

Java第一次实验报告

学院：信息学院
专业：计算机科学与技术
姓名：李哲彦
学号：37220222203675
时间：2024年4月7日

实验题目

1. 庆祝厦门大学校庆

解题思路：

首先根据图片样式，手动调整空格数量，直接打印标题。
然后将内容部分视为三个不同的字母，在同一行输出时根据各自的形状特征分别进行调整。

源代码核心部分解析

首先，在外层，通过变量 i 循环每一行，然后对每个字母的部分进行分别讨论。

字母 I:

```
for(int j = 1; j ≤ 5; j++)
    System.out.print(" ");
for(int j = 1; j ≤ 3; j++)
    System.out.print("*");
for(int j = 1; j ≤ 5; j++)
    System.out.print(" "); // 绘制 I
```

其特征为每一行左右两侧是空格，中间固定为 3 个 '*'。故直接循环输出即可。

爱心

然后是爱心。根据其图形特征，可以发现其为轴对称图形，只需要获得左侧 13 个格子，然后倒着输出即为右侧 13 个格子。故此处定义一字符串存储左侧部分：

```
char[] s = new char[14];
```

然后根据其特征，将爱心可大致分为 1 – 2 行、3 – 5 行以及 9 – 13 行。
对于前两行，发现其空格为等差数列，可以用式子进行计算：

```
if(i ≤ 2){
    int now = 1;
    for(int j = 1; j ≤ 5 - 2 * i; j++){
        s[now++] = ' ';
    }
    for(int j = 1; j ≤ 5 * i; j++){
        s[now++] = '*';
    }
    for(int j = now; j ≤ 13; j++){
        s[now++] = ' ';
    }
}
```

3 – 5 行全部为 * 符号，故：

```
else if(i ≤ 5){
    for(int j = 1; j ≤ 13; j++)
        s[j] = '*';
}
```

发现最后几行的左侧空格的数量差有规律，故用一变量 *del* 来进行调整：

```
int now = 1;
if(i ≤ 9) del++;
else if(i ≤ 11 || i == 13) del += 2;
else if(i == 12) del += 3;

for(int j = 1; j ≤ del; j++){
    s[now++] = ' ';
}
while(now ≤ 13){
    s[now++] = '*';
}
```

最后，利用对称性输出：

```
for(int j = 1; j ≤ 13; j++)
    System.out.print(s[j]);
if(i ≤ 3) System.out.print(" ");
else System.out.print("*");
```

```

for(int j = 13; j ≥ 1; j--)
    System.out.print(s[j]); // 对称for(int j = 1; j ≤ 13; j++)
    System.out.print(s[j]);
if(i ≤ 3) System.out.print(" ");
else System.out.print("*");
for(int j = 13; j ≥ 1; j--)
    System.out.print(s[j]); // 对称

