Chapter 8 Multidimensional Arrays

编程练习题 Exercise04　11.12 交

书7.21（游戏：豆机）豆机，也称为梅花瓶或高尔顿瓶，他是一个用来做统计实验的设备，是用英国科学家瑟弗兰克斯高尔顿的名字来命名的。它是一个三角形状的均匀放置钉子（或钩子）的直立板子，如图7-13所示。

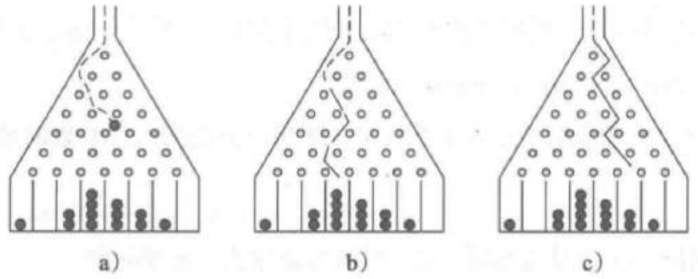
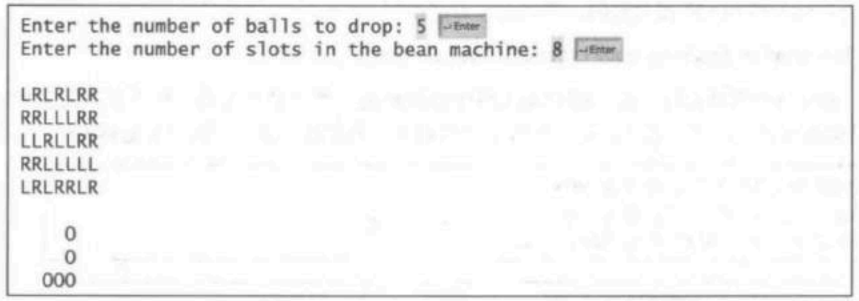


图 7-13　每个球都选取一个随机路径，然后掉入一个槽中

球都是从板子口落下的。每当球碰到钉子，它就有50%的机会落下左边或落向右边。在板子底部的槽子（slot）中都会积累一堆球。

编写程序模拟豆机。程序应该提示用户输入球的个数以及机器的槽数。打印每个球的路径模拟它的下落。例如，在图7-13b中求得路径是LLRRLLR，而在图7-13c中球的路径是RLRRLRR。使用条形图显示槽中秋的最终储备量。下面是程序的一个运行:



```
Enter the number of balls to drop: 5 ↵Enter
Enter the number of slots in the bean machine: 8 ↵Enter

LRLRLRR
RRLLLRR
LLRLLRR
RRLLLLL
LRLRRLR

    0
    0
   000
```

提示：创建一个名为slots的数组。数组slots中的每个元素存储的是一个槽中球的个数。每个球都经过一条路径落入一个槽中。路径上R的个数表示球落下的槽的位置。例如：对于路径LRLRLRR而言，球落到slots[4]中，而对路径RRLLLLL而言，球落到slots[2]中。

编程练习题 Exercise04　11.12 交

书7.29（游戏：选出四张牌）

编写一个程序，从一副52张的牌中选出四张，然后计算它们的和。Ace、King、Queen和Jack分别表示1、13、12和11。程序应该显示得到的和为24的选牌次数。

编程练习题 Exercise04 11.12 交

书7.31（合并两个有序列表）

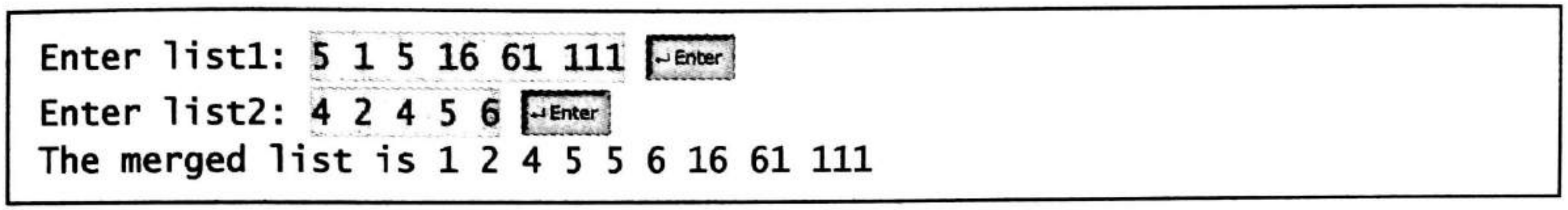
编写下面的方法，将两个有序列表合并成一个新的有序列表。

public static int[] merge(int[] list1, int list2)

只进行list1.length+ list2.length次比较来实现该方法。编写一个测试程序，提示用户输入两个有序列表，然后显示合并的列表。

下面是一个运行示例。

注意，输入中的第一个数表示列表中元素的个数。该数不是列表的一部分。



```
Enter list1: 5 1 5 16 61 111 ↵Enter
Enter list2: 4 2 4 5 6 ↵Enter
The merged list is 1 2 4 5 5 6 16 61 111
```

