|  |  |
| --- | --- |
| ***A logo of a university  AI-generated content may be incorrect.*** | ***Patuakhali Science and Technology University*** |

Assignment on

***“***Deitel and Liang book Array chapter and Section 7.2 solve***”***

Course Code: CCE-121

Course Title: Object Oriented Programming

Level - I; Semester - II

|  |
| --- |
| **Submitted By**  **Name: M.D. Sakibul Islam Shovon**  **ID:** 2302056  **REG:** 11834  **Session:** 2023-2024  Faculty of Computer Science and Engineering |

|  |
| --- |
| **Submitted To**  **Prof. Dr. Md. Samsuzzaman**  Professor of Computer and Communication Engineering Department  Faculty of Computer Science and Engineering |

**Deitel Book Array Chapter solve**

**7.1 Fill in the blank(s) in each of the following statements:**

**a)** Lists and tables of values can be stored in **arrays** and **arrays of arrays (multidimensional arrays)**.

**b)** An array is a group of **variables** (called elements or components) containing values that all have the same **type**.

**c)** The **enhanced for statement (for-each loop)** allows you to iterate through an array’s elements without using a counter.

**d)** The number used to refer to a particular array element is called the element’s **index (or subscript)**.

**e)** An array that uses two indices is referred to as a(n) **two-dimensional** array.

**f)** Use the enhanced for statement **for(double number : numbers)** to walk through double array numbers.

**g)** Command-line arguments are stored in **String[] args**.

**h)** Use the expression **args.length** to receive the total number of arguments in a command line.

**i)** Given the command java MyClass test, the first command-line argument is **"test"**.

**j)** A(n) **ellipsis**  in the parameter list of a method indicates that the method can receive a variable number of arguments.

**7.2 Determine whether each of the following is true or false. If false, explain why.**

a) An array can store many different types of values.

Ans: False.  
Arrays in Java are *homogeneous* – they can only store values of the same type (e.g., int[], double[], String[]).

b) An array index should normally be of type float.  
Ans: False.  
Array indices in Java must be of integer type (byte, short, int, char), not float or double.

c) An individual array element that’s passed to a method and modified in that method will contain the modified value when the called method completes execution.  
Ans: True.  
When you pass an array element (like arr[2]) to a method, if it’s a primitive type, the value is copied (so modification inside won’t affect the original).  
But since the question says "an individual array element" and arrays hold actual objects for reference types, if the element is a reference type object, modifications inside the method *will persist*.  
So for primitives → False, for references → True.  
Usually in textbooks, the intended answer is: False (for primitive arrays, because only a copy is passed).

d) Command-line arguments are separated by commas.  
Ans: False.  
Command-line arguments are separated by whitespace (spaces or tabs), not commas. Example:

java MyClass hello world

gives args[0] = "hello" and args[1] = "world".

**7.3 Perform the following tasks for an array called fractions:**

a) Declare a constant ARRAY\_SIZE that’s initialized to 10:

Solve:

final int ARRAY\_SIZE = 10;

b) Declare an array with ARRAY\_SIZE elements of type double, and initialize the elements to 0:

Solve:

double[] fractions = new double[ARRAY\_SIZE]; // all elements initialized to 0 by default

c) Refer to array element 4:

Solve:

fractions[4];

d) Assign the value 1.667 to array element 9:

Solve:

fractions[9] = 1.667;

e) Assign the value 3.333 to array element 6:

Solve:

fractions[6] = 3.333;

Full Combined Code Example:

public class FractionsArray {

public static void main(String[] args) {

// a) Declare a constant

final int ARRAY\_SIZE = 10;

// b) Declare array and initialize

double[] fractions = new double[ARRAY\_SIZE];

// c) Refer to element 4

System.out.println("Element 4: " + fractions[4]);

// d) Assign 1.667 to element 9

fractions[9] = 1.667;

// e) Assign 3.333 to element 6

fractions[6] = 3.333;

// Display results

System.out.println("Element 9: " + fractions[9]);

System.out.println("Element 6: " + fractions[6]);

}

}

**7.4 Perform the following tasks for an array called table:**

a) Declare and create the array as an integer array with three rows and three columns (assuming ARRAY\_SIZE is already declared as 3):

int[][] table = new int[ARRAY\_SIZE][ARRAY\_SIZE];

b) How many elements does the array contain?  
A 2D array with 3 rows and 3 columns contains:

3×3=9 elements3 \times 3 = 9 \text{ elements}3×3=9 elements

Answer: 9 elements.

c) Use a for statement to initialize each element to the sum of its indices (assuming x and y are control variables):

for (int x = 0; x < ARRAY\_SIZE; x++) {

for (int y = 0; y < ARRAY\_SIZE; y++) {

table[x][y] = x + y;

