Chapter 8 Multidimensional Arrays

Objectives

To give examples of representing data using two-dimensional arrays

To declare variables for **two-dimensional arrays**, create arrays, and access array elements in a two-dimensional array using row and column indexes

To program common operations for two-dimensional arrays (displaying arrays, summing all elements, finding min and max elements, and random shuffling)

To pass two-dimensional arrays to methods

To write a program for grading multiple-choice questions using two-dimensional arrays

To solve the closest-pair problem using two-dimensional arrays

To check a Sudoku solution using two-dimensional arrays

To use **multidimensional arrays**

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
dataType refVar[][] = new dataType[10][10];
```

Example

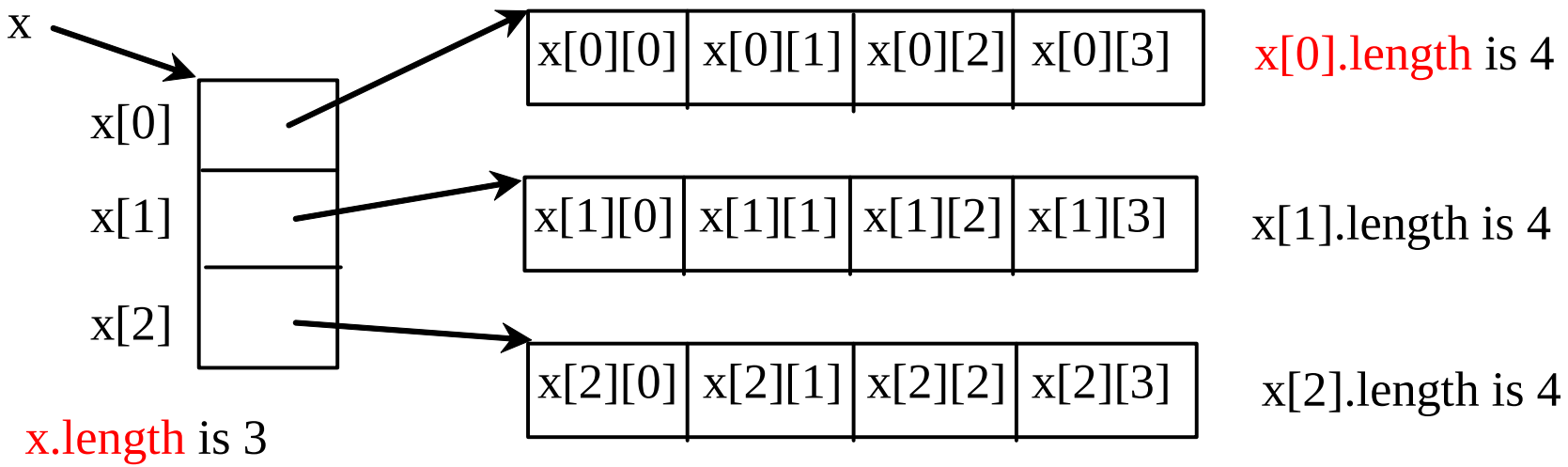
```
int[][] matrix = new int[10][10];
```

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;
```

Lengths of Two-dimensional Arrays

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



Obtain number of rows : using **x.length**

Obtain number of columns in a specified row : using **x[row].length**

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

array[0].length

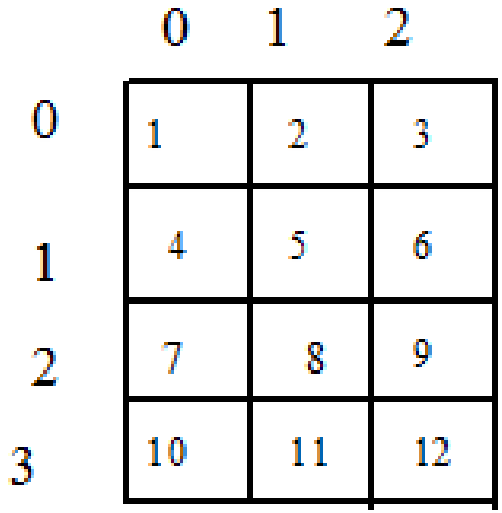
array[1].length

array[2].length

array[3].length

array[4].length

ArrayIndexOutOfBoundsException




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

array.length?

array[0].length?

