

# Java 第二次作业

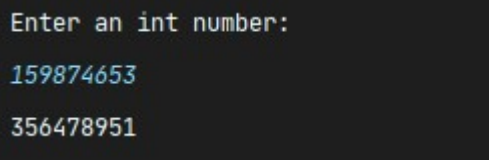
## 6.4 (反向显示一个整数)

使用下面的方法头编写方法，反向显示一个整数： `public static void reverse(int number)` 例如：`reverse(3456)`返回6543。编写一个测试程序，提示用户输入一个整数，然后显示 它的反向数。

```
In [1]: import java.util.Scanner;
public class reverseNumber{
    public static void reverse(int number) {
        while (number > 10) {
            System.out.print(number % 10);
            number = number / 10;
        }
        System.out.print(number);
    }

    public static void main(String[] args) {
        // 输入一个数
        System.out.println("Enter an int number: ");
        Scanner input = new Scanner(System.in);
        int number = input.nextInt();
        // 调用reverse翻转
        reverse(number);
    }
}
```

### 运行结果



```
Enter an int number:
159874653
356478951
```

## 6.17 (显示0和1构成的矩阵)

编写一个方法，使用下面的方法头显示  $n \times n$  的矩阵： `public static void printMatrix(int n)` 每个元素都是随机产生的0 或1。编写一个测试程序，提示用户输入n, 显示如下所示的  $n \times n$  矩阵：

```
In [22]: import java.util.Scanner;

public class generateMatrix {
    public static void printMatrix(int n) {
        int i, j;
        int k;
        for (i = 0; i < n; i++) {
            for (j=0; j < n; j++) {
                k = Math.random() > 0.5? 1:0; // 随机生成01值
                System.out.print(k);
                System.out.print(' ');
            }
            System.out.print('\n'); // 换行
        }
    }

    public static void main(String[] args) {
        System.out.println("Enter an int number: ");
        Scanner input = new Scanner(System.in); // 输入nxn矩阵大小
    }
}
```

```

        int number = input.nextInt();
        printMatrix(number);
    }
}

```

## 运行结果

```

Enter an int number:
4
0 0 0 1
0 1 0 0
0 1 1 0
1 1 1 0

```

## 7.7 (统计一位数的个数)

编写一个程序，生成0和9之间的100个随机整数，然后显示每一个数出现的次数。

```

In [19]: int i,k;
int[] count = new int[10];
for (i=0; i<100; i++) {
    k = (int) (Math.random() * 10);
    System.out.print(k);
    count[k] += 1; // count数组中统计出现次数
}
System.out.println();
// 打印count
for (i=0; i<10; i++){
    System.out.println("第" + (i+1) + "个数的出现次数: " + count[i]);
}

```

```

99611302058089046548857223976125281121270836736505832179107028539672410880506190373013
02986775899109
第1个数的出现次数: 15
第2个数的出现次数: 12
第3个数的出现次数: 11
第4个数的出现次数: 9
第5个数的出现次数: 3
第6个数的出现次数: 9
第7个数的出现次数: 8
第8个数的出现次数: 10
第9个数的出现次数: 12
第10个数的出现次数: 11

```

## 7.10 (找出最小元素下标)

编写一个方法，求出整数数组中最小元素的下标。如果这样的元素个数大于1，则返回最小的下标。使用下面的方法头：public static int indexOfSmallestElement(double[] array) 编写测试程序，提示用户输入10个数字，调用这个方法，返回最小元素的下标，然后显示这个下标值。

```

In [20]: import java.util.Scanner;
public class indexSmall {
    public static int indexOfSmallestElement(int[] array) {
        int i;
        int index = 0; // 存储最小值位置
        for (i = 0; i < 10; i++) {
            if (array[i] < array[index]) { // 只有当比原数小时才更新
                index = i;
            }
        }
    }
}

```

```

    }
    return index;
}

public static void main(String[] args) {
    int[] myList = new int[10];
    int i;
    // 用户输入10 个数字
    for (i = 0; i < 10; i++) {
        System.out.print("Enter the " + (i+1) + " number: ");
        Scanner input = new Scanner(System.in);
        myList[i] = input.nextInt();
    }
    int result = indexOfSmallestElement(myList); // 接受方法返回值并输出
    System.out.print("最大值为第" + (result+1) + "个数");
}
}

```

## 运行结果

```

Enter the 1 number: 1
Enter the 2 number: 4
Enter the 3 number: 6
Enter the 4 number: 8
Enter the 5 number: 2
Enter the 6 number: 1
Enter the 7 number: 9
Enter the 8 number: 1
Enter the 9 number: 7
Enter the 10 number: 7
最大值为第1个数

```

## 8.2（求矩阵主对角线元素的和）

编写一个方法，求 $n \times n$  的double 类型矩阵中主对角线上所有数字 的和，使用下面的方法头：

public static double sumMajorDiagonal (double[][] m) 编写一个测试程序，读取一个4 x 4 的矩阵，然后显示它的主对角线上的所有元素的和。下面是一个运行示例：

```

In [21]: import java.util.Scanner;

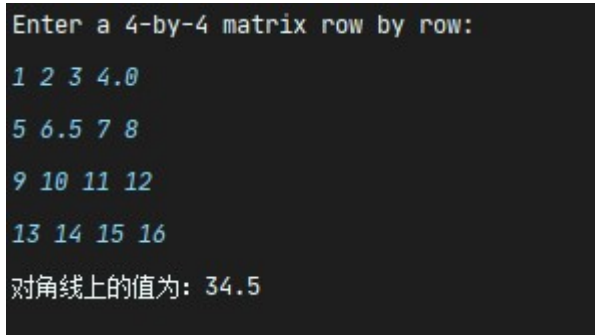
public class countMatrixDiagonal {
    public static double sumMajorDiagonal (double[][] m) {
        double sum_reult = 0;
        int i;
        for (i=0; i<m.length; i++) {
            sum_reult += m[i][i];
        }
        return sum_reult;
    }
    public static void main(String[] args) {
        double[][] myMat = new double[4][4];
        int i,j;
        System.out.println("Enter a 4-by-4 matrix row by row: ");
        for (i = 0; i < 4; i++) {
            Scanner input = new Scanner(System.in);
            String number = input.nextLine();

```

```
String[] strs = number.split("\\ "); // 把一行输入的数分开
for (j=0; j<4; j++) {
    myMat[i][j] = new Double(strs[j]);
}
}
double sum_result = sumMajorDiagonal(myMat); // 计算对角线上的值

System.out.println("对角线上的值为: " + sum_result);
}
```

## 运行结果



```
Enter a 4-by-4 matrix row by row:
1 2 3 4.0
5 6.5 7 8
9 10 11 12
13 14 15 16
对角线上的值为: 34.5
```