}

}

Full Example in Context:

public class TableArray {

public static void main(String[] args) {

final int ARRAY\_SIZE = 3;

// a) Declare and create the 3x3 array

int[][] table = new int[ARRAY\_SIZE][ARRAY\_SIZE];

// c) Initialize each element to the sum of its indices

for (int x = 0; x < ARRAY\_SIZE; x++) {

for (int y = 0; y < ARRAY\_SIZE; y++) {

table[x][y] = x + y;

}

}

// Display the array

for (int x = 0; x < ARRAY\_SIZE; x++) {

for (int y = 0; y < ARRAY\_SIZE; y++) {

System.out.print(table[x][y] + " ");

}

System.out.println();

}

}

}

Output:

0 1 2

1 2 3

2 3 4

**7.5 Find and correct the error in each of the following program segments:**

a)

final int ARRAY\_SIZE = 5;

ARRAY\_SIZE = 10;

Error:  
ARRAY\_SIZE is declared final, meaning it cannot be reassigned.

Correction:  
Either remove the reassignment or remove the final keyword if reassignment is intended. Corrected version if you want a constant:

final int ARRAY\_SIZE = 5; // cannot change ARRAY\_SIZE

Or, if reassignment is needed:

int ARRAY\_SIZE = 5;

ARRAY\_SIZE = 10; // now valid

b)

int[] b = new int[10];

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

b[i] = 1;

Error:  
Array indices go from 0 to b.length - 1. Using i <= b.length will cause ArrayIndexOutOfBoundsException.

Correction:  
Use < instead of <= in the loop condition:

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

b[i] = 1;

c)

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

a[1, 1] = 5;

Error:  
In Java, 2D arrays use separate brackets for each index, not a comma.

Correction:

a[1][1] = 5;

**7.6 Fill in the blanks in each of the following statements:**

a) A one-dimensional array p contains five elements. The names of the third and fourth elements are p[2] and p[3].

b) A one-dimensional array k has three elements. The statement k[1] = 2; sets the value of the second element to 2.

c) A statement to declare a two-dimensional int array r that has 3 rows and 4 columns is int[][] r = new int[3][4];

d) A 5-by-6 array contains 5 rows, 6 columns, and 30 elements.

e) The name of the element in column 5 and row 6 of an array d is d[5][4] .

7.8 Determine whether each of the following is true or false. If false, explain why.

a) To refer to a particular location or element within an array, we specify the name of the array and the order of the element in the array, assuming ordering starts at position 1.  
Ans: False.  
In Java, array indices start at 0, not 1. So the first element is array[0], the second is array[1], etc.

b) An array declaration initializes the elements in the array to the integer 0 by default.  
Ans: True.  
When you declare an integer array in Java, all elements are automatically initialized to 0.

c) To indicate that 200 locations should be reserved for integer array p, you write the declaration:

int p[] = new int[200];

Ans: True.  
This is the correct syntax to create an integer array with 200 elements.

d) For an application that initializes the elements of a twenty-element integer array to zero, it is preferable to use some kind of loop.  
Ans: False (not strictly necessary).  
Integer arrays in Java are automatically initialized to zero, so you don’t need a loop. A loop is only needed if you want to set nonzero values.

e) To access all the elements in a two-dimensional array using a loop, the traversal across rows must be done in the outer loop and the traversal across columns in the inner loop.  
Ans: True.  
The standard approach is:

for (int row = 0; row < numRows; row++) {

for (int col = 0; col < numCols; col++) {

// access element[row][col]

}

}

Rows in the outer loop, columns in the inner loop.

**7.8 Write Java statements to accomplish each of the following tasks:**

a) Display the value of the tenth element of array r:

System.out.println(r[9]); // indices start at 0, so 10th element is index 9

b) Initialize each of the six elements of one-dimensional integer array g to -1:

for (int i = 0; i < 6; i++) {

g[i] = -1;

}

c) Find the maximum of the first one-hundred elements of floating-point array c:

float max = c[0]; // assume array has at least 100 elements

for (int i = 1; i < 100; i++) {

if (c[i] > max) {

max = c[i];

}

}

System.out.println("Maximum value: " + max);

d) Copy a hundred-element array a into a hundred-element array b, but in reverse order:

for (int i = 0; i < 100; i++) {

b[i] = a[99 - i];

}

e) Compute the product of the third to the tenth elements (inclusive) in a hundred-element integer array w:

int product = 1;

for (int i = 2; i <= 9; i++) { // indices 2 to 9 correspond to 3rd to 10th elements

product \*= w[i];

}

System.out.println("Product: " + product);

**7.9 Consider a two-by-three integer array t.**

a) Declare and create t:

int[][] t = new int[2][3]; // 2 rows, 3 columns

b) How many rows does t have?