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9
3	10	11	12

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

array.length? 4

array[0].length? 3

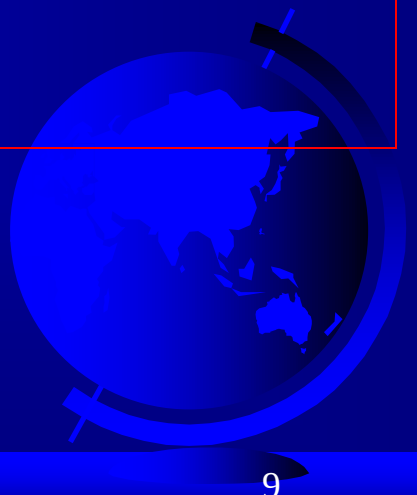
Declaring, Creating, and Initializing Using Shorthand Notations (array initializer)

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

Same as

```
int[][] array = new int[4][3];
```

array[0]= **{1, 2, 3};** **Wrong!**



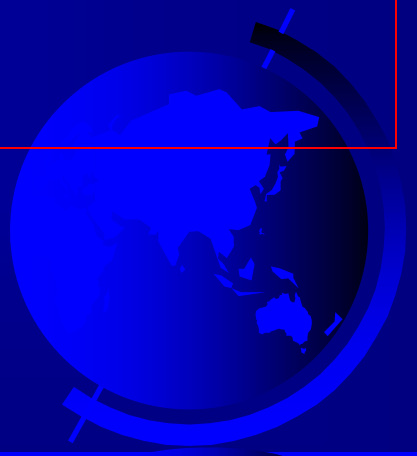
Declaring, Creating, and Initializing Using Shorthand Notations (array initializer)

```
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[0]= **{1, 2, 3};** **Wrong!**



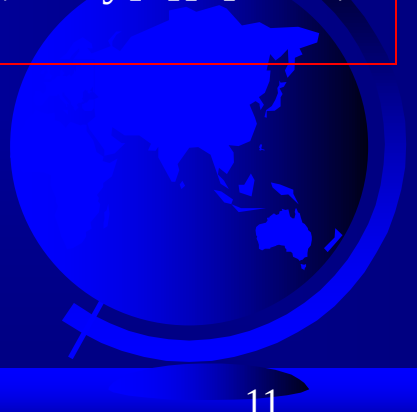
Declaring, Creating, and Initializing Using Shorthand Notations (array initializer)