```

字母U

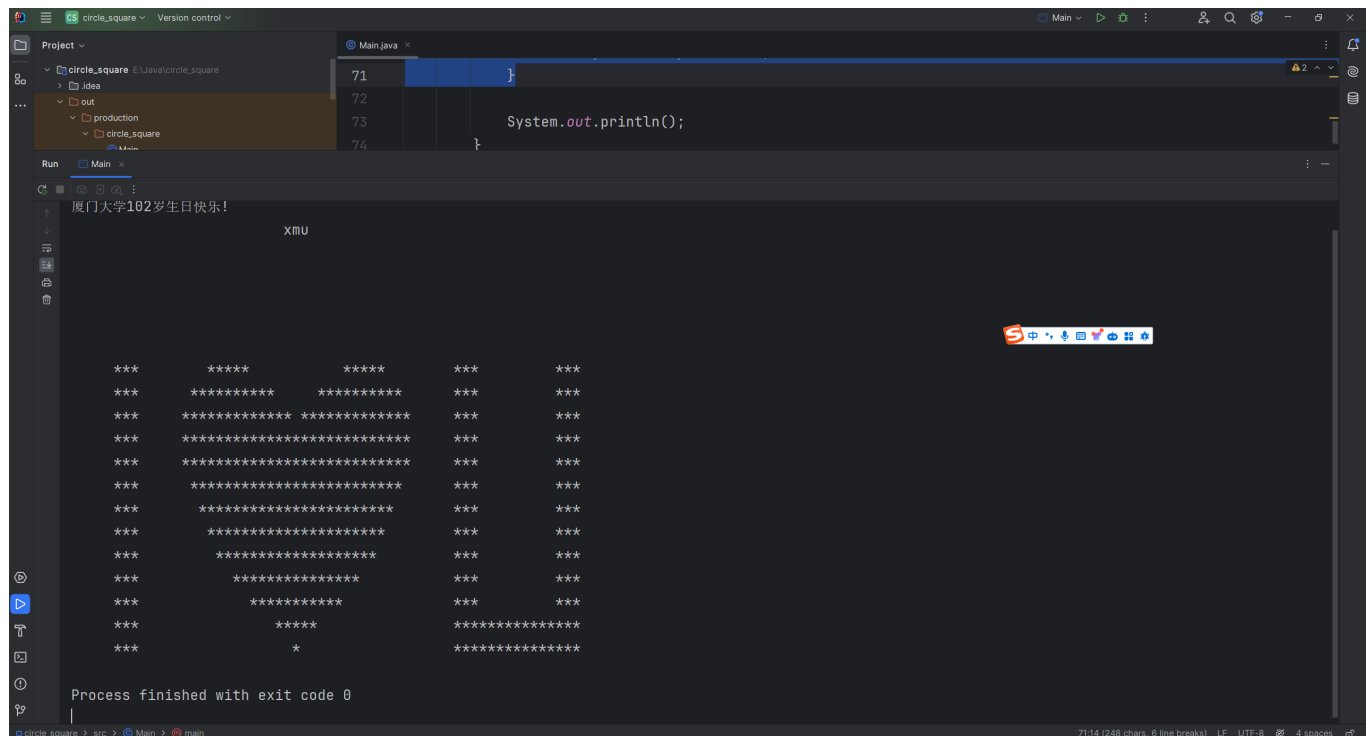
最后输出字母 *U*,规律较为简单。

```

for(int j = 1; j ≤ 15; j++){
    if(i ≤ 11){
        if(j ≤ 3 || j ≥ 13) System.out.print("*");
        else System.out.print(" ");
    }
    else System.out.print("*");
}

```

运行效果：



The screenshot shows an IDE window with a project named 'circle_square'. The code editor displays a Java file 'Main.java' with the following code:

```

71
72
73 System.out.println();
74

```

The Run console shows the output of the program:

```

厦门大学102生日快乐!
XMU

```

The output is a large ASCII art heart shape made of asterisks, with the text '厦门大学102生日快乐!' and 'XMU' above it. The IDE interface includes a project explorer, a code editor, and a run console.

调试流程与心得

第一次尝试输出爱心时，我并没有想到利用其对称性进行绘图，而是选择一次性画完一整行。虽然理论上可行，但是这大大加大了代码错误的概率，并且也会严重降低代码的易读性。在

写完爱心部分后，输出并不正确，非常混乱。在想到利用其对称性质后，大大减少了代码长度，也降低了错误的概率。

通过这道题，我收获了一个宝贵的经验：在写代码之前，一定要想好在写。先多发现一些题目的性质，降低代码的难度，设计好了代码的总体的结构后再开始编码。

水仙花数

解题思路

思路1

直接用 *for* 循环，判断所有三位数，然后依次取出所有位数进行判断：

```
for(int i = 100; i ≤ 999; i++){
    int tmp = i;
    int sum = 0;
    while(tmp > 0){
        int x = tmp % 10;
        sum += x * x * x;
        tmp /= 10;
    }
    if(sum == i){ // 是水仙花数
        ans++;
        G[++top] = i;
    }
}
```