2

c) How many columns does t have?

3

d) How many elements does t have?

2 \* 3 = 6 elements

e) Access expressions for all elements in row 1 (second row):

t[1][0], t[1][1], t[1][2]

f) Access expressions for all elements in column 2 (third column):

t[0][2], t[1][2]

g) Set element in row 0, column 1 to zero:

t[0][1] = 0;

h) Initialize each element individually to zero:

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) Nested for statement to initialize all elements to zero:

for (int i = 0; i < 2; i++) {

for (int j = 0; j < 3; j++) {

t[i][j] = 0;

}

}

j) Nested for statement to input values from user:

import java.util.Scanner;

Scanner input = new Scanner(System.in);

for (int i = 0; i < 2; i++) {

for (int j = 0; j < 3; j++) {

System.out.print("Enter element t[" + i + "][" + j + "]: ");

t[i][j] = input.nextInt();

}

}

k) Determine and display the smallest value in t:

int min = t[0][0];

for (int i = 0; i < 2; i++) {

for (int j = 0; j < 3; j++) {

if (t[i][j] < min) {

min = t[i][j];

}

}

}

System.out.println("Smallest value: " + min);

l) Single printf statement to display the elements of the first row:

System.out.printf("%d %d %d\n", t[0][0], t[0][1], t[0][2]);

m) Total the elements of the third column (column index 2) without repetition:

int total = t[0][2] + t[1][2];

n) Display contents of t in tabular format with column and row indices:

System.out.print(" "); // spacing for row indices

for (int col = 0; col < 3; col++) {

System.out.print("Col" + col + " ");

}

System.out.println();

for (int i = 0; i < 2; i++) {

System.out.print("Row" + i + " ");

for (int j = 0; j < 3; j++) {

System.out.print(t[i][j] + " ");

}

System.out.println();

}

**Liang book Array Chapter solve**

7.1

Write a program that reads student scores, gets the best score, and then assigns grades based on the following scheme:

* Grade is A if score ≥ best – 5
* Grade is B if score ≥ best – 10
* Grade is C if score ≥ best – 15
* Grade is D if score ≥ best – 20
* Grade is F otherwise

The program prompts the user to enter the total number of students, and then prompts the user to enter all of the scores, and concludes by displaying the grades.

Sample run:

Enter the number of students: 4

Enter 4 scores: 40 55 70 58

Student 0 score is 40 and grade is F

Student 1 score is 55 and grade is C

Student 2 score is 70 and grade is A

Student 3 score is 58 and grade is C

Solution (Java)

import java.util.Scanner;

public class AssignGrades {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

// total number of students

System.out.print("Enter the number of students: ");

int numberOfStudents = input.nextInt();

int[] scores = new int[numberOfStudents];

int best = 0;

// read scores

System.out.print("Enter " + numberOfStudents + " scores: ");

for (int i = 0; i < numberOfStudents; i++) {

scores[i] = input.nextInt();

if (scores[i] > best) {

best = scores[i];

}

}

// assign grades

for (int i = 0; i < numberOfStudents; i++) {

char grade;

if (scores[i] >= best - 5) {

grade = 'A';

} else if (scores[i] >= best - 10) {

grade = 'B';

} else if (scores[i] >= best - 15) {

grade = 'C';

} else if (scores[i] >= best - 20) {

grade = 'D';

} else {

grade = 'F';

}

System.out.println("Student " + i + " score is " + scores[i] + " and grade is " + grade);

}

input.close();

}

}

7.2

Write a program that reads 11 integers, compares each of the first 10 with the 11th, and displays whether the integer is greater, smaller, or equal to the 11th integer.

Answer:

import java.util.Scanner;

public class CompareNumbers {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int[] numbers = new int[11];

System.out.println("Enter 11 integers: ");

for (int i = 0; i < 11; i++) {

numbers[i] = input.nextInt();

}

int last = numbers[10];

for (int i = 0; i < 10; i++) {

if (numbers[i] > last)

System.out.println(numbers[i] + " is greater than " + last);

else if (numbers[i] < last)

System.out.println(numbers[i] + " is smaller than " + last);

else

System.out.println(numbers[i] + " is equal to " + last);

}

}

}

7.3

Write a program that reads integers between 1 and 50 and counts the occurrences of each. Input ends with 0.