编程练习题 Exercise04 11.12 交

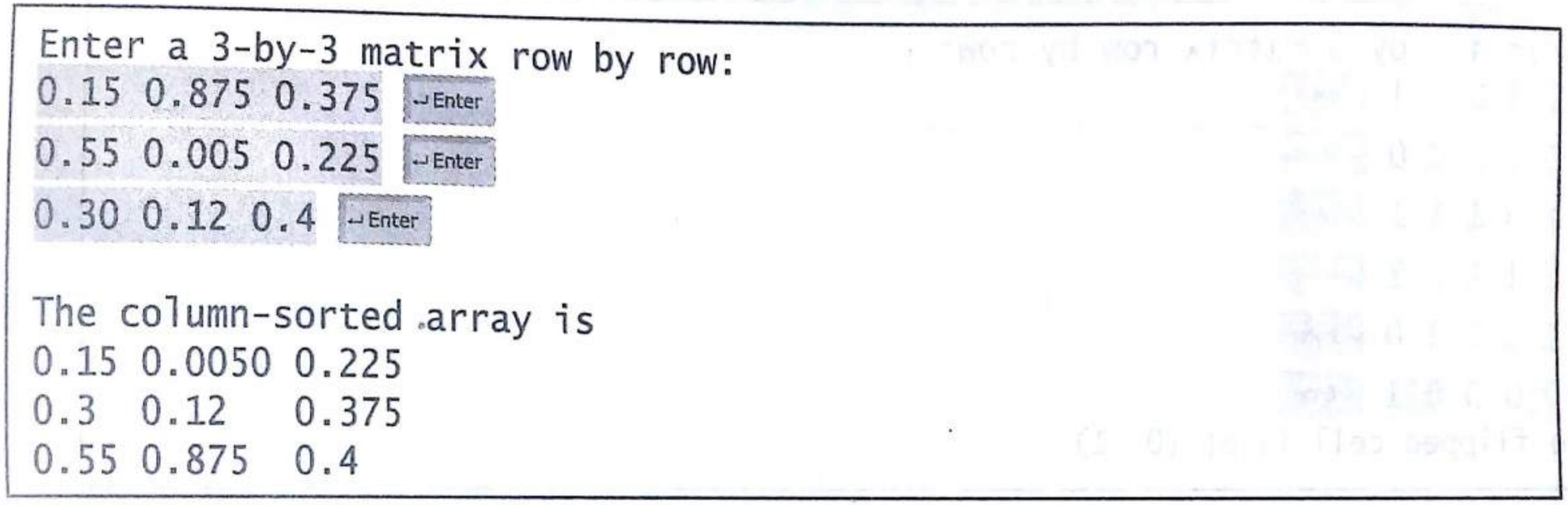
8.27 （列排序）

使用下面的方法实现二维数组的列排序。返回一个新的数组，并且原数组保持不变。

public static double[][] sortColumns(double[][] m)

编写一个测试程序，提示用户输入一个3×3的double值的矩阵，显示一个新的每列排好序的矩阵。

下面是一个运行示例。



```
Enter a 3-by-3 matrix row by row:
0.15 0.875 0.375  ↵Enter
0.55 0.005 0.225  ↵Enter
0.30 0.12 0.4  ↵Enter

The column-sorted array is
0.15 0.0050 0.225
0.3  0.12   0.375
0.55 0.875  0.4
```

Motivations

Thus far, you have used one-dimensional arrays to model linear collections of elements. You can use a two-dimensional array to represent a matrix or a table. For example, the following table that describes the distances between the cities can be represented using a two-dimensional array.

Distance Table (in miles)

	Chicago	Boston	New York	Atlanta	Miami	Dallas	Houston
Chicago	0	983	787	714	1375	967	1087
Boston	983	0	214	1102	1763	1723	1842
New York	787	214	0	888	1549	1548	1627
Atlanta	714	1102	888	0	661	781	810
Miami	1375	1763	1549	661	0	1426	1187
Dallas	967	1723	1548	781	1426	0	239
Houston	1087	1842	1627	810	1187	239	0

Motivations

```
double[][] distances = {
  {0, 983, 787, 714, 1375, 967, 1087},
  {983, 0, 214, 1102, 1763, 1723, 1842},
  {787, 214, 0, 888, 1549, 1548, 1627},
  {714, 1102, 888, 0, 661, 781, 810},
  {1375, 1763, 1549, 661, 0, 1426, 1187},
  {967, 1723, 1548, 781, 1426, 0, 239},
  {1087, 1842, 1627, 810, 1187, 239, 0},
};
```

Declare/Create Two-dimensional Arrays

```
// Declare array ref var
dataType[][] refVar;

// Create array and assign its reference to variable
refVar = new dataType[10][10];

// Combine declaration and creation in one statement
dataType[][] refVar = new dataType[10][10];

// Alternative syntax  (not recommended)
dataType refVar[][] = new dataType[10][10];
```