时间复杂度： $O(N)$ 。因为每一个数字的位数都是 3，为常数，且只用了一层 *for* 循环枚举所有三位数，效率较高。

思路2

用 *dfs* 递归方法，依次枚举选择哪 3 位数字，然后复原出原数字进行比较。最后注意特判前缀 0 的情况即可。

```
public static void dfs(int now){
    if(now > 3){
        int x = a[1] * 100 + a[2] * 10 + a[3]; // 复原出原数字
        if(x < 100) return ; // 特判前缀 0 的情况
        int sum = 0;
        for(int i = 1; i ≤ 3; i++){
            sum += a[i] * a[i] * a[i];
        }
        if(sum == x){
            // ...
        }
    }
}
```

```

        G[++ans] = sum; // 存储答案
    }
    return ;
}
for(int i = 0; i ≤ 9; i++){
    a[now] = i;
    dfs(now + 1);
}
return ;
}

```

由于依然要遍历所有的三位数，所以时间复杂度依然为 $O(N)$ 。由于使用了递归算法，在函数调用等方面的开销会大于第一种方法，效率略低于前者。

运行结果如下图：

```

1 //TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
2 // click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
3 public class Main {
4     static int[] a = new int[4];
5     static boolean[] vis = new boolean[11];
6     static int ans = 0;
7     static int[] G = new int[1000];
8     public static void dfs(int now){
9         if(now > 3){
10             int x = a[1] * 100 + a[2] * 10 + a[3];
11             if(x < 100) return ;
12             int sum = 0;
13             for(int i = 1; i <= 3; i++)

```

Run Main x

```

C:\Users\lzy\.jdk\corretto-1.8.0_402\bin\java.exe ...
4
153 370 371 407
Process finished with exit code 0

```

调试流程与心得

由于循环写法在之前已用别的语言训练过多次，故没有什么调试过程。

对于第二种写法，一开始我忘记了特判前缀 0 的情况，导致了最终输出了 6 个水仙花数：多了 0 和 1 两个错误的答案。他们显然不是 3 位数。所以在最终判断时，要特判复原数字小于 100 的情况。

通过这次调试，我对特殊情况的重要性有了更加清晰的认知。在很多问题下，如果不考虑特殊情况，就会导致答案出错，甚至造成在某些情况下程序运行错误的后果。这道题目就是前者的情况。在以后的代码过程中，一定要多加注意特殊情况的考虑。

3. 设计一个简化的取棍游戏

解题思路

本体思路较为简单，按照题意模拟即可。此处着重记录几个重要部分。

输入整数检查

首先，对于输入时的整形判断，由于对于每一个输入的整数都要进行此操作，故在此将其封装为一个方法，通过强制类型转换判断其是否为整数：

```
public static int Check_input(String s){
    System.out.println(s);
    double x = input.nextDouble();
    while(x != (int)x){
        System.out.println("Please input an int, not a double!");
        System.out.println(s);
        x = input.nextDouble();
    }
    return (int)x;
}
```

其中传入的参数 s 为输入前的提示，以方便在输入非整数后重新进行提示，增加该方法的通用性。

当前操作角色切换

用户输入先手操作的顺序后，用一个变量 $user$ 表示当前操作的角色(0 表示电脑，1 表示用户)。每次操作结束后，将其异或 1 即可方便地表示切换用户。在最后判断胜负的时候，需要注意题目要求的是最后一个拿走木棍的。所以出循环之后的结果应该与 $user$ 变量反过来。

```
if(user == 1) System.out.println("You win!");
else System.out.println("You lose!");
```

电脑的最优操作

运用博弈论知识：设 f_i 表示在双方都足够聪明，永远按照最优情况进行游戏的情况下，还剩下 i 根木棍时进行操作是必败还是必胜。根据游戏规则， f_1 为必败状态，故 f_2, f_3, f_4 为必胜状态。相应地，剩下 5 根木棍时，无论如何操作，都会让对手落入必胜状态，故 f_5 为必败状态。

类似伪代码地，可以写出：

$f_i = false$ if all $k \in [i - 3, i - 1], f_k = true$
otherwise $f_i = true$

本题中，易发现 $f_1, f_5, f_9 \dots f_{29}$ 都是必败状态，因为他们都只能拿走 1~3 根木棍，从而只能让对手进入必胜状态。所以，先预处理所有必败状态和必胜状态。对于电脑，如果当前是必败状态，则利用随机数随机进行操作，因为无论如何都会使用户进入必胜状态。如果用户不够聪明使电脑落入了必胜状态，则向前进行查询，使用户落入必败状态。由前推到可知，其实一旦使得电脑进入必胜状态，由于其永远按照最优解进行游戏，用户无论如何进行操作，都会获得失败的结果。