Answer:

import java.util.Scanner;

public class CountOccurrences {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int[] counts = new int[51];

System.out.print("Enter numbers between 1 and 50 (end with 0): ");

while (true) {

int num = input.nextInt();

if (num == 0) break;

if (num >= 1 && num <= 50) counts[num]++;

}

for (int i = 1; i <= 50; i++) {

if (counts[i] > 0) {

System.out.println(i + " occurs " + counts[i] + (counts[i] > 1 ? " times" : " time"));

}

}

}

}

7.4

Write a program that reads an unspecified number of scores and determines how many scores are above or equal to the average and how many scores are below the average. Input ends with a negative number.

Answer:

import java.util.Scanner;

public class AnalyzeScores {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int[] scores = new int[100];

int count = 0, sum = 0;

System.out.print("Enter scores (negative to end): ");

while (true) {

int score = input.nextInt();

if (score < 0) break;

scores[count++] = score;

sum += score;

}

double avg = (double) sum / count;

int above = 0, below = 0;

for (int i = 0; i < count; i++) {

if (scores[i] >= avg) above++;

else below++;

}

System.out.println("Average is " + avg);

System.out.println("Scores >= average: " + above);

System.out.println("Scores < average: " + below);

}

}

7.5

Write a program that reads integers (ending with 0) and counts the number of even and odd numbers.

Answer:

import java.util.Scanner;

public class CountEvenOdd {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int even = 0, odd = 0;

System.out.print("Enter numbers (end with 0): ");

while (true) {

int num = input.nextInt();

if (num == 0) break;

if (num % 2 == 0) even++;

else odd++;

}

System.out.println("The number of odd numbers: " + odd);

System.out.println("The number of even numbers: " + even);

}

}

7.6

Write a program that generates 100 random integers between 0 and 9 and displays the count for each number.

Answer:

public class CountRandom {

public static void main(String[] args) {

int[] counts = new int[10];

for (int i = 0; i < 100; i++) {

int num = (int)(Math.random() \* 10);

counts[num]++;

}

for (int i = 0; i < counts.length; i++) {

System.out.println(i + " occurs " + counts[i] + " times");

}

}

}

7.7

Write a method that finds the smallest element in an array of double values.

public static double min(double[] array)

Write a test program that prompts the user to enter 10 numbers, invokes this method to return the minimum value, and displays it.

Answer:

import java.util.Scanner;

public class SmallestElement {

public static double min(double[] array) {

double min = array[0];

for (double v : array) {

if (v < min) min = v;

}

return min;

}

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

double[] numbers = new double[10];

System.out.print("Enter 10 numbers: ");

for (int i = 0; i < 10; i++) {

numbers[i] = input.nextDouble();

}

System.out.println("The minimum number is: " + min(numbers));

}

}

7.8

Write two overloaded methods that return the average of an array:

public static int average(int[] array)

public static double average(double[] array)

Answer:

public class AverageArray {

public static int average(int[] array) {

int sum = 0;

for (int v : array) sum += v;

return sum / array.length;

}

public static double average(double[] array) {

double sum = 0;

for (double v : array) sum += v;

return sum / array.length;

}

public static void main(String[] args) {

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

double[] doubleArr = {1.5, 2.5, 3.5, 4.5};

System.out.println("Average of int array: " + average(intArr));

System.out.println("Average of double array: " + average(doubleArr));

}

}

7.9

Write a method that returns the index of the largest element in an array of integers. If more than one element is equal to the maximum, return the smallest index.

public static int indexOfLargestElement(int[] array)

Answer:

public class LargestIndex {

public static int indexOfLargestElement(int[] array) {

int max = array[0];

int index = 0;

for (int i = 1; i < array.length; i++) {

if (array[i] > max) {

max = array[i];

index = i;

}

}

return index;

}

public static void main(String[] args) {

int[] arr = {1, 5, 3, 5, 2};

System.out.println("Index of largest element: " + indexOfLargestElement(arr));

}

}

7.10

Write a method that returns the index of the smallest element in an array of integers. If more than one element is equal to the minimum, return the smallest index.

public static int indexOfSmallestElement(double[] array)

Answer:

public class SmallestIndex {

public static int indexOfSmallestElement(double[] array) {

double min = array[0];

int index = 0;

for (int i = 1; i < array.length; i++) {

if (array[i] < min) {

min = array[i];

index = i;

}

}

return index;

}

public static void main(String[] args) {

double[] arr = {3.5, 1.2, 5.7, 1.2};

System.out.println("Index of smallest element: " + indexOfSmallestElement(arr));

}

}

7.11

Write two methods to compute the deviation and mean of an array of double values:

public static double deviation(double[] x)

public static double mean(double[] x)