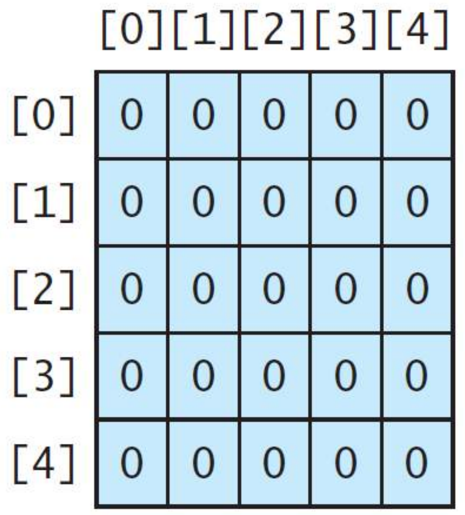
Declaring Variables of Two -dimensional Arrays and Creating Two-dimensional Arrays

```
int[][] matrix = new int[10][10];
```

or

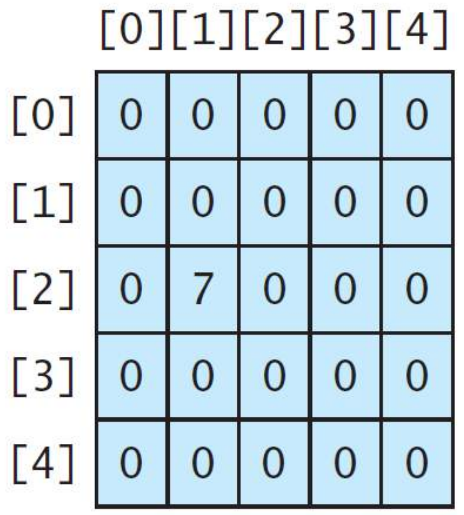
```
int matrix[][] = new int[10][10];
matrix[0][0] = 3;

for (int i = 0; i < matrix.length; i++)
  for (int j = 0; j < matrix[i].length; j++)
    matrix[i][j] = (int)(Math.random() * 1000);

double[][] x;
```

Two-dimensional Array Illustration



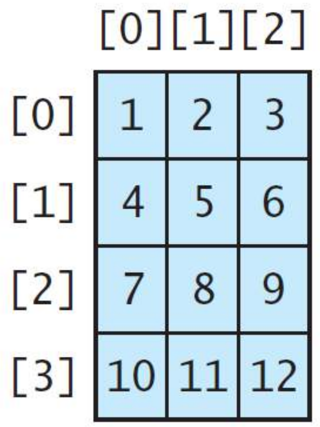
```
matrix = new int[5][5];
```

(a)



```
matrix[2][1] = 7;
```

(b)



```
int[][] array = {
  {1, 2, 3},
  {4, 5, 6},
  {7, 8, 9},
  {10, 11, 12}
};
```

(c)

matrix.length? 5

matrix[0].length? 5

array.length? 4

array[0].length? 3

Declaring, Creating, and Initializing Using Shorthand Notations

You can also use an array initializer to declare, create and initialize a two-dimensional array. For example,

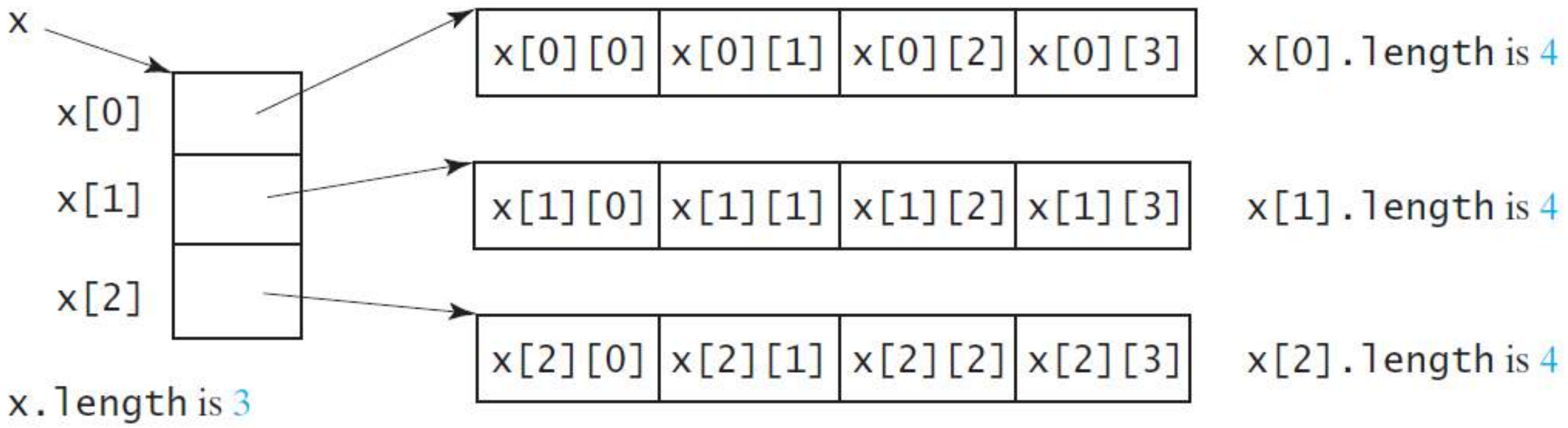
```
int[][] array = {
  {1, 2, 3},
  {4, 5, 6},
  {7, 8, 9},
  {10, 11, 12}
};
```