下面给出预处理和电脑进行操作的代码：

```
for(int i = 1; i ≤ 30; i++) // 预处理
    win[i] = true;
for(int i = 1; i ≤ 30; i += 4){
    win[i] = false;
}
```

```
// 电脑操作
if(!win[n]){ // 必败状态，随便输出一个即可
    int x = random.nextInt(3) + 1;
    System.out.printf("Computer takes %d\n", Math.min(n, x));
    n -= x;
}
else{
    int x = 0;
    for(int i = n; i ≥ 1; i--){
        if(!win[i]){ // 找到下一个必败状态
            x = n - i;
            break;
        }
    }
    System.out.printf("Computer takes %d\n", x);
    n -= x; // 总数减掉 x
}
```

调试过程与心得

本次调试过程难点主要在于对用户输入是否为整型的合法性判断。最开始，我想要对所有的输入都进行一次强制转换判断，但发现这样子代码量非常大，且其代码内容其实大部分都是重复的，故调整为将其封装为一个方法进行复用。但第二个问题是，在用户输入错误后，我们希望重新输出询问内容，故需要给输入函数传递一个 *String* 类，方便输出不同的问题。

通过这个调试内容，我对 *Java* 中的方法的作用有了非常深刻的认知。对于大量重复性的工作，我们完全可以将其封装为一个方法，方便代码复用，而不是简单粗暴的进行复制粘贴，然后通过手动修改为每个情况进行“定制”。这样操作将会大幅降低代码的可读性。

4. 设计一个交互式的程序，响应用户输入，创建一个多行的数字阵列

解题思路

用 m 存储总行数，然后用一个数组 n ， n_i 表示第 i 行的列数即可。

代码实现细节

对于每一行，使用 *Random* 类来生成随机数，并记录在二维数组中。为了方便查看和调试，此处限制了随机数大小在 1000 以内。

```
for(int i = 1; i ≤ m; i++){
    n[i] = Check_input("Input next row length: ");
    for(int j = 1; j ≤ n[i]; j++){
        a[i][j] = random.nextInt(1000);
        System.out.printf("%d ", a[i][j]);
    }
    System.out.println();
}
```

代码调试与心得

在调用 *Random* 类时，一开始并没有设置其 *bound* 属性，导致生成的数字非常大，且包含有负数，不太方便进行检查。后来经过查阅，得知设置 *bound* 属性后，可以使其生成 $[0, x)$ 之间的所有随机整数。

通过这道题目，让我得知了 *Java* 提供的已经封装好的类的功能强大与实用性，以后要多加学习其各种预封装好的类的使用。

5. 滚动阵列

解题思路

使用二维数组，对每种输入进行分类讨论，使用 *for* 循环依次进行赋值，注意同时也交换每一行的列数。

核心代码

通过方法，按照 4 的过程生成一个随机二维数组。

```
public static int[][] Get_array(int n, int m[], int maxm){
    int[][] a = new int[n + 1][maxm + 1];
```



```

    for(int i = 1; i ≤ n; i++){
        for(int j = 1; j ≤ m[i]; j++){
            a[i][j] = random.nextInt(100);
        }
    }
    return a;
}

```

对于左右滚动操作，直接进行依次赋值即可。需要注意处理 1 和 m_i 的边界处理。此处以 a 操作为例。 d 操作反过来即可。

```

if(ch == 'a'){
    for(int i = 1; i ≤ n; i++){
        int tmp = a[i][1];
        for(int j = 1; j < m[i]; j++){
            a[i][j] = a[i][j+1];
        }
        a[i][m[i]] = tmp;
    }
}

```

对于上下滚动操作，在进行依次赋值的同时，还要注意 m_i 列数的交换。可以知道其轮换规律与原矩阵轮换规律完全一致，故对其额外进行一轮操作即可。此处以 w 操作为例， s 操作类似，反过来即可。

```

if(ch == 'w'){
    for(int j = 1; j ≤ maxm; j++){
        int tmp = a[1][j];
        for(int i = 1; i < n; i++){
            a[i][j] = a[i+1][j];
        }
        a[n][j] = tmp;
    }
    int tmp1 = m[1];    // 对 m 进行单独轮换操作
    for(int i = 1; i < n; i++){
        m[i] = m[i+1];
    }
    m[n] = tmp1;
}