Use the following formula for deviation:

deviation=∑(xi−mean)2n−1\text{deviation} = \sqrt{\frac{\sum (x\_i - mean)^2}{n - 1}}deviation=n−1∑(xi​−mean)2​​

Answer:

import java.util.Scanner;

public class Statistics {

public static double mean(double[] x) {

double sum = 0;

for (double v : x) sum += v;

return sum / x.length;

}

public static double deviation(double[] x) {

double mean = mean(x);

double sum = 0;

for (double v : x) sum += Math.pow(v - mean, 2);

return Math.sqrt(sum / (x.length - 1));

}

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

double[] numbers = new double[10];

System.out.print("Enter 10 numbers: ");

for (int i = 0; i < numbers.length; i++) numbers[i] = input.nextDouble();

System.out.println("The mean is " + mean(numbers));

System.out.println("The standard deviation is " + deviation(numbers));

}

}

7.12

Write a method that reverses an array:

public static int[] reverse(int[] list)

Answer:

public class ReverseArray {

public static int[] reverse(int[] list) {

int[] result = new int[list.length];

for (int i = 0; i < list.length; i++) {

result[i] = list[list.length - 1 - i];

}

return result;

}

public static void main(String[] args) {

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

int[] rev = reverse(arr);

for (int v : rev) System.out.print(v + " ");

}

}

7.13

Write a method that returns a random number between 1 and 54, excluding numbers passed in the argument:

public static int getRandom(int... numbers)

Answer:

public class RandomChooser {

public static int getRandom(int... numbers) {

int num;

while (true) {

num = 1 + (int)(Math.random() \* 54);

boolean excluded = false;

for (int n : numbers) {

if (n == num) {

excluded = true;

break;

}

}

if (!excluded) return num;

}

}

public static void main(String[] args) {

System.out.println("Random number (excluding 2,3,5): " + getRandom(2,3,5));

}

}

7.14

Write a method that returns the gcd of an unspecified number of integers:

public static int gcd(int... numbers)

Answer:

public class GCD {

public static int gcd(int... numbers) {

int gcd = numbers[0];

for (int n : numbers) gcd = gcdTwo(gcd, n);

return gcd;

}

private static int gcdTwo(int a, int b) {

while (b != 0) {

int t = b;

b = a % b;

a = t;

}

return a;

}

public static void main(String[] args) {

System.out.println("GCD of 24, 16, 40: " + gcd(24, 16, 40));

}

}

7.15

Write a method that returns a new array eliminating duplicate values:

public static int[] eliminateDuplicates(int[] list)

Answer:

import java.util.\*;

public class EliminateDuplicates {

public static int[] eliminateDuplicates(int[] list) {

ArrayList<Integer> result = new ArrayList<>();

for (int n : list) {

if (!result.contains(n)) result.add(n);

}

int[] arr = new int[result.size()];

for (int i = 0; i < result.size(); i++) arr[i] = result.get(i);

return arr;

}

public static void main(String[] args) {

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

int[] noDup = eliminateDuplicates(arr);

for (int v : noDup) System.out.print(v + " ");

}

}

7.16

Write a program that randomly generates an array of 100,000 integers and sorts them using selection sort and Java’s built-in Arrays.sort. Display execution time for each.

Answer:

import java.util.\*;

public class SortTime {

public static void selectionSort(int[] arr) {

for (int i = 0; i < arr.length - 1; i++) {

int min = i;

for (int j = i+1; j < arr.length; j++) {

if (arr[j] < arr[min]) min = j;

}

int temp = arr[i]; arr[i] = arr[min]; arr[min] = temp;

}

}

public static void main(String[] args) {

int[] arr1 = new int[100000];

for (int i = 0; i < arr1.length; i++) arr1[i] = (int)(Math.random() \* 100000);

int[] arr2 = Arrays.copyOf(arr1, arr1.length);

long start = System.currentTimeMillis();

selectionSort(arr1);

long end = System.currentTimeMillis();

System.out.println("Selection sort time: " + (end - start) + " ms");

start = System.currentTimeMillis();

Arrays.sort(arr2);

end = System.currentTimeMillis();

System.out.println("Arrays.sort time: " + (end - start) + " ms");

}

}

7.17

Write a program that prompts user to enter number of students, names, and scores, then sorts them in decreasing order of scores.

Answer:

import java.util.Scanner;

public class SortStudents {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Enter number of students: ");

int n = input.nextInt();

String[] names = new String[n];

int[] scores = new int[n];

for (int i = 0; i < n; i++) {

System.out.print("Enter name and score: ");

names[i] = input.next();

scores[i] = input.nextInt();

}

for (int i = 0; i < n - 1; i++) {

int maxIndex = i;

for (int j = i + 1; j < n; j++) {

if (scores[j] > scores[maxIndex]) maxIndex = j;

}

int tempScore = scores[i];

scores[i] = scores[maxIndex];

scores[maxIndex] = tempScore;

String tempName = names[i];

names[i] = names[maxIndex];

names[maxIndex] = tempName;

}

for (int i = 0; i < n; i++) {

System.out.println(names[i] + " : " + scores[i]);

}

}

}

7.18

Implement bubble sort.

