

Revision Exercise 5

(1) Answer each of the following:

- a. Lists and tables of values can be stored in _____.
- b. The elements of an array are related by the fact that they have the same _____ and _____.
- c. The number used to refer to a particular element of an array is called its _____.
- d. The process of placing the elements of an array in order is called _____ the array.
- e. The process of determining if an array contains a certain key value is called _____ the array.
- f. An array that uses two subscripts is referred to as a _____ array.
- g. Java stores lists of values in _____.
- h. The elements of an array are related by the fact that they _____.
- i. When referring to an array element, the position number contained within brackets is called a _____.
- j. The names of the four elements of array **p** are _____, _____, _____ and _____.
- k. Naming an array, stating its type and specifying the number of dimensions in the array is called _____ the array.
- l. The process of placing the elements of an array into either ascending or descending order is called _____.
- m. In a double-subscripted array, the first subscript identifies the _____ of an element and the second subscript identifies the _____ of an element.
- n. An m -by- n array contains _____ rows, _____ columns and _____ elements.
- o. The name of the element in row 3 and column 5 of array **d** is _____.

(2) State whether each of the following is *true* or *false*. If *false*, explain why.

- a. An array can store many different types of values.
- b. An array subscript should normally be of data type **float**.
- c. An individual array element that is passed to a method and modified in that method will contain the modified value when the called method completes execution.

(3) Answer the following questions regarding an array called **fractions**.

- a. Define a constant variable **ARRAY_SIZE** initialized to 10.
- b. Declare an array with **ARRAY_SIZE** elements of type **float** and initialize the elements to 0.
- c. Name the fourth element of the array.
- d. Refer to array element 4.
- e. Assign the value **1.667** to array element 9.
- f. Assign the value **3.333** to the seventh element of the array.
- g. Sum all the elements of the array using a **for** repetition structure. Define the integer variable **x** as a control variable for the loop.

(4) Answer the following questions regarding an array called **table**.

- a. Declare and create the array as an integer array and with 3 rows and 3 columns. Assume the constant variable **ARRAY_SIZE** has been defined to be 3.
- b. How many elements does the array contain?
- c. Use a **for** repetition structure to initialize each element of the array to the sum of its subscripts. Assume the integer variables **x** and **y** are declared as control variables.

(5) Find the error in each of the following program segments and correct the error.

- a.

```
final int ARRAY_SIZE = 5;
ARRAY_SIZE = 10;
```
- b.

```
Assume int b[] = new int[ 10 ];
for ( int i = 0; i <= b.length; i++ )
    b[ i ] = 1;
```
- c.

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

(6) Consider a 2- by-3 integer array **t**.

- a. Write a declaration for **t**.
- b. How many rows does **t** have?
- c. How many columns does **t** have?
- d. How many elements does **t** have?
- e. Write the names of all the elements in the second row of **t**.
- f. Write the names of all the elements in the third column of **t**.
- g. Write a single statement that sets the element of **t** in row 1 and column 2 to

zero.

- h. Write a series of statements that initializes each element of **t** to zero. Do not use a repetition structure.
- i. Write a nested **for** structure that initializes each element of **t** to zero.
- j. Write a statement that inputs the values for the elements of **t** from the keyboard.
- k. Write a series of statements that determines and prints the smallest value in array **t**.
- l. Write a statement that displays the elements of the first row of **t**.
- m. Write a statement that totals the elements of the fourth column of **t**.
- n. Write a series of statements that prints the array **t** in neat, tabular format. List the column subscripts as headings across the top and list the row subscripts at the left of each row.

(7) Use a single-subscripted array to solve the following problem. A company pays its salespeople on a commission basis. The salespeople receive \$200 per week plus 9% of their gross sales for that week. For example, a salesperson who grosses \$5000 in sales in a week receives \$200 plus 9% of \$5000 or a total of \$650. Write a program (using an array of counters) that determines how many of the salespeople earned salaries in each of the following ranges (assume that each salesperson's salary is truncated to an integer amount):

- a. \$200-\$299
- b. \$300-\$399
- c. \$400-\$499
- d. \$500-\$599
- e. \$600-\$699
- f. \$700-\$799
- g. \$800-\$899
- h. \$900-\$999
- i. \$1000 and over