```
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;
```

array[0]= **{1, 2, 3};** **Wrong!**

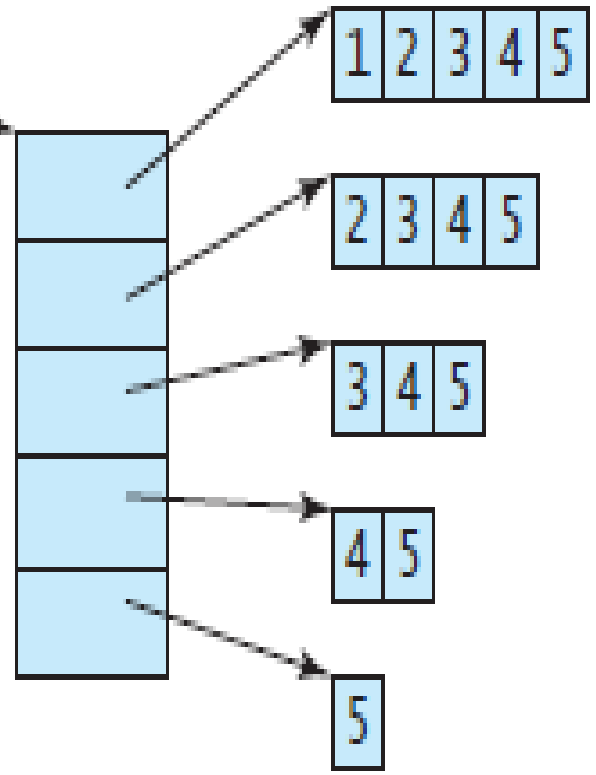


Ragged Arrays

Each row in a two-dimensional array is itself an array.

ragged array : **the rows have different lengths.**

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



matrix.length is 5

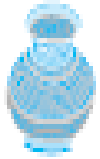
matrix[0].length is 5
matrix[1].length is 4
matrix[2].length is 3
matrix[3].length is 2
matrix[4].length is 1

If you don't know the values in a ragged array in advance, but know the sizes, say the same as before, you can create a ragged array using the syntax that follows:

```
int[][] triangleArray = new int[5][];
triangleArray[0] = new int[5];
triangleArray[1] = new int[4];
triangleArray[2] = new int[3];
triangleArray[3] = new int[2];
triangleArray[4] = new int[1];
```

You can now assign values to the array. For example,

```
triangleArray[0][3] = 50;
triangleArray[4][0] = 45;
```



Note

The syntax `new int[5][]` for creating an array requires the first index to be specified. The syntax `new int[][]` would be wrong.

Processing Two-Dimensional Arrays

7 examples

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*)

1. 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();
  }
}
```

3. 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();
}
```

5. 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);
}
```

7. 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.java

```
1  import java.util.Scanner;
2
3  public class PassTwoDimensionalArray {
4    public static void main(String[] args) {
5      // Create a Scanner
6      Scanner input = new Scanner(System.in);
7
8      // Enter array values
9      int[][] m = new int[3][4];
10     System.out.println("Enter " + m.length + " rows and "
11       + m[0].length + " columns: ");
12     for (int i = 0; i < m.length; i++)
13       for (int j = 0; j < m[i].length; j++)
14         m[i][j] = input.nextInt();
15
16     // Display result
17     System.out.println("\nSum of all elements is " + sum(m));
18   }
19
20   public static int sum(int[][] m) {
21     int total = 0;
22     for (int row = 0; row < m.length; row++) {
23       for (int column = 0; column < m[row].length; column++) {
24         total += m[row][column];
25       }
26     }
27
28     return total;
29   }
30 }
```

```
12      for (int i = 0; i < m.length; i++)
13        for (int j = 0; j < m[i].length; j++)
14          m[i][j] = input.nextInt();
15
16      // Display result
17      System.out.println("\nSum of all elements is " + sum(m) );
18    }
19
20    public static int sum(int[][] m) {
21      int total = 0;
22      for (int row = 0; row < m.length; row++) {
23        for (int column = 0; column < m[row].length; column++) {
24          total += m[row][column];
25        }
26      }
27
28      return total;
29    }
```

//returns the sum of all elements in a matrix.

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

number-placement puzzle

A grid divided into smaller boxes.

fixed cells: populated with **1** to **9**.

The objective is to fill *free cells* with **1** to **9,** so that every row, every column, and every box contains the numbers **1** to **9**

Write a **program** to **verify** whether a **solution** is correct

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

Program: Checking Whether a Solution Is Correct

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

```
A solution grid is
  {{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}
  };
```

```
Enter a Sudoku puzzle solution:
9 6 3 1 7 4 2 5 8  Enter
1 7 8 3 2 5 6 4 9  Enter
2 5 4 6 8 9 7 3 1  Enter
8 2 1 4 3 7 5 9 6  Enter
4 9 6 8 5 2 3 1 7  Enter
7 3 5 9 6 1 8 2 4  Enter
5 8 9 7 1 3 4 6 2  Enter
3 1 7 2 4 6 9 8 5  Enter
6 4 2 5 9 8 1 7 3  Enter
Valid solution
```

