 - i; j++) {

**if** (num[j] > num[j + 1]) {

**int** temp = num[j];

num[j] = num[j + 1];

num[j + 1] = temp;

}

}

}

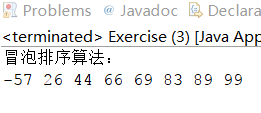
**for** (**int** i = 0; i < 8; i++) {

System.***out***.print(num[i] + " ");

}

}

}



6、15 个红球和 15 个绿球排成一圈，从第一个球开始数，当数到第 13 个球时就拿出此球，然后再从下一个球开始数，当再数到第 13 个球时又取出此球，如此循环进行，直到仅剩 15 个球为止，问怎样排法才能使每次取出的球都是红球。

**package** test;

**public** **class** Exercise {

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

**int**[] ball = **new** **int**[30];

**for** (**int** j = 0, temp = 0; temp <= 15; temp++) {

**int** count = 1, i = j;

**while** (count < 13 || ball[i] == 1) {//(均为false，结果才为false)

**if** (ball[i] != 1)

count++;

i++;

**if** (i == 30)

i = 0;

}

ball[i] = 1;

j = i + 1;

**if** (j == 30)

j = 0;

}

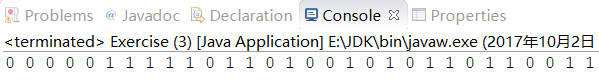
**for** (**int** i = 0; i < 30; i++) {

System.***out***.print(ball[i] + " ");

}

}

}



10、编程统计用户从键盘输入的字符串中所包含的字母，数字和其它字符的个数。

**package** test;

**import** java.util.\*;//导入完整的实用工具（Utility）库

**public** **class** Exercise {

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

System.***out***.print("请输入字符串：");

Scanner in = **new** Scanner(System.***in***);//新创建一个输入的Scanner对象，然后赋值给in

String s = in.nextLine();

**int** ch = 0, digital = 0, other = 0;

**for** (**int** i = 0; i < s.length(); i++) {

**char** c = s.charAt(i);

**if** ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z'))//(&&简洁与，均为true结果才为true；||简洁或，均为false结果才为false)

ch++;

**else** **if** (c >= '0' && c <= '9')

digital++;

**else**

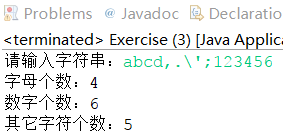
other++;

}

System.***out***.println("字母个数："+ch + "\n" +"数字个数："+ digital + "\n" +"其它字符个数："+ other);

}

}



第六章

1、类与对象的区别是什么？

类是对某一类事物的描述，是抽象的、概念上的定义；对象则是实际存在的属该类事物的具体的个体

3、定义一个类时所使用的修饰符有哪几个？每个修饰符的作用是什么？是否可以混用？

定义一个类时所使用的修饰符有 **4** 个，一个类可以有多个修饰符，且无先后顺序之分，但 abstract 和 final 相互独立，所以不能同时应用在一个类的定义中，每个修饰符的作用如下：

| **修饰符** | **含义** |
| --- | --- |
| **public** | 将一个类声明为公共类，它可以被任何对象访问 |
| **abstract** | 将一个类声明为抽象类，没有实现方法，需要子类提供方法的实现，  所以不能创建该类的实例 |
| **final** | 将一个类声明为最终类即非继承类，表示它不能被其他类所继承 |
| **缺省** | 缺省修饰符时，则表示只有在相同包中的对象才能使用这样的类 |

5、成员方法的修饰符有哪些？各修饰符的功能是什么？是否可以混用？

成员方法的修饰符有 **9** 个，成员方法与成员变量同样有多个控制修饰符，当用两个以上的修饰符来修饰一个方法时，需要注意，有的修饰符之间是互斥的，所以不能同时使用。

| **修饰符** | **含义** |
| --- | --- |
| **public** | 公共访问控制符。指定该方法为公共的，它可以被任何对象的方法访问 |
| **private** | 私有访问控制符。指定该方法只允许自己类的方法访问，  其他任何类（包括子类）中的方法均不能访问此方法 |
| **protected** | 保护访问控制符。指定该方法只可以被它的类及其子类或同一包中的其他类访问 |
| **缺省** | 缺省访问控制符。表示在同一个包中的其他类可以访问此成员方法，  其他包中的类不能访问该成员方法 |
| **final** | 最终修饰符。指定该方法不能被重载 |
| **static** | 静态修饰符。指定不需要实例化一个对象就可以激活的方法 |
| **abstract** | 抽象修饰符。指定该方法只声明方法头，而没有方法体，抽象方法需在子类中被实现 |
| **synchronized** | 同步修饰符。在多线程程序中，该修饰符用于在运行前，  对它所属的方法加锁，以防止其他线程访问，运行结束后解锁 |
| **native** | 本地修饰符。指定此方法的方法体是用其他语言（如 C）在程序外部编写的 |