(8) What does the following program do?

```
1. // WhatDoesThisDo.java
2. import java.awt.*;
3. import java.applet.Applet;
4.
5. public class WhatDoesThisDo extends Applet {
6. int result;
```

```

7.
8.     public void init()
9.     {
10.         int a[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
11.
12.         result = whatIsThis( a, a.length );
13.     }
14.
15.     public int whatIsThis( int b[], int size )
16.     {
17.         if ( size == 1 )
18.             return b[ 0 ];
19.         else
20.             return b[ size - 1 ] + whatIsThis( b, size
21. - 1 );
22.     }
23.
24.     public void paint( Graphics g)
25. {
26.
27. g.drawString( "Result is " + result , 25 , 25);
28.
29. }
30.
31. }

```

(9) What does the following program do?

```

1. // WhatDoesThisDo2.java
2. import java.awt.*;
3. import java.applet.Applet;
4.
5. public class WhatDoesThisDo2 extends Applet {
6.     int yPosition;
7.     int a[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
8.     public void paint( Graphics g )
9.     {
10.         yPosition = 25;
11.         someFunction( a, 0, g );

```

```

12.     }
13.
14.     public void someFunction( int b[], int x, Graphics
      g )
15.     {
16.         if ( x < b.length ) {
17.             someFunction( b, x + 1, g );
18.             g.drawString(String.valueOf( b[x] ) , 25 ,
      yPosition);
19.             yPosition += 15;
20.         }
21.     }
22. }

```

(10) (*Airline Reservations System*) A small airline has just purchased a computer for its new automated reservations system. You have been asked to program the new system. You are to write a program to assign seats on each flight of the airline's only plane (capacity: 10 seats).

Your program should display the following menu of alternatives:

```

      Please type 1 for "smoking"
      Please type 2 for "nonsmoking"

```

If the person types 1, your program should assign a seat in the smoking section (seats 1-5). If the person types 2, your program should assign a seat in the nonsmoking section (seats 6-10). Your program should then print a boarding pass indicating the person's seat number and whether it is in the smoking or nonsmoking section of the plane.

Use a single-subscripted array to represent the seating chart of the plane. Initialize all the elements of the array to 0 to indicate that all seats are empty. As each seat is assigned, set the corresponding elements of the array to 1 to indicate that the seat is no longer available.

Your program should, of course, never assign a seat that has already been assigned. When the smoking section is full, your program should ask the person if it is acceptable to be placed in the nonsmoking section (and vice versa). If yes, make the appropriate seat assignment. If no, print the message **"Next flight leaves in 3 hours."**

(11) Use a double-subscripted array to solve the following problem. A company has four salespeople (1 to 4) who sell five different products (1 to 5). Once a day, each salesperson passes in a slip for each different type of product sold. Each slip contains:

- 1.The salesperson number
- 2.The product number
- 3.The total dollar value of that product sold that day

Thus, each salesperson passes in between 0 and 5 sales slips per day. Assume that the information from all of the slips for last month is available. Write a program that will read all this information for last month's sales and summarize the total sales by salesperson by product. All totals should be stored in the double-subscripted array **sales**. After processing all the information for last month, display the results in tabular format with each of the columns representing a particular salesperson and each of the rows representing a particular product. Cross total each row to get the total sales of each product for last month; cross total each column to get the total sales by salesperson for last month. Your tabular printout should include these cross totals to the right of the totaled rows and to the bottom of the totaled columns.

(12) (*The Sieve of Eratosthenes*) A prime integer is any integer that is evenly divisible only by itself and 1. The Sieve of Eratosthenes is a method of finding prime numbers. It operates as follows:

- a. Create an array with all elements initialized to 1 (true). Array elements with prime subscripts will remain 1. All other array elements will eventually be set to zero.
- b. Starting with array subscript 2 (subscript 1 must be prime), every time an array element is found whose value is 1, loop through the remainder of the array and set to zero every element whose subscript is a multiple of the subscript for the element with value 1. For array subscript 2, all elements beyond 2 in the array that are multiples of 2 will be set to zero (subscripts 4, 6, 8, 10, etc.); for array subscript 3, all elements beyond 3 in the array that are multiples of 3 will be set to zero (subscripts 6, 9, 12, 15, etc.); and so on.

When this process is complete, the array elements that are still set to one indicate that the subscript is a prime number. These subscripts can then be printed. Write a program that uses an array of 1000 elements to determine and print the prime numbers between 1 and 999. Ignore element 0 of the array.