CheckSudokuSolution.java

```
 1 import java.util.Scanner;
 2
 3 public class CheckSudokuSolution {
 4   public static void main(String[] args) {
 5     // Read a Sudoku solution
 6     int[][] grid = readASolution();
 7
 8     System.out.println(isValid(grid) ? "Valid solution" :
 9       "Invalid solution");
10   }
11
12   /** Read a Sudoku solution from the console */
13   public static int[][] readASolution() {
14     // Create a Scanner
15     Scanner input = new Scanner(System.in);
16
17     System.out.println("Enter a Sudoku puzzle solution:");
18     int[][] grid = new int[9][9];
19     for (int i = 0; i < 9; i++)
20       for (int j = 0; j < 9; j++)
21         grid[i][j] = input.nextInt();
22
23     return grid;
24   }
```

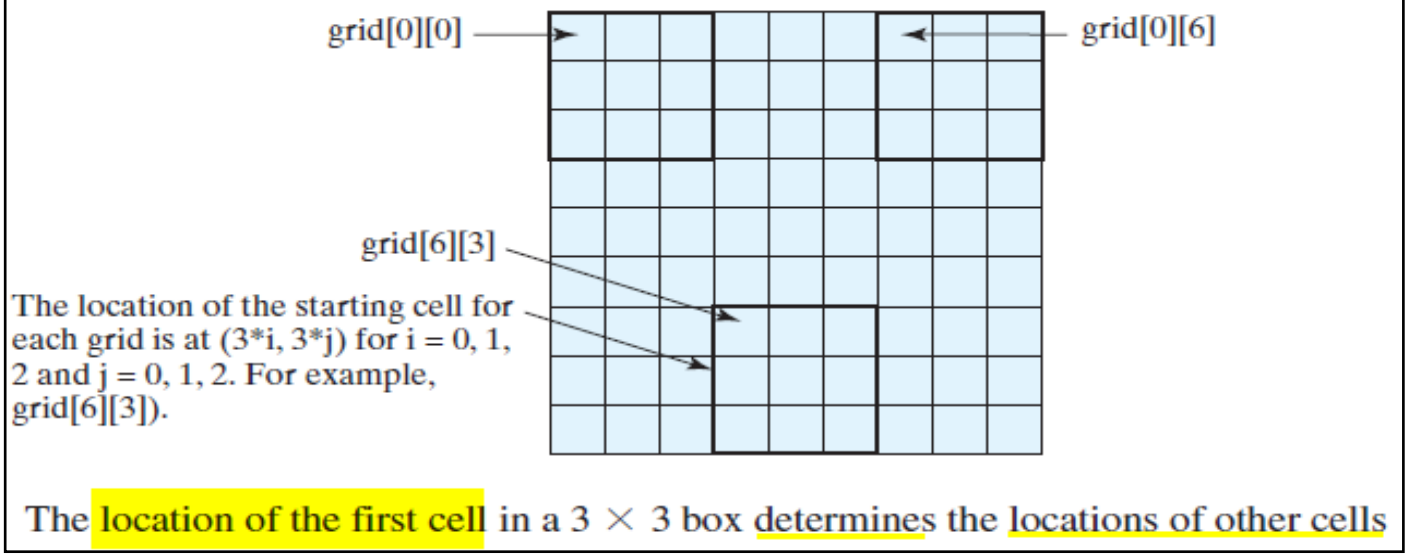
```
63   /** Check whether the one-dimensional array contains 1 to 9 */
64   public static boolean is1To9(int[] list) {
65     // Make a copy of the array
66     int[] temp = new int[list.length];
67     System.arraycopy(list, 0, temp, 0, list.length);
68
69     // Sort the array
70     java.util.Arrays.sort(temp);
71
72     // Check whether the list contains 1, 2, 3, ..., 9
73     for (int i = 0; i < 9; i++)
74       if (temp[i] != i + 1)
75         return false;
76
77     return true; // The list contains exactly 1 to 9
78   }
79 }
```