```

代码调试

在第一次编译运行时，我在 w 和 s 操作处遇到了问题。经过检查发现，有一些行的列数并不正确，导致额外输出了 0 (该行多余的默认数字)或者遗失少部分数字。后发现是因为将对 m 数组的

交换操作写在了对 *maxm* 循环的内部，即循环了 *maxm* 次。这导致了 *m* 数组的内容错位，最终展现出了输出错误。

心得

通过这次实验，我得知了代码细节的重要性。原先写在循环内部，只是为了贪图一时方便，可以少写一次 *for* 循环。但事实证明，有时候偷懒反而会带来适得其反的效果。因此，之后在其他场景下，一定要认真考虑各种偷懒和简写是否正确，不能随意“偷懒”。

实验总结

通过今天的这次实验，我对 *Java* 语言的运用变得更加熟练了。在许多方面，*Java* 与 *C++* 语言有着相似之处，但是仍在许多地方差异较大。例如，其对于数据类型赋值有着更加严格的要求，在 *int* 和 *boolean* 类型间不会自动转换。表面上这会导致一些在 *C++* 中的简易写法失效，但是也增加了代码的易读性和严谨性。

其次是对于 *Java* 提供的一些基础类的使用，变得更加熟练了，也更加理解其重要性。比如对于 *String* 类，使用方法与 *C++* 中有些许不同，需用使用提供的方法接口来进行修改等。

在之后的实验与日常训练中，我还需要进一步熟练 *Java* 的各项语法以及类的使用方法。

完整代码

1

```
import java.util.Scanner;

public class Main{
    public static void main(String[] args) {
        System.out.println("厦门大学102生日快乐!");
        for (int i = 1; i ≤ 25; i++) {
            System.out.print(" ");
        }
        System.out.println("xmu");
        for (int i = 1; i ≤ 5; i++) {
            System.out.println();
        }

        int del = 0; // 爱心的空格数量
        for(int i = 1; i ≤ 13; i++){
            for(int j = 1; j ≤ 5; j++)
                System.out.print(" ");
            for(int j = 1; j ≤ 3; j++)
```

```

        System.out.print("*");
    for(int j = 1; j ≤ 5; j++)
        System.out.print(" "); // 绘制 I//
System.out.println();
// 总共 27 列
char[] s = new char[14];
if(i ≤ 2){
    int now = 1;
    for(int j = 1; j ≤ 5 - 2 * i; j++){
        s[now++] = ' ';
    }
    for(int j = 1; j ≤ 5 * i; j++){
        s[now++] = '*';
    }
    for(int j = now; j ≤ 13; j++){
        s[now++] = ' ';
    }
}
else if(i ≤ 5){
    for(int j = 1; j ≤ 13; j++)
        s[j] = '*';
}
else{
    int now = 1;
    if(i ≤ 9)del++;
    else if(i ≤ 11 || i == 13) del += 2;
    else if(i == 12) del += 3;

    for(int j = 1; j ≤ del; j++){
        s[now++] = ' ';
    }
    while(now ≤ 13){
        s[now++] = '*';
    }
}
for(int j = 1; j ≤ 13; j++)
    System.out.print(s[j]);
if(i ≤ 3) System.out.print(" ");
else System.out.print("*");
for(int j = 13; j ≥ 1; j--)
    System.out.print(s[j]); // 对称

for(int j = 1; j ≤ 5; j++)
    System.out.print(" ");
for(int j = 1; j ≤ 15; j++){
    if(i ≤ 11){

```

```

        if(j ≤ 3 || j ≥ 13) System.out.print("*");
        else System.out.print(" ");
    }
    else System.out.print("*");
}

System.out.println();
}
}
}