Answer:

public class BubbleSort {

public static void bubbleSort(int[] arr) {

boolean swapped;

do {

swapped = false;

for (int i = 0; i < arr.length - 1; i++) {

if (arr[i] > arr[i+1]) {

int temp = arr[i]; arr[i] = arr[i+1]; arr[i+1] = temp;

swapped = true;

}

}

} while (swapped);

}

public static void main(String[] args) {

int[] arr = {5, 1, 4, 2, 8};

bubbleSort(arr);

for (int v : arr) System.out.print(v + " ");

}

}

7.19

Write a method that returns true if the list is already sorted.

public static boolean isSorted(int[] list)

Answer:

public class SortedCheck {

public static boolean isSorted(int[] list) {

for (int i = 0; i < list.length - 1; i++) {

if (list[i] > list[i+1]) return false;

}

return true;

}

public static void main(String[] args) {

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

System.out.println(isSorted(arr));

}

}

7.20

Revise the selection sort to return the largest number in each pass and move it to the end.

Answer:

public class ReviseSelectionSort {

public static void selectionSort(int[] arr) {

for (int i = arr.length - 1; i > 0; i--) {

int maxIndex = 0;

for (int j = 1; j <= i; j++) {

if (arr[j] > arr[maxIndex]) maxIndex = j;

}

int temp = arr[i]; arr[i] = arr[maxIndex]; arr[maxIndex] = temp;

}

}

public static void main(String[] args) {

int[] arr = {64, 25, 12, 22, 11};

selectionSort(arr);

for (int v : arr) System.out.print(v + " ");

}

}

7.21

Simulate a bean machine (like Galton board) where balls drop through pegs.

Answer (simplified):

import java.util.\*;

public class BeanMachine {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Enter number of balls: ");

int balls = input.nextInt();

System.out.print("Enter number of slots: ");

int slots = input.nextInt();

int[] slotCounts = new int[slots];

for (int i = 0; i < balls; i++) {

int pos = 0;

for (int j = 0; j < slots - 1; j++) {

if (Math.random() < 0.5) System.out.print("L");

else {

System.out.print("R");

pos++;

}

}

slotCounts[pos]++;

System.out.println();

}

int max = Arrays.stream(slotCounts).max().getAsInt();

for (int i = max; i > 0; i--) {

for (int j = 0; j < slots; j++) {

if (slotCounts[j] >= i) System.out.print("O");

else System.out.print(" ");

}

System.out.println();

}

}

}

7.22

There are 100 lockers and 100 students. Student 1 opens all lockers. Student 2 closes every second locker. Student 3 toggles every third locker, etc. Find which lockers are open after all students finish.

Answer:

public class LockerPuzzle {

public static void main(String[] args) {

boolean[] lockers = new boolean[100];

for (int student = 1; student <= 100; student++) {

for (int locker = student; locker <= 100; locker += student) {

lockers[locker - 1] = !lockers[locker - 1];

}

}

for (int i = 0; i < lockers.length; i++) {

if (lockers[i]) System.out.println("Locker " + (i+1) + " is open");

}

}

}

**Section 7.2**

Section 7.2

▼7.2.1

Question:  
How do you declare an array reference variable and how do you create an array?

Answer:

* Array reference variable declaration:

int[] myList; // declares an array of int

* Array creation:

myList = new int[10]; // creates an array of 10 integers

One line code:

int[] myList = new int[10];

▼7.2.2

Question:  
When is the memory allocated for an array?

Answer:  
Memory is allocated when the array is created using the new operator (not during declaration).

Example:

int[] list; // no memory yet

list = new int[5]; // memory allocated here

▼7.2.3

Question:  
What is the output of the following code?

int x = 30;

int[] numbers = new int[x];

x = 60;

System.out.println("x is " + x);

System.out.println("The size of numbers is " + numbers.length);

Answer:

x is 60

The size of numbers is 30

Because the array was created when x=30. Subsequent changes do not affect the size of the array.

▼7.2.4

Question:  
Indicate true or false for the following statements:

(a) Every element in an array has the same type.  
(b) The array size is fixed after an array reference variable is declared.  
(c) The array size is fixed after it is created.  
(d) The elements in an array must be of a primitive data type.