Answer

(1) a) Arrays. b) Name, type. c) Subscript. d) Sorting. e) Searching. f) Double-subscripted.

g) arrays. h) have the same name and type. i) subscript. j) `p[0]`, `p[1]`, `p[2]`, and `p[3]` k) declaring and instantiating. l) sorting. m) row, column. n) `m`, `n`, `m * n`
o) `d[2][4]`

(2)

- a. False. An array can store only values of the same type.
- b. False. An array subscript must be an integer or an integer expression.
- c. False for individual primitive-data-type elements of an array because they are passed call-by-value. If an entire array is passed to a method, then any modifications to the array elements will be reflected in the original. Also, an individual element of a class type passed to a method is passed call-by-reference and changes to the object will be reflected in the original array element.

(3)

```
a. final int ARRAY_SIZE = 10;
b. float fractions[] = new float[ ARRAY_SIZE ];
c. fractions[ 3 ]
d. fractions[ 4 ]
e. fractions[ 9 ] = 1.667;
f. fractions[ 6 ] = 3.333;
g. float total = 0;
   for ( int x = 0; x < fractions.length; x++ )
       total += fractions[ x ];
```

(4)

```
a. int table[][] = new int[ ARRAY_SIZE ][ ARRAY_SIZE ];
b. Nine.
c. for ( int x = 0; x < table.length; x++ )
    for ( int y = 0; y < table[ x ].length; y++ )
        table[ x ][ y ] = x + y;
```

(5)

- a. Error: Assigning a value to a constant variable using an assignment statement.
Correction: Assign the correct value to the constant variable in a **final int ARRAY_SIZE** declaration or create another variable.
- b. Error: Referencing an array element outside the bounds of the array (**b[10]**).
Correction: Change the **<=** operator to **<**.
- c. Error: Array subscripting done incorrectly.
Correction: Change the statement to **a[1][1] = 5;**.

(6)

- a. Write a declaration for **t**.
ANS: int t[][] = new int[2][3];
- b. How many rows does **t** have?
ANS: two
- c. How many columns does **t** have?
ANS: three
- d. How many elements does **t** have?
ANS: six
- e. Write the names of all the elements in the second row of **t**.
ANS: t[1][0], t[1][1], t[1][2]
- f. Write the names of all the elements in the third column of **t**.
ANS: t[0][2], t[1][2]
- g. Write a single statement that sets the element of **t** in row 1 and column 2 to zero.
ANS: t[0][1] = 0;
- h. Write a series of statements that initializes each element of **t** to zero. Do not use a repetition structure.
ANS:
t[0][0] = 0;
t[0][1] = 0;
t[0][2] = 0;
t[1][0] = 0;
t[1][1] = 0;
t[1][2] = 0;

- i. Write a nested **for** structure that initializes each element of **t** to zero.

ANS:

```
for ( int j = 0; j < t.length; j++ )
    for ( int k = 0; k < t[ j ].length; k++ )
        t[ j ][ k ] = 0;
```

- j. Write a statement that inputs the values for the elements of **t** from the keyboard.

ANS:

```
for ( int r = 0; r < t.length; r++ )
    for ( int c = 0; c < t[ r ].length; c++ )
        t[ r ][ c ] = System.in.read();
```

- k. Write a series of statements that determines and prints the smallest value in array **t**.

ANS:

```
// assume small is declared and initialized
for ( int x = 0; x < t.length; x++ )
    for ( int y = 0; y < t[ x ].length; y++ )
        if ( t[ x ][ y ] < small )
            small = t[ x ][ y ];
System.out.println( "Smallest is " + small );
```

- l. Write a statement that displays the elements of the first row of **t**.

ANS: `System.out.println(t[0][0] + " " + t[0][1] + " " + t[0][2]);`

- m. Write a statement that totals the elements of the fourth column of **t**.

ANS: **t** does not have a fourth column.

- n. Write a series of statements that prints the array **t** in neat, tabular format. List the column subscripts as headings across the top and list the row subscripts at the left of each row.