```

2

方法 1

```

//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
public class Main {
    public static void main(String[] args) {
        int ans = 0;
        int[] G = new int[(int)1e5]; // 存储所有水仙花数
        int top = 0;
        for(int i = 100; i ≤ 999; i++){
            int tmp = i;
            int sum = 0;
            while(tmp > 0){
                int x = tmp % 10;
                sum += x * x * x;
                tmp /= 10;
            }
            if(sum == i){ // 是水仙花数
                ans++;
                G[++top] = i;
            }
        }
        System.out.println(ans);
        for(int i = 1; i ≤ top; i++){
            System.out.printf("%d ", G[i]);
        }
    }
}

```

方法 2

```
//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
public class Main {
    static int[] a = new int[4];
    static boolean[] vis = new boolean[11];
    static int ans = 0;
    static int[] G = new int[1000];
    public static void dfs(int now){
        if(now > 3){
            int x = a[1] * 100 + a[2] * 10 + a[3];
            if(x < 100) return ;
            int sum = 0;
            for(int i = 1; i ≤ 3; i++){
                sum += a[i] * a[i] * a[i];
            }
            if(sum == x){
                G[++ans] = sum;
            }
            return ;
        }
        for(int i = 0; i ≤ 9; i++){
            a[now] = i;
            dfs(now +1);
        }
        return ;
    }

    public static void main(String[] args) {
        dfs(1);
        System.out.println(ans);
        for(int i = 1; i ≤ ans; i++){
            System.out.printf("%d ", G[i]);
        }
        return ;
    }
}
```

3

```
import java.util.Random;
import java.util.Scanner;

//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
public class Main {

    static boolean[] win = new boolean[31];
```

```

static Scanner input = new Scanner(System.in);
public static int Check_input(String s){
    System.out.println(s);
    double x = input.nextDouble();
    while(x != (int)x){
        System.out.println("Please input an int, not a double!");
        System.out.println(s);
        x = input.nextDouble();
    }
    return (int)x;
}

public static void main(String[] args) {
    for(int i = 1; i ≤ 30; i++){
        win[i] = true;
    }
    for(int i = 1; i ≤ 30; i += 4){
        win[i] = false;
    }
    Random random = new Random();

    int n = Check_input("Please input the initial stix number: ");
    while(n < 5 || n > 30){
        System.out.println("The number must be between 5 and 30. Input
again!");
        n = Check_input("Please input the initial stix number: ");
    }

    int user = Check_input("Computer first, type 1. You first, type 2") -
1;

    while(user != 0 && user != 1){
        System.out.println("Error! Input again!");
        user = Check_input("Computer first, type 1. You first, type 2") -
1;
    }

    while(n > 0){
        for(int i = 1; i ≤ n; i++){
            System.out.print("|");
            System.out.println();
            if(user == 1){

                int x = Check_input("How many strix do you want to take: ");
                while(x < 1 || x > 3){
                    System.out.println("Must between 1 and 3. Input again!");
                    x = Check_input("How many strix do you want to take: ");
                }
            }
        }
    }
}

```

```

        while(n - x < 0){
            System.out.println("The input is bigger than the number of
the Stix! Input again!");
            x = Check_input("How many strix do you want to take: ");
        }
        n -= x;
    }
    else {

        if(!win[n]){ // 必败状态, 随便输出一个即可
            int x = random.nextInt(3) + 1;
            System.out.printf("Computer takes %d\n", Math.min(n, x));
            n -= x;
        }
        else{
            int x = 0;
            for(int i = n; i ≥ 1; i--){
                if(!win[i]){ // 找到下一个必败状态
                    x = n - i;
                    break;
                }
            }
            System.out.printf("Computer takes %d\n", x);
            n -= x;
        }
    }
    user ^= 1;
}
if(user == 1) System.out.println("You win!");
else System.out.println("You lose!");
return ;
}
}

```

4

```

import java.util.Scanner;
import java.util.Random;
//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
public class Main {
    static Scanner input = new Scanner(System.in);
    static Random random = new Random((int)1000);
    public static int Check_input(String s){