Same as

```
int[][] array = new int[4][3];
array[0][0] = 1; array[0][1] = 2; array[0][2] = 3;
array[1][0] = 4; array[1][1] = 5; array[1][2] = 6;
array[2][0] = 7; array[2][1] = 8; array[2][2] = 9;
array[3][0] = 10; array[3][1] = 11; array[3][2] = 12;
```

Lengths of Two-dimensional Arrays

int[][] x = new int[3][4];



Lengths of Two-dimensional Arrays, cont.

```
int[][] array = {
  {1, 2, 3},
  {4, 5, 6},
  {7, 8, 9},
  {10, 11, 12}
};
```

```
array.length
array[0].length
array[1].length
array[2].length
array[3].length
```

array[4].length   ArrayIndexOutOfBoundsException

Ragged Arrays

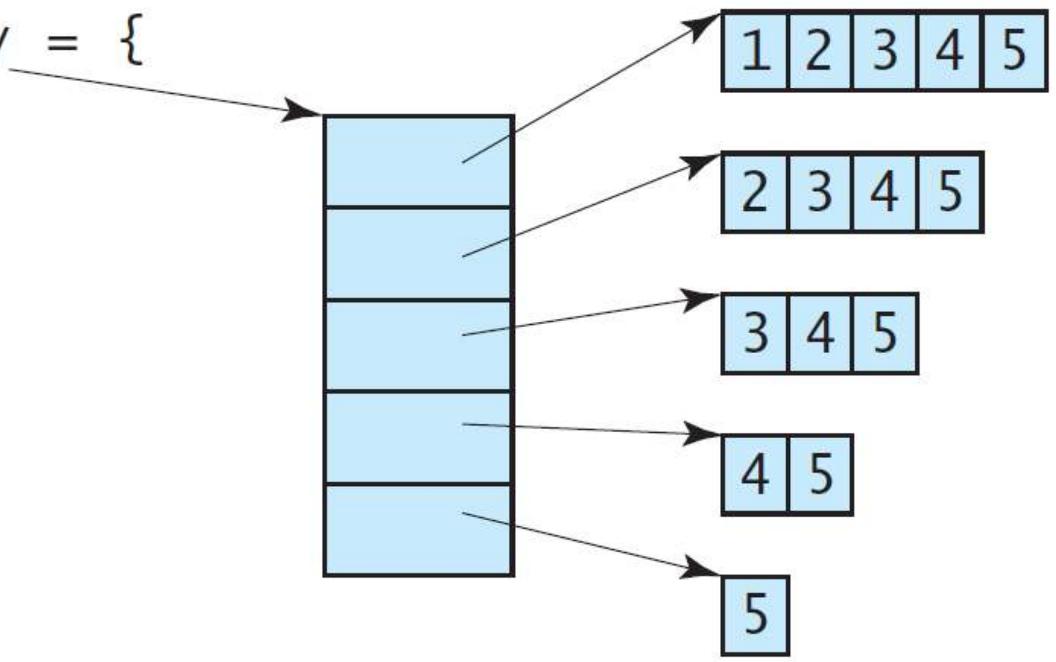
Each row in a two-dimensional array is itself an array. So, the rows can have different lengths. Such an array is known as *a ragged array*. For example,

```
int[][] matrix = {
  {1, 2, 3, 4, 5},
  {2, 3, 4, 5},
  {3, 4, 5},
  {4, 5},
  {5}
};
```

```
matrix.length is 5
matrix[0].length is 5
matrix[1].length is 4
matrix[2].length is 3
matrix[3].length is 2
matrix[4].length is 1
```

Ragged Arrays, cont.

```
int[][] triangleArray = {
  {1, 2, 3, 4, 5},
  {2, 3, 4, 5},
  {3, 4, 5},
  {4, 5},
  {5}
};
```



Processing Two-Dimensional Arrays

See the examples in the text.

1. (Initializing arrays with input values)
2. (Printing arrays)
3. (Summing all elements)
4. (Summing all elements by column)
5. (Which row has the largest sum)
6. (Finding the smallest index of the largest element)
7. (*Random shuffling*)

Initializing arrays with input values

```
java.util.Scanner input = new Scanner(System.in);
System.out.println("Enter " + matrix.length + " rows and " +
  matrix[0].length + " columns: ");
for (int row = 0; row < matrix.length; row++) {
  for (int column = 0; column < matrix[row].length; column++) {
    matrix[row][column] = input.nextInt();
  }
}
```

Initializing arrays with random values

```
for (int row = 0; row < matrix.length; row++) {
  for (int column = 0; column < matrix[row].length; column++) {
    matrix[row][column] = (int)(Math.random() * 100);
  }
}
```

Printing arrays

```
for (int row = 0; row < matrix.length; row++) {
  for (int column = 0; column < matrix[row].length; column++) {
    System.out.print(matrix[row][column] + " ");
  }

  System.out.println();
}
```

Summing all elements

```
int total = 0;
for (int row = 0; row < matrix.length; row++) {
  for (int column = 0; column < matrix[row].length; column++) {
    total += matrix[row][column];
  }
}
```