ANS:

```
System.out.println( " 0 1 2" );
for ( int e = 0; e < t.length; e++ ) {
    System.out.print( e + " " );

    for ( int r = 0; r < t[ e ].length; r++ )
        System.out.print( t[ e ][ r ] + " " );

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

```
}
```

(7)

```
// Sales.java
```

```
// Program calculates the amount of pay
```

```
// for a salesperson
```

```
import java.applet.Applet;
```

```
import java.awt.*;
```

```
import java.awt.event.*;
```

```
public class Sales extends Applet
```

```
    implements ActionListener {
```

```
    TextField input;
```

```
    Label prompt;
```

```
    int total[];
```

```
    public void init()
```

```
    {
```

```
        total = new int[ 9 ];
```

```
        for ( int i = 0; i < total.length; i++ )
```

```
            total[ i ] = 0;
```

```
        input = new TextField( 5 );
```

```
        input.addActionListener( this );
```

```
        prompt = new Label( "Enter sales amount:" );
```

```
        add( prompt );
```

```

        add( input );
    }

    public void paint( Graphics g )
    {
        super.paint( g );

        int x = 105, y = 70;

        g.drawString( "Range", 5, 50 );

        g.drawString( "Number", 105, 50 );

        g.drawString( "$200-$299", 5, 70 );

        g.drawString( "$300-$399", 5, 80 );

        g.drawString( "$400-$499", 5, 90 );

        g.drawString( "$500-$599", 5, 100 );

        g.drawString( "$600-$699", 5, 110 );

        g.drawString( "$700-$799", 5, 120 );

        g.drawString( "$800-$899", 5, 130 );

        g.drawString( "$900-$999", 5, 140 );

        g.drawString( "$1000 and over", 5, 150 );

        for ( int i = 0; i < total.length; i++, y += 10 )

            g.drawString( String.valueOf( total[ i ] ), x, y );

    }

    public void actionPerformed((ActionEvent e)

    {

```

```

double dollars = Double.parseDouble( input.getText() );

double salary = dollars * 0.09 + 200;

int x = ( int ) ( salary / 100 );

if ( salary < 0 )

    return;

else if ( x > 9 )

    x = 10;

++total[ x - 2 ];

input.setText( "" );

repaint();

}

}

```

(8) Result is 55

(9) 10

9

8

7

6

5

4

3

2

(10)

`// Plane.java``// Airline Reservation System program``import java.applet.Applet;``import java.awt.*;``import java.awt.event.*;``public class Plane extends Applet``implements ActionListener {` `TextField input;` `Label prompt1, prompt2;` `Button yesButton, noButton;` `int section, seats[], smoking;` `int nonsmoking, people;` `public void init()``{` `input = new TextField(4);` `input.addActionListener(this);` `prompt1 = new Label("Please type 1 for smoking");` `prompt2 = new Label("Please type 2 for nonsmoking");` `yesButton = new Button("Yes");` `noButton = new Button("No");`

```

yesButton.addActionListener( this );

noButton.addActionListener( this );

// The enabled method has not been introduced up
// to this point in the book. It is used here to
// disable the buttons by "graying" them.

yesButton.setEnabled( false );

noButton.setEnabled( false );

seats = new int[ 11 ];

smoking = 6;

nonsmoking = 1;

add( prompt1 );

add( prompt2 );

add( input );

add( yesButton );

add( noButton );

}

public void actionPerformed((ActionEvent e)

{

    if ( e.getSource() == input && people <= 10 ) {

        section = Integer.parseInt( input.getText() );

        String s = "";

        if ( section == 1 ) {

```

```

if ( smoking <= 10 && seats[ smoking ] == 0 ) {

    s = "Smoking. Seat #" + smoking;

    seats[ smoking++ ] = 1;

    people++;

}

else if ( smoking > 10 && nonsmoking <= 5 ) {

    // enable buttons

    yesButton.setEnabled( true );

    noButton.setEnabled( true );

    s = "Smoking is full. Non-smoking?";

}

else

    s = "Next flight leaves in 3 hours.";

}

else if ( section == 2 ) {

    if ( seats[ nonsmoking ] == 0 && nonsmoking <= 5 ) {

        s = "Nonsmoking. Seat #" + nonsmoking;

        seats[ nonsmoking++ ] = 1;

        people++;

    }

    else if ( nonsmoking > 5 && smoking <= 10 ) {

        // enable buttons

```

```

        yesButton.setEnabled( true );

        noButton.setEnabled( true );

        s = "Nonsmoking is full. Smoking?";

    }

    else

        s = "Next flight leaves in 3 hours.";

    }

    else

        s = "Invalid input.";

        showStatus( s );

