# 实验2：方法与数组

## 一、实验目的和要求

**掌握方法的定义、调用及传值**

**掌握方法的重载**

**理解不同变量的作用域**

**掌握数组的声明、创建、引用方法**

**掌握使用下标访问数组元素的方法**

**掌握数组常用的操作**

**掌握数组作方法参数及返回值**

## **二、实验内容与结果及分析**

### P198 6.3

原题：（回文整数）使用下面的方法编写两个方法：

//return the reveral of an inter, i.e.,reverse(456) returns 654

Public static int reverse(int number)

//return true if number is a palindrome

Public static Boolean isPalindrome(int number)

源代码：

**package** zuoye;

**import** java.util.Scanner;

**public** **class** T1 {

//P198 6.3

**public** **static** **int** reverse(**int** number){

**int** b = 0;

**int** i = 0;

**int** flag = number;

**do**

{

**if**(number==0)

**break**;

number/=10;

i++;

}**while**(number!=0);

number = flag;

**int** j = i - 1;

String s = number+"";

**for**(**int** k=(i-1); k>=0; k--,j--)

{

b += ((s.charAt(k)-'0')\*Math.*pow*(10, j));

}

**return** b;

}

**public** **static** **boolean** isPalindrome(**int** number){

**if**(number==0)

**return** **true**;

**else**

**return** **false**;

}

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

// **TODO** Auto-generated method stub

Scanner in = **new** Scanner(System.***in***);

System.***out***.println("请输入一个整数：");

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

**if**(*isPalindrome*(*reverse*(a)-a))

{

System.***out***.println(a+"是回文整数");

}

**else**

{

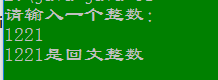
System.***out***.println(a+"不是回文整数");

}

}

}

结果及截图：



### P236 7.3

原题：（计算数字出现的次数）编写程序，读取在1到100之间的整数，然后计算每个数出现的次数。假定输入是以0为结束的。下面的这个程序的一个运行实例：

Enter the integers between 1 and 100 : 2 5 6 4 3 23 43 2 0

2 occurs 2 times

3 occurs 1 times

4 occurs 1 times

5 occurs 2 times

6 occurs 1 times

23 occurs 1 times

43 occurs 1 times

源代码：

package zuoye;

import java.util.Arrays;

import java.util.Scanner;

public class T2 {

public static void main(String[] args) {

// P236 7.3

int i = 0;

int b = 0;

int m = 0;

int[] c = new int[100];

Scanner in = new Scanner(System.in);

System.out.println("请输入数据，输入0结束：");

while(true){

b = in.nextInt();

if(b==0)

break;

c[i++] = b;

}

int[] a = new int[i];

for(int n=0; n<a.length;n++){

a[n] = c[n];

}

Arrays.sort(a);

for(int j=0; j<a.length; j++){

m=1;

if(j>0&&a[j]==a[j-1])

continue;

for(int k=j+1; k<a.length; k++){

if(a[k]==a[j])

m++;

}

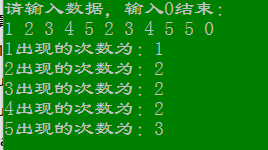
System.out.println(a[j]+"出现的次数为："+m);

}

}

}

结果及截图：



### P240 7.30

原题：（模式识别方面： 四个连续相等的数）编写下面的方法，测试某个数组是否有四个连续的值相同的数字。

Public static Boolean isConsecutiveFour(int [] values)

编写测试程序，提示用户输入一个整数列表，如果这个列表中有四个连续的具有相同值的数，那就显示true；否则，显示false。程序应该首先提示用户键入输入的大小，即列表中值的个数。这里是一个运行实例。

Enter the number of value :8

Enter the values : 3 4 5 5 5 5 4 5

The list has consecutive fours

Enter the number of values : 9

Enter the values :3 4 5 5 6 5 5 4 5

The list has no consecutive fours

源代码：

**package** zuoye;

**import** java.util.Scanner;

**public** **class** T3 {

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

//P240 7.30

Scanner in = **new** Scanner(System.***in***);

**int** i = 0;

**int** b = 0;

**int** m = 0;

**boolean** flag = **false**;

System.***out***.println("请输入数的数量：");

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

**int**[] c = **new** **int**[100];

**while**(i<num){

b = in.nextInt();

c[i++] = b;

}

**int**[] a = **new** **int**[i];

**for**(**int** n=0; n<a.length;n++){

a[n] = c[n];

}

**for**(**int** j=0; j<a.length; j++){

m=0;

**for**(**int** k=j; k<a.length-1; k++){

**if**(a[k]==a[k+1])

m++;