Summing elements by column

```
for (int column = 0; column < matrix[0].length; column++) {
  int total = 0;
  for (int row = 0; row < matrix.length; row++)
    total += matrix[row][column];
  System.out.println("Sum for column " + column + " is "
    + total);
}
```

Random shuffling

```
for (int i = 0; i < matrix.length; i++) {
  for (int j = 0; j < matrix[i].length; j++) {
    int i1 = (int)(Math.random() * matrix.length);
    int j1 = (int)(Math.random() * matrix[i].length);
    // Swap matrix[i][j] with matrix[i1][j1]
    int temp = matrix[i][j];
    matrix[i][j] = matrix[i1][j1];
    matrix[i1][j1] = temp;
  }
}
```

Passing Tow-Dimensional Arrays to Methods

PassTwoDimensionalArray

Problem: Grading Multiple -Choice Test

Students' answer

	0	1	2	3	4	5	6	7	8	9
Student 0	A	B	A	C	C	D	E	E	A	D
Student 1	D	B	A	B	C	A	E	E	A	D
Student 2	E	D	D	A	C	B	E	E	A	D
Student 3	C	B	A	E	D	C	E	E	A	D
Student 4	A	B	D	C	C	D	E	E	A	D
Student 5	B	B	E	C	C	D	E	E	A	D
Student 6	B	B	A	C	C	D	E	E	A	D
Student 7	E	B	E	C	C	D	E	E	A	D

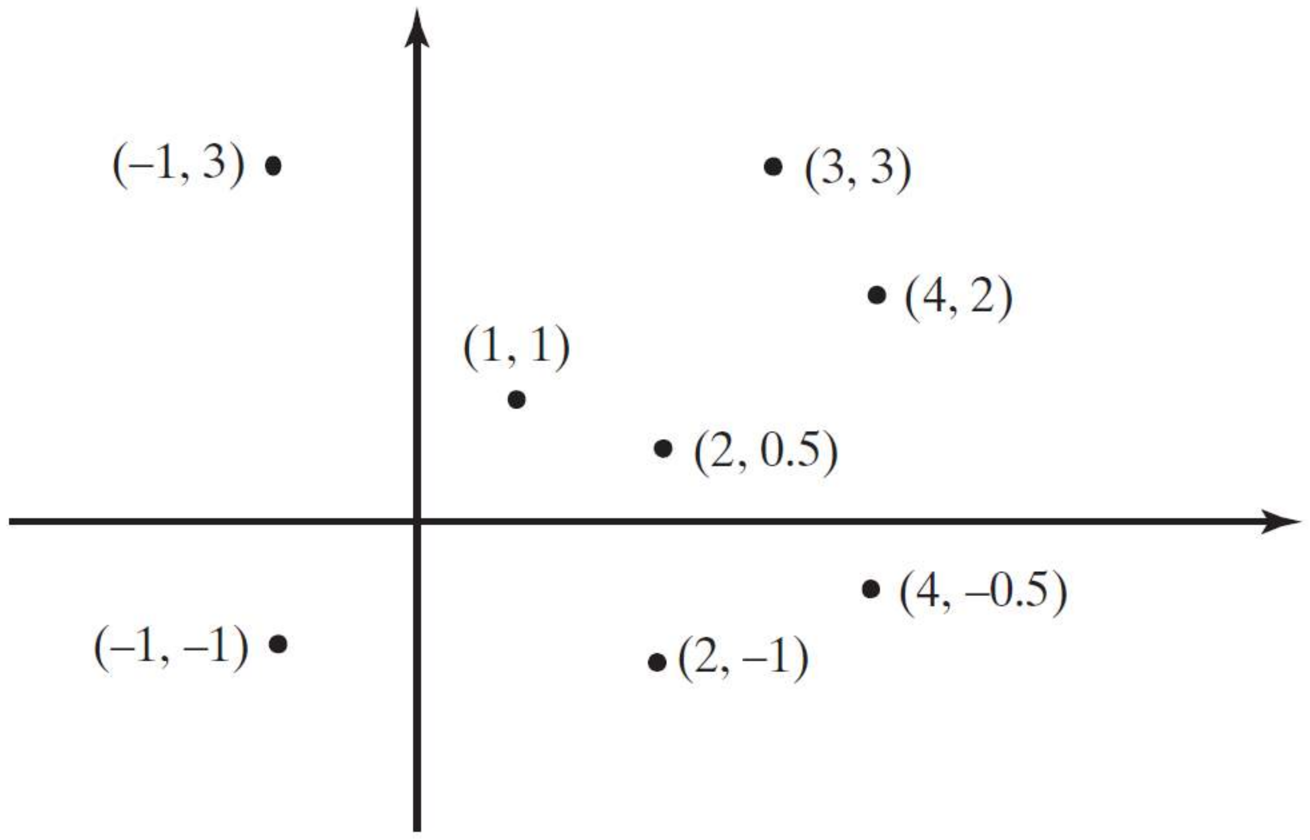
Objective: write a program that grades multiple-choice test.