check solution

check rows

check columns

They are at (3i, 3j) for i = 0, 1, 2 and j = 0, 1, 2, as illustrated in



The location of the first cell in a 3 × 3 box determines the locations of other cells

check small boxes

```
45      // Check whether each 3-by-3 box has numbers 1 to 9
46      for (int i = 0; i < 3; i++) {
47        for (int j = 0; j < 3; j++) {
48          // The starting element in a small 3-by-3 box
49          int k = 0;
50          int[] list = new int[9]; // Get all numbers in the box to list
51          for (int row = i * 3; row < i * 3 + 3; row ++)
52            for (int column = j * 3; column < j * 3 + 3; column++)
53              list[k++] = grid[row][column];
54
55          if (!is1To9(list)) // If list does not contain 1 to 9
56            return false;
57        }
58      }
59
60      return true; // The fixed cells are valid
61    }
```

all valid

Multidimensional Arrays

Occasionally, you will need to represent n-dimensional data structures. In Java, you can create n-dimensional arrays (n >= 3, n is an integer).

For example,

```
double[][][] data = new double[10][5][2];
```

Multidimensional Arrays

A **multidimensional array** is actually an array in which each **element is another array.**

A three-dimensional array consists of an array of two-dimensional arrays, each of which is an array of one-dimensional arrays.

x = new int[2][2][5];

x.length is 2

x[0] and **x[1]** are two-dimensional arrays.

- x[0].length and x[1].length are 2
- **X[0][0], x[0][1], x[1][0],** and **x[1][1]** are one-dimensional arrays and each contains five elements

  X[0][0].length, x[0][1].length, x[1][0].length, and x[1][1].length are 5.

Problem: Guessing Birthdays

program that guesses a birthday.

simplifies it by storing the numbers in five sets in a three-dimensional array, and prompts the user for the answers using a loop.

```
1  import java.util.Scanner;
2
3  public class GuessBirthdayUsingArray {
4    public static void main(String[] args) {
5      int day = 0; // Day to be determined
6      int answer;
7
8      int[][][] dates = {
9        {{ 1,  3,  5,  7},
10        { 9, 11, 13, 15},
11        {17, 19, 21, 23},
12        {25, 27, 29, 31}},
13       {{ 2,  3,  6,  7},
14        {10, 11, 14, 15},
15        {18, 19, 22, 23},
16        {26, 27, 30, 31}},
17       {{ 4,  5,  6,  7},
18        {12, 13, 14, 15},
19        {20, 21, 22, 23},
20        {28, 29, 30, 31}},
21       {{ 8,  9, 10, 11},
22        {12, 13, 14, 15},
23        {24, 25, 26, 27},
24        {28, 29, 30, 31}},
25       {{16, 17, 18, 19},
26        {20, 21, 22, 23},
27        {24, 25, 26, 27},
28        {28, 29, 30, 31}}};
```

```
30      // Create a Scanner
31      Scanner input = new Scanner(System.in);
32
33      for (int i = 0; i < 5; i++) {
34        System.out.println("Is your birthday in Set" + (i + 1) + "?");
35        for (int j = 0; j < 4; j++) {
36          for (int k = 0; k < 4; k++)
37            System.out.printf("%4d", dates[i][j][k]);
38          System.out.println();
39        }
40
41        System.out.print("\nEnter 0 for No and 1 for Yes: ");
42        answer = input.nextInt();
43
44        if (answer == 1)
45          day += dates[i][0][0];
46      }
47
48      System.out.println("Your birth day is " + day);
49    }
50  }
```