}

**if**(m==3)

flag=**true**;

}

**if**(flag)

System.***out***.println("存在四个连续整数");

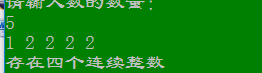
**else**

System.***out***.println("不存在四个连续整数");

}

}

结果及截图：



### P260 8.7

原题：（距离最近的两个点）程序清单8-3给出二维空间中距离最近的两个点的程序。修改该程序，让程序能够找出在三维空间上距离最近的两个点。使用一个二维数组表示这些点。使用下面的点来测试这个程序：

Double[][] points = {{-1,0,3},{-1,-1,-1},{4,1,1},{2,0.5,9},{3.5,2,-1},{3,1.5,3},{-1.5,4,2},{5.5,4,-0.5};

源代码：

**package** zuoye;

**import** java.util.Scanner;

**public** **class** T4 {

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

// P260 8.7

Scanner in = **new** Scanner(System.***in***);

System.***out***.println("Enter the number of points:");

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

**double**[][] points = **new** **double**[numberofPoints][3];

System.***out***.println("Enter "+numberofPoints + " points:");

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

{

points[i][0] = in.nextDouble();

points[i][1] = in.nextDouble();

points[i][2] = in.nextDouble();

}

**int** p1 = 0,p2 = 1;

**double** shortDistance = *distance*(points[p1][0],points[p1][1],points[p1][2],points[p2][0],points[p2][1],points[p2][2]);

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

{

**for**(**int** j=i+1; j<points.length; j++)

{

**double** distance = *distance*(points[i][0],points[i][1],points[i][2],points[j][0],points[j][1],points[j][2]);

**if**(shortDistance > distance)

{

p1 = i;

p2 = j;

shortDistance = distance;

}

}

}

System.***out***.println("The closest two points are "+"("+points[p1][0]+", "+points[p1][1]+", "+points[p1][2]+")and("+

points[p2][0]+", "+points[p2][1]+", "+points[p2][2]+")");

}

**public** **static** **double** distance(**double** x1,**double** y1,**double** z1,**double** x2,**double** y2,**double** z2)

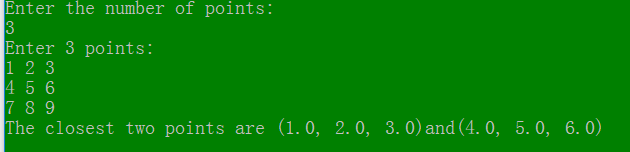
{

**return** Math.*sqrt*((x2-x1)\*(x2-x1)+(y2-y1)\*(y2-y1)+(z1-z2)\*(z1-z2));

}

}

结果及截图：



### P262 8.14

原题：（探索矩阵）编写程序，提示用户输入一个方阵的长度，随机地在矩阵中填入0和1，打印这个矩阵，然后着找出整行、整列或者对角线都是0或1的行列和对角线。下面是这个程序的一个运行示例：

Enter the size for the matrix:4

0111

0000

0100

1111

All 0s on row 1

All 1s on row 3

No same numbers on a column

No same numbers on the major diagonal

No same numbers on the sub-diagonal

源程序：

**package** zuoye;

**import** java.util.Scanner;

**public** **class** T5 {

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

//262 8.14

Scanner in = **new** Scanner(System.***in***);

**int** sum1 = 0;

**int** sum2 = 0;

System.***out***.println("请输入矩阵行列数:");

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

**int**[][] a = **new** **int** [n][n];

System.***out***.println("请输入矩阵元素:");

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

{

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

{

a[i][j] = in.nextInt();

}

}

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

{

**int** sum = 0;

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

{

sum += a[i][j];

}

**if**(sum==n)

System.***out***.println("all 1 is in row"+i);

**else** **if**(sum==0)

System.***out***.println("all 0 is in row"+i);

}

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

{

**int** sum = 0;

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

{

sum += a[j][i];

}

**if**(sum==n)

System.***out***.println("all 1 is in column"+i);

**else** **if**(sum==0)

System.***out***.println("all 0 is in column"+i);

}

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

{

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

{

**if**(i==j)

sum1 += a[i][j];

**if**((i+j)==(n-1))

sum2 += a[i][j];

}

**if**(sum1==n)

System.***out***.println("all 1 is in diagonal");

**if**(sum1==0)

System.***out***.println("all 0 is in diagonal");

**if**(sum2==n)

System.***out***.println("all 1 is in counter-diagonal");

**if**(sum2==0)

System.***out***.println("all 0 is in counter-diagonal ");

}

}

}

结果及截图：