Answer:  
(a) True  
(b) False  
(c) True  
(d) False (Objects He can stay (Example: String[]).

▼7.2.5

Question:  
Which of the following statements are valid?

int i = new int(30);

double d[] = new double[30];

char[] r = new char(1..30);

int i[] = (3, 4, 3, 2);

float f[] = {2.3, 4.5, 6.6};

char[] c = new char();

Answer:  
Valid ones:

double d[] = new double[30];

float f[] = {2.3, 4.5, 6.6};

Invalid ones:

* int i = new int(30); (should be new int[30])
* char[] r = new char(1..30); (invalid syntax)
* int i[] = (3, 4, 3, 2); (initializer must use {})
* char[] c = new char(); (array size must be given or initializer used)

▼7.2.6

Question:  
How do you access elements in an array?

Answer:  
By using indexing, starting from 0.

int[] list = {10, 20, 30};

System.out.println(list[0]); // 10

System.out.println(list[2]); // 30

▼7.2.7

Question:  
What is the array index type? What is the lowest index? What is the representation of the third element in an array named a?

Answer:

* Index type: int
* Lowest index: 0
* Third element of array a: a[2]

▼7.2.8

Question:  
Write statements to do the following:

a. Create an array to hold 10 double values.  
b. Assign the value 5.5 to the last element in the array.  
c. Display the sum of the first two elements.  
d. Write a loop that computes the sum of all elements in the array.  
e. Write a loop that finds the minimum element in the array.  
f. Randomly generate an index and display the element of this index in the array.  
g. Use an array initializer to create another array with the initial values 3.5, 5.5, 4.52, and 5.6.

Answer:

double[] list = new double[10]; // (a)

list[list.length - 1] = 5.5; // (b)

System.out.println(list[0] + list[1]); // (c)

// (d) sum of all elements

double sum = 0;

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

sum += list[i];

// (e) minimum element

double min = list[0];

for (int i = 1; i < list.length; i++)

if (list[i] < min) min = list[i];

// (f) random index element

int index = (int)(Math.random() \* list.length);

System.out.println(list[index]);

// (g) array initializer

double[] list2 = {3.5, 5.5, 4.52, 5.6};

▼7.2.9

Question:  
What happens when your program attempts to access an array element with an invalid index?

Answer:  
It throws a ArrayIndexOutOfBoundsException at runtime.

▼7.2.10

Question:  
Identify and fix the errors in the following code:

1 public class Test {

2 public static void main(String[] args) {

3 double[100] r;

4

5 for (int i = 0; i < r.length(); i++);

6 r(i) = Math.random \* 100;

7 }

8 }

Answer (fixed code):

public class Test {

public static void main(String[] args) {

double[] r = new double[100]; // FIX 1

for (int i = 0; i < r.length; i++) { // FIX 2 (remove ;)

r[i] = Math.random() \* 100; // FIX 3 (use [ ] and method call)

}

}

}

▼7.2.11

Question:  
What is the output of the following code?

public class Test {

public static void main(String[] args) {

int list[] = {1, 2, 3, 4, 5, 6};

for (int i = 1; i < list.length; i++)

list[i] = list[i - 1];

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

System.out.print(list[i] + " ");

}

}

Answer:

1 1 1 1 1 1

Section 7.4

▼7.4.1

Question:  
Will the program pick four random cards if you replace lines 22–27 in Listing 7.2 DeckOfCards.java with the following code?

for (int i = 0; i < 4; i++) {

int cardNumber = (int)(Math.random() \* deck.length);

String suit = suits[cardNumber / 13];

String rank = ranks[cardNumber % 13];

System.out.println("Card number " + cardNumber + ": "

+ rank + " of " + suit);

}

Answer:

* The program will pick four random cards, but duplicates may occur because each random pick is independent.
* If you want distinct cards, you should shuffle the deck and pick the first four cards instead of generating random indices.

## Section 7.5

### ▼7.5.1

**Question:**  
Use the arraycopy method to copy the following array to a target array t:

int[] source = {3, 4, 5};

**Answer:**

int[] source = {3, 4, 5};

int[] t = new int[source.length];

System.arraycopy(source, 0, t, 0, source.length);

// Print to verify

for (int i : t) {

System.out.print(i + " ");

}

// Output: 3 4 5

▼7.5.2

Question:  
Once an array is created, its size cannot be changed. Does the following code resize the array?

int[] myList;

myList = new int[10];

// Sometime later you want to assign a new array to myList

myList = new int[20];

Answer:

* This does not resize the original array.
* It creates a new array of size 20 and the reference myList points to it.
* The original array of size 10 is discarded (garbage collected if no references exist).