    }

else if ( e.getSource() == yesButton ) {

    if ( section == 1 ) {

        showStatus( "Your seat assignment is " + nonsmoking );

        seats[ nonsmoking++ ] = 1;

    }

    else { // section is 2

        showStatus( "Your seat assignment is " + smoking );

        seats[ smoking++ ] = 1;

    }

    ++people;

    noButton.setEnabled( false );

```



```

        yesButton.setEnabled( false );

    }

    else if ( e.getSource() == noButton ) {

        showStatus( "Next flight leaves in 3 hours." );

        noButton.setEnabled( false );

        yesButton.setEnabled( false );

    }

}

}

```

(11)

```

// Sales.java

// Program totals sales for salespeople

// and products.

import java.applet.Applet;

import java.awt.*;

import java.awt.event.*;

public class Sales extends Applet

    implements ActionListener {

    Label prompt1, prompt2, prompt3;

    TextField input1, input2, input3;

    double sales[ ][ ], totalPerson[ ], totalProducts[ ];

    int person, product;

```

```

public void init()

{

    sales = new double[ 4 ][ 5 ];

    totalPerson = new double[ 4 ];

    totalProducts = new double[ 5 ];

    prompt1 = new Label( "Enter sales person number: " );

    prompt2 = new Label( "Enter product number: " );

    prompt3 = new Label( "Enter sales amount: " );

    input1 = new TextField( 5 );

    input2 = new TextField( 5 );

    input3 = new TextField( 5 );

    input3.addActionListener( this );

    add( prompt1 );

    add( input1 );

    add( prompt2 );

    add( input2 );

    add( prompt3 );

    add( input3 );

}

public void actionPerformed( ActionEvent e )

{

    showStatus( "" );

```

```

double d = Double.parseDouble( input3.getText() );

person = Integer.parseInt( input1.getText() );

product = Integer.parseInt( input2.getText() );

if ( person >= 1 && person < 5 &&

        product >= 1 && product < 6 ) {

        sales[ person - 1 ][ product - 1 ] += d;

    }

else

        showStatus( "Invalid input!" );

repaint();

}

public void paint( Graphics g )

{

    super.paint( g );

    int x = 100, y = 120;

    double total = 0.0;

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

        totalProducts[ j ] = 0;

    g.drawString( "Product", 5, 110 );

    g.drawString( "1", x += 30, 110 );

    g.drawString( "2", x += 30, 110 );

    g.drawString( "3", x += 30, 110 );

```

```

g.drawString( "4", x += 30, 110 );

g.drawString( "5", x += 30, 110 );

g.drawString( "Total", x += 30, 110 );

for ( int r = 0; r < sales.length; r++ ) {

    g.drawString( "Sales person " + ( r + 1 ), 5, y );

    total = 0.0;

    x = 100;

    for ( int c = 0; c < sales[ r ].length; c++ ) {

        total += sales[ r ][ c ];

        g.drawString( String.valueOf( sales[ r ][ c ] ), x += 30, y );

        totalProducts[ c ] += sales[ r ][ c ];

    }

    g.drawString( String.valueOf( total ), x += 30, y );

    y += 10;

}

g.drawString( "Total", 5, y );

x = 100;

for ( int k = 0; k < totalProducts.length; k++ )

    g.drawString( String.valueOf( totalProducts[ k ] ), x += 30, y );

}

}

```

(12)

```
// Sieve.java
```

```
// Sieve of Eratosthenes
```

```
public class Sieve {  
  
    public static void main( String args[] )  
  
    {  
  
        int count = 0;  
  
        String result = "";  
  
        int a[] = new int[ 1000 ];  
  
        for ( int z = 0; z < a.length; z++ )  
  
            a[ z ] = 1;  
  
        for ( int i = 1; i < a.length; i++ )  
  
            if ( a[ i ] == 1 && i != 1 )  
  
                for ( int j = i; j < a.length; j++ )  
  
                    if ( j % i == 0 && j != i )  
  
                        a[ j ] = 0;  
  
        // range 2 - 197  
  
        for ( int k = 2; k < a.length; k++ )  
  
            if ( a[ k ] == 1 ) {  
  
                result += k + " is prime." + "\n";  
  
                ++count;  
            }  
    }  
}
```

```
    }  
  
    result += "\n" + count + " primes found.";  
  
    System.out.println(result);  
  
    }  
  
}
```