Key to the Questions:

	0	1	2	3	4	5	6	7	8	9
Key	D	B	D	C	C	D	A	E	A	D

GradeExam

Problem: Finding Two Points Nearest to Each Other



	x	y
0	–1	3
1	–1	–1
2	1	1
3	2	0.5
4	2	–1
5	3	3
6	4	2
7	4	–0.5

FindNearestPoints

What is Sudoku?

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9

Every row contains the numbers 1 to 9

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9

5	3	*4*	*6*	7	*8*	*9*	*1*	*2*
6	*7*	*2*	1	9	5	*3*	*4*	*8*
1	9	8	*3*	*4*	*2*	*5*	6	*7*
8	*5*	*9*	*7*	6	*1*	*4*	*2*	3
4	*2*	*6*	8	*5*	3	*7*	*9*	1
7	*1*	*3*	*9*	2	*4*	*8*	*5*	6
9	6	*1*	*5*	*3*	*7*	*2*	*8*	*4*
2	*8*	*7*	4	1	9	*6*	*3*	5
3	*4*	*5*	*2*	8	*6*	*1*	7	9

Every column contains the numbers 1 to 9

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9

5	3	*4*	*6*	7	*8*	*9*	*1*	*2*
6	*7*	*2*	1	9	5	*3*	*4*	*8*
1	9	8	*3*	*4*	*2*	*5*	6	*7*
8	*5*	*9*	*7*	6	*1*	*4*	*2*	3
4	*2*	*6*	8	*5*	3	*7*	*9*	1
7	*1*	*3*	*9*	2	*4*	*8*	*5*	6
9	6	*1*	*5*	*3*	*7*	*2*	*8*	*4*
2	*8*	*7*	4	1	9	*6*	*3*	5
3	*4*	*5*	*2*	8	*6*	*1*	7	9

Every 3×3 box contains the numbers 1 to 9

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9

5	3	*4*	*6*	7	*8*	*9*	*1*	*2*
6	*7*	*2*	1	9	5	*3*	*4*	*8*
1	9	8	*3*	*4*	*2*	*5*	6	*7*
8	*5*	*9*	*7*	6	*1*	*4*	*2*	3
4	*2*	*6*	8	*5*	3	*7*	*9*	1
7	*1*	*3*	*9*	2	*4*	*8*	*5*	6
9	6	*1*	*5*	*3*	*7*	*2*	*8*	*4*
2	*8*	*7*	4	1	9	*6*	*3*	5
3	*4*	*5*	*2*	8	*6*	*1*	7	9

Checking Whether a Solution Is Correct

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9

5	3	*4*	*6*	7	*8*	*9*	*1*	*2*
6	*7*	*2*	1	9	5	*3*	*4*	*8*
1	9	8	*3*	*4*	*2*	*5*	6	*7*
8	*5*	*9*	*7*	6	*1*	*4*	*2*	3
4	*2*	*6*	8	*5*	3	*7*	*9*	1
7	*1*	*3*	*9*	2	*4*	*8*	*5*	6
9	6	*1*	*5*	*3*	*7*	*2*	*8*	*4*
2	*8*	*7*	4	1	9	*6*	*3*	5
3	*4*	*5*	*2*	8	*6*	*1*	7	9

CheckSudokuSolution

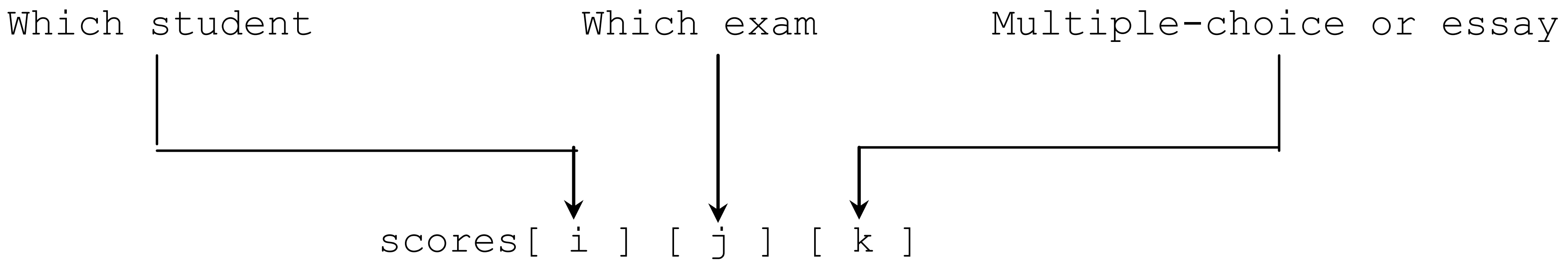
Multidimensional Arrays

Occasionally, you will need to represent n -dimensional data structures. In Java, you can create n-dimensional arrays for any integer n.

The way to declare two-dimensional array variables and create two-dimensional arrays can be generalized to declare n-dimensional array variables and create n-dimensional arrays for n >= 3.