6、成员变量与局部变量的区别有哪些？

由类和方法的定义可知，在类和方法中均可定义属于自己的变量。类中定义的变量是成员变量，而方法中定义的变量是局部变量。

1. 从语法形式上看，成员变量是属于类的，而局部变量是在方法中定义的变量或是方法的参数；成员变量可以被 public、private、static 等修饰符所修饰，而局部变量则不能被访问控制符及 static 所修饰；成员变量和局部变量都可以被 final 修饰。

2. 从变量在内存中的存储方式上看，成员变量是对象的一部分，而对象是存在与堆内存中的，而局部变量是存在与栈内存中的。

3. 从变量在内存中的生存时间上看，成员变量是对象的一部分，它随着对象的创建而存在，而局部变量随着方法的调用而产生，随着方法调用的结束而自动消失。

4. 成员变量如果没有被赋初值，则会自动以类型的默认值赋值（有一种情况例外，被 final 修饰但没有被 static 修饰的成员变量必须显式的赋值）；而局部变量则不会自动赋值，必须显式地赋值后才能使用。

7、创建一个对象使用什么运算符？对象实体与对象的引用有何不同？

创建一个对象使用 **new** 运算符；对象实体是实在存在于**堆内存**中的，而对象的引用是管理对象实体的句柄存在于**栈内存**中。

9、在成员变量或成员方法前加上关键字 this 表示什么含义？

this.成员名表示对象本身的成员。this代表调用此成员的对象。

10、什么是方法的返回值？返回值在类的方法里面的作用是什么？

方法的返回值是指我们获取到的某个方法体中的代码执行后产生的结果！（前提是该方法可能产生结果）。返回值的作用:接收产生的结果，使得它可以用于其他的操作。

第七章

1、一个类的公共成员与私有成员有何区别？

私有成员无法从该类的外部访问到该类内部的成员，而只能被该类自身访问和修改，而不能被任何其他类包括该类的子类来获取或引用；公共成员则可以被其他的类访问。

2、什么是方法的重载？

重载是指在同一个类内具有相同名称的多个方法，这多个同名方法如果参数个数不同，或者是参数个数相同，但类型不同，则这些同名的方法就具有不同的功能。方法的重载是实现“多态”的一种方法。

3、一个类的构造方法的作用是什么？若一个类没有声明构造方法，该程序能正确执行吗？为什么？

构造方法的作用是在对象被创建时初始化对象的成员的方法；如果一个类没有声明构造方法，该类也可以被正确实例化，Java 编译器会自动为该类生成一个默认的构造方法。

4、构造方法有哪些特性？

1. 构造方法的方法名与类名相同

2. 构造方法没有返回值，但不能加 void

3. 构造方法的主要作用是完成对类对象的初始化工作

4. 构造方法一般不能由编程人员显式地直接调用，而是用 new 来调用

5. 在创建一个类的对象的同时，系统会自动调用该类的构造方法为对象初始化

5、在一个构造方法内可以调用另一个构造方法吗？如果可以，如何调用？

可以；Java 语言允许在类内从某一构造方法内调用另一个构造方法；在某一构造方法内调用另一个构造方法时，必须使用 this 关键字来调用，否则编译时将出现错误，而且 this 关键字必须写在构造方法内的第一行的位置。

7、静态变量与实例变量有哪些不同？

用 static 修饰符修饰的成员变量称为“静态变量”，静态变量也称类变量。

静态变量是隶属于类的变量，而不是属于任何一个类的具体变量，静态变量不需要实例化就可以使用。

实例变量是隶属于对象的变量，是属于具体的一个对象的，是需要把类实例化为对象才可以使用的。

8、在一个静态方法内调用一个非静态成员为什么是非法的？

静态方法是属于整个类的，所以它不能操纵和处理属于某个对象的成员，而只能处理属于整个类的成员，即静态方法只能访问静态成员变量或静态成员方法。

9、对象的相等与指向它们的引用相等，两者有什么不同？

对象的相等指的是对象的内容相等；

指向它们的引用相等指的是引用变量指向的地址相同；

== 操作符专门用来比较两个变量的值是否相等；

equals 方法是用于比较两个独立对象的内容是否相同。