```

```

        System.out.println(s);
        double x = input.nextDouble();
        while(x != (int)x){
            System.out.println("Please input an int, not a double!");
            System.out.println(s);
            x = input.nextDouble();
        }
        return (int)x;
    }

    public static void main(String[] args) {
        int[][] a = new int[1001][1001];
        int[] n = new int[1001];
        System.out.println("All the input should be between 1 and 1000");
        int m = Check_input("Input the total number of input rows: ");
        for(int i = 1; i ≤ m; i++){
            n[i] = Check_input("Input next row length: ");
            for(int j = 1; j ≤ n[i]; j++){
                a[i][j] = random.nextInt(1000);
                System.out.printf("%d ", a[i][j]);
            }
            System.out.println();
        }
        System.out.println("Finish input.Here is the whole array of the numbers created.");
        System.out.println();
        for(int i = 1; i ≤ m; i++){
            for(int j = 1; j ≤ n[i]; j++){
                System.out.printf("%d ", a[i][j]);
            }
            System.out.println();
        }
        return ;
    }
}

```

5

```

import java.util.Random;
import java.util.Scanner;

```

//TIP To Run code, press <shortcut actionId="Run"/> or
 // click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.


```

public class Main {
    static Random random = new Random();
    static int maxm = 0;
    public static int[][] Get_array(int n, int m[], int maxm){
        int[][] a = new int[n + 1][maxm + 1];
        for(int i = 1; i ≤ n; i++){
            for(int j = 1; j ≤ m[i]; j++){
                a[i][j] = random.nextInt(100);
            }
        }
        return a;
    }
    static Scanner input = new Scanner(System.in);
    public static int Check_input(String s){
        System.out.println(s);
        double x = input.nextDouble();
        while(x ≠ (int)x){
            System.out.println("Please input an int, not a double!");
            System.out.println(s);
            x = input.nextDouble();
        }
        return (int)x;
    }
    public static void Print_array(int[][] a, int n, int m[]){
        for(int i = 1; i ≤ n; i++){
            for(int j = 1; j ≤ m[i]; j++){
                // if(a[i][j] == 0) System.out.print(" ");
                System.out.printf("%d ", a[i][j]);
            }
            System.out.println();
        }
        return ;
    }

    public static void main(String[] args) {
        int n = random.nextInt(3) + 3; // 不要只有一行
        int[] m = new int[n + 1];

        for(int i = 1; i ≤ n; i++)
            m[i] = random.nextInt(5);
        for(int i = 1; i ≤ n; i++){
            maxm = Math.max(maxm, m[i]);
        }
        int[][] a = Get_array(n, m, maxm);
        System.out.println("Finish. Here is the initial array.");
        Print_array(a, n, m);
    }
}

```

```

while(true){
    System.out.println("Enter a char(w, a, s, d) to scroll the array,
q to quit!");
    char ch = input.next().charAt(0);
    if(ch == 'q'){
        Print_array(a, n, m);
        break;
    }
    if(ch == 'a'){
        for(int i = 1; i ≤ n; i++){
            int tmp = a[i][1];
            for(int j = 1; j < m[i]; j++){
                a[i][j] = a[i][j+1];
                a[i][m[i]] = tmp;
            }
        }
    }
    else if(ch == 'd'){
        for(int i = 1; i ≤ n; i++){
            int tmp = a[i][m[i]];
            for(int j = m[i]; j ≥ 2; j--){
                a[i][j] = a[i][j-1];
                a[i][1] = tmp;
            }
        }
    }
    else if(ch == 'w'){
        for(int j = 1; j ≤ maxm; j++){
            int tmp = a[1][j];
            for(int i = 1; i < n; i++){
                a[i][j] = a[i+1][j];
            }
            a[n][j] = tmp;
        }
        int tmp1 = m[1];
        for(int i = 1; i < n; i++){
            m[i] = m[i+1];
        }
        m[n] = tmp1;
    }
    else if(ch == 's'){
        for(int j = 1; j ≤ maxm; j++){
            int tmp = a[n][j];
            for(int i = n; i ≥ 2; i--){
                a[i][j] = a[i-1][j];
            }
            a[1][j] = tmp;
        }
    }
}

```

```
        int tmp2 = m[n];
        for(int i = n; i ≥ 2; i--){
            m[i] = m[i-1];
        }
        m[1] = tmp2;
    }
    else{
        System.out.println("Error! Input again!");
        continue;
    }
    Print_array(a, n, m);
}
}
```