Multidimensional Arrays

```
double[][][] scores = {
  {{7.5, 20.5}, {9.0, 22.5}, {15, 33.5}, {13, 21.5}, {15, 2.5}},
  {{4.5, 21.5}, {9.0, 22.5}, {15, 34.5}, {12, 20.5}, {14, 9.5}},
  {{6.5, 30.5}, {9.4, 10.5}, {11, 33.5}, {11, 23.5}, {10, 2.5}},
  {{6.5, 23.5}, {9.4, 32.5}, {13, 34.5}, {11, 20.5}, {16, 7.5}},
  {{8.5, 26.5}, {9.4, 52.5}, {13, 36.5}, {13, 24.5}, {16, 2.5}},
  {{9.5, 20.5}, {9.4, 42.5}, {13, 31.5}, {12, 20.5}, {16, 6.5}}};
```



Problem: Calculating Total Scores

Objective: write a program that calculates the total score for students in a class. Suppose the scores are stored in a three-dimensional array named scores. The first index in scores refers to a student, the second refers to an exam, and the third refers to the part of the exam. Suppose there are 7 students, 5 exams, and each exam has two parts--the multiple-choice part and the programming part. So, scores[i][j][0] represents the score on the multiple-choice part for the i's student on the j's exam. Your program displays the total score for each student.

TotalScore

Problem: Weather Information

Suppose a meteorology station records the temperature and humidity at each hour of every day and stores the data for the past ten days in a text file named weather.txt. Each line of the file consists of four numbers that indicate the day, hour, temperature, and humidity. Your task is to write a program that calculates the average daily temperature and humidity for the 10 days.

```
1 1 76.4 0.92
1 2 77.7 0.93
...
10 23 97.7 0.71
10 24 98.7 0.74
```

(a)

```
10 24 98.7 0.74
1 2 77.7 0.93
...
10 23 97.7 0.71
1 1 76.4 0.92
```

(b)

Weather

Problem: Guessing Birthday

Listing 4.3, GuessBirthday.java, gives a program that guesses a birthday. The program can be simplified by storing the numbers in five sets in a three-dimensional array, and it prompts the user for the answers using a loop.

GuessBirthdayUsingArray

编程练习题 Exercise04 11.12 交

书7.21（游戏：豆机）豆机，也称为梅花瓶或高尔顿瓶，他是一个用来做统计实验的设备，是用英国科学家瑟弗兰克斯高尔顿的名字来命名的。它是一个三角形状的均匀放置钉子（或钩子）的直立板子，如图7-13所示。

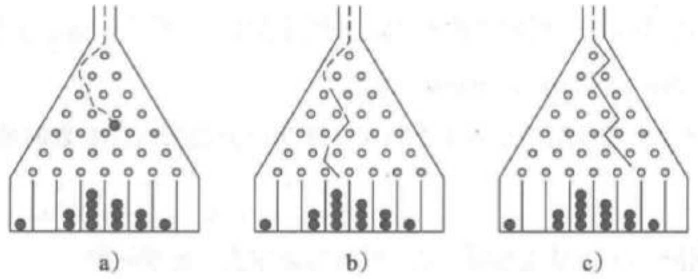
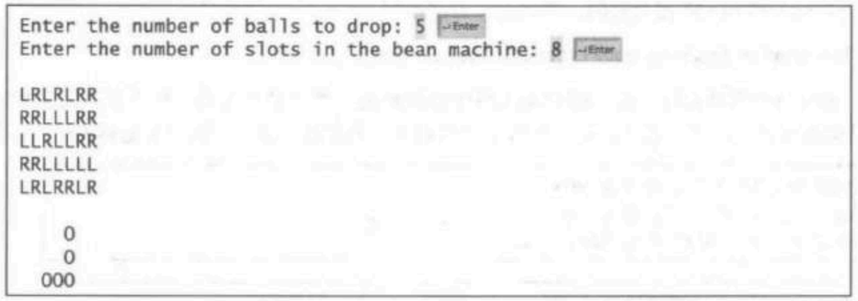


图 7-13　每个球都选取一个随机路径，然后掉入一个槽中

球都是从板子口落下的。每当球碰到钉子，它就有50%的机会落下左边或落向右边。在板子底部的槽子（slot）中都会积累一堆球。

编写程序模拟豆机。程序应该提示用户输入球的个数以及机器的槽数。打印每个球的路径模拟它的下落。例如，在图7-13b中求得路径是LLRRLLR，而在图7-13c中球的路径是RLRRLRR。使用条形图显示槽中秋的最终储备量。下面是程序的一个运行:



```
Enter the number of balls to drop: 5 ↵Enter
Enter the number of slots in the bean machine: 8 ↵Enter

LRLRLRR
RRLLLRR
LLRLLRR
RRLLLLL
LRLRRLR

    0
    0
   000
```

提示：创建一个名为 slots 的数组。数组 slots 中的每个元素存储的是一个槽中球的个数。每个球都经过一条路径落入一个槽中。路径上 R 的个数表示球落下的槽的位置。例如：对于路径 LRLRLRR 而言，球落到 slots[4] 中，而对路径 RRLLLLL 而言，球落到 slots[2] 中。

编程练习题 Exercise04　11.12 交

书7.29（游戏：选出四张牌）

编写一个程序，从一副52张的牌中选出四张，然后计算它们的和。Ace、King、Queen和Jack分别表示1、13、12和11。程序应该显示得到的和为24的选牌次数。

编程练习题 Exercise04 11.12 交

书7.31（合并两个有序列表）

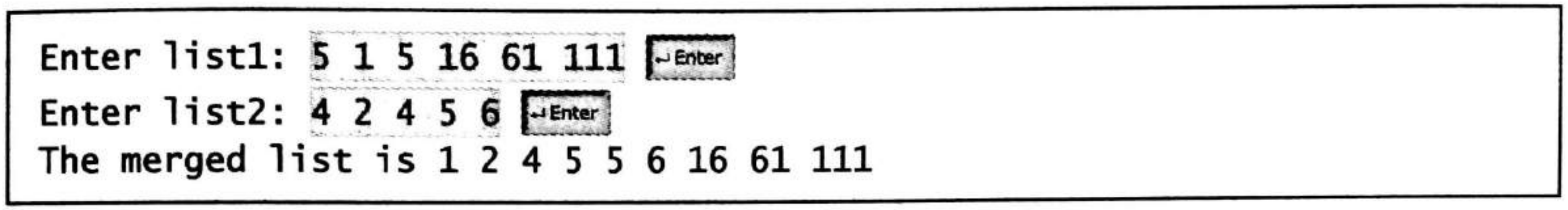
编写下面的方法，将两个有序列表合并成一个新的有序列表。

public static int[] merge(int[] list1, int list2)

只进行list1.length+ list2.length次比较来实现该方法。编写一个测试程序，提示用户输入两个有序列表，然后显示合并的列表。

下面是一个运行示例。

注意，输入中的第一个数表示列表中元素的个数。该数不是列表的一部分。



```
Enter list1: 5 1 5 16 61 111 ↵Enter
Enter list2: 4 2 4 5 6 ↵Enter
The merged list is 1 2 4 5 5 6 16 61 111
```

编程练习题 Exercise04　11.12 交

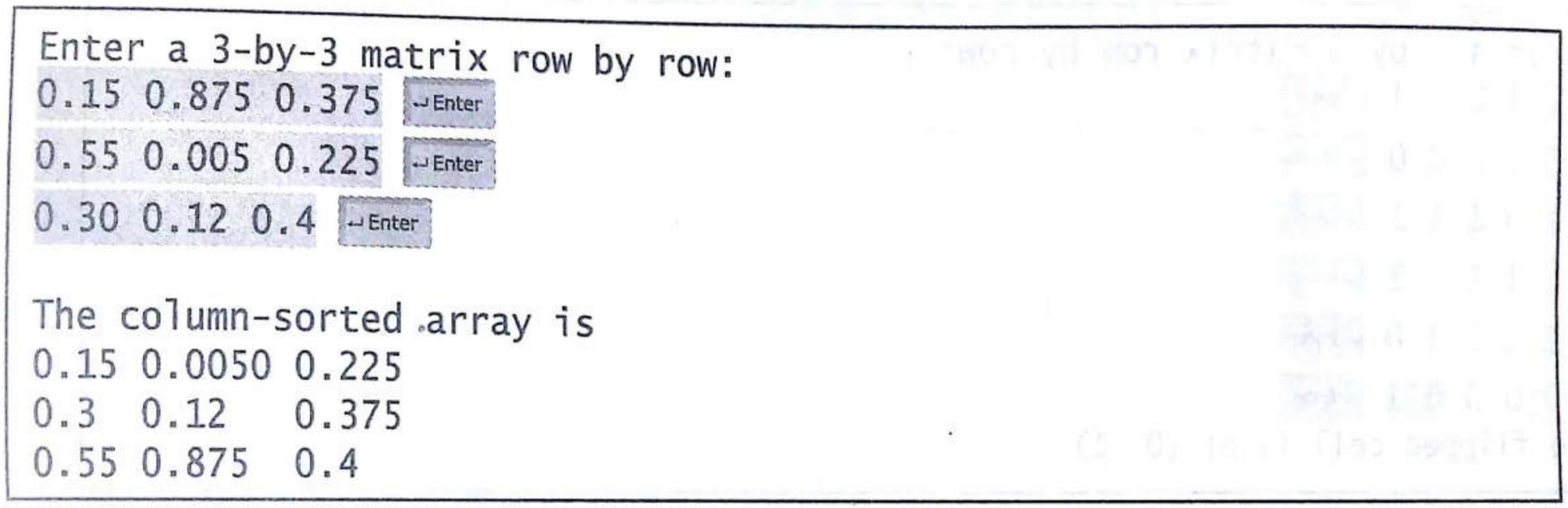
8.27（列排序）

使用下面的方法实现二维数组的列排序。返回一个新的数组，并且原数组保持不变。

public static double[][] sortColumns(double[][] m)

编写一个测试程序，提示用户输入一个3×3的double值的矩阵，显示一个新的每列排好序的矩阵。

下面是一个运行示例。



```
Enter a 3-by-3 matrix row by row:
0.15 0.875 0.375 ↵Enter
0.55 0.005 0.225 ↵Enter
0.30 0.12 0.4 ↵Enter

The column-sorted array is
0.15 0.0050 0.225
0.3  0.12   0.375
0.55 0.875  0.4
```