Section 7.7

▼7.7.1

Question:  
Suppose the following code is written to reverse the contents in an array, explain why it is wrong. How do you fix it?

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

for (int i = 0, j = list.length - 1; i < list.length; i++, j--) {

// Swap list[i] with list[j]

int temp = list[i];

list[i] = list[j];

list[j] = temp;

}

Answer:

* ❌ Problem: i < list.length makes the loop continue beyond the middle, swapping elements twice, returning the original array.
* Fix: Stop at the middle of the array:

for (int i = 0, j = list.length - 1; i < j; i++, j--) {

int temp = list[i];

list[i] = list[j];

list[j] = temp;

}

Section 7.8

▼7.8.1

Question:  
True or false? When an array is passed to a method, a new array is created and passed to the method.

Answer:

* False.
* Arrays are passed by reference; the method receives a reference to the original array, not a copy.

▼7.8.2

Question:  
Show the output of the following two programs:

(a)

public class Test {

public static void main(String[] args) {

int number = 0;

int[] numbers = new int[1];

m(number, numbers);

System.out.println("number is " + number

+ " and numbers[0] is " + numbers[0]);

}

public static void m(int x, int[] y) {

x = 3;

y[0] = 3;

}

}

Answer (a):

number is 0 and numbers[0] is 3

* number is primitive, so passed by value.
* numbers is array, passed by reference; element changes persist.

(b)

public class Test {

public static void main(String[] args) {

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

reverse(list);

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

System.out.print(list[i] + " ");

}

public static void reverse(int[] list) {

int[] newList = new int[list.length];

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

newList[i] = list[list.length - 1 - i];

list = newList;

}

}

Answer (b):

1 2 3 4 5

* list = newList; only changes the local reference inside reverse. Original array remains unchanged.

▼7.8.3

Question:  
Where are the arrays stored during execution? Show the contents of the stack and heap during and after executing displayArray, countLetters, displayCounts in Listing 7.4.

Answer:

* Heap: All arrays (int[], char[]) are stored here.
* Stack: Method calls, primitive local variables, and references to arrays are stored here.
* Array objects themselves never go on the stack, only references.

Section 7.9

▼7.9.1

Question:  
What is wrong with each of the following method headers?

public static void print(String... strings, double... numbers)

public static void print(double... numbers, String name)

public static double... print(double d1, double d2)

Answer:

* String... strings, double... numbers (cross) Varargs must be last parameter.
* double... numbers, String name (cross) Varargs must be last parameter.
* double... print(double d1, double d2) (cross) Syntax is wrong, ... goes after type only: public static double print(double... numbers)

▼7.9.2

Question:  
Can you invoke the printMax method in Listing 7.5 using the following statements?

printMax(1, 2, 2, 1, 4);

printMax(new double[]{1, 2, 3});

printMax(new int[]{1, 2, 3});

Answer:

* printMax(1, 2, 2, 1, 4); valid
* printMax(new double[]{1, 2, 3}); valid
* printMax(new int[]{1, 2, 3}); invalid if method expects double...

Section 7.10

▼7.10.1

Question:  
If high is a very large integer such as the maximum int value 2147483647, (low + high)/2 may cause overflow. How do you fix it to avoid overflow?

Answer:  
Use:

int mid = low + (high - low) / 2;

* This prevents low + high from exceeding Integer.MAX\_VALUE.

▼7.10.2

Question:  
Use Figure 7.9 as an example to show how to apply the binary search approach to a search for key 10 and key 12 in list {2, 4, 7, 10, 11, 45, 50, 59, 60, 66, 69, 70, 79}.

Answer:

* Key 10: found at index 3 after 2–3 iterations.
* Key 12: not found; binary search returns negative insertion point -6.

▼7.10.3

Question:  
If the binary search method returns -4, is the key in the list? Where should the key be inserted if you wish to insert the key into the list?

Answer:

* Key is not in the list.
* Insert at -(returned value) - 1 = 3.

Section 7.11

▼7.11.1

Question:  
Use Figure 7.11 as an example to show how to apply the selection-sort approach to sort {3.4, 5, 3, 3.5, 2.2, 1.9, 2}.

Answer:

* Step 1: find min 1.9 → swap with first → {1.9, 5, 3, 3.5, 2.2, 3.4, 2}
* Step 2: find min 2 → swap with second → {1.9, 2, 3, 3.5, 2.2, 3.4, 5}
* … continue until sorted ascending: {1.9, 2, 2.2, 3, 3.4, 3.5, 5}

▼7.11.2

Question:  
How do you modify the selectionSort method in Listing 7.8 to sort numbers in decreasing order?

Answer:

* Replace if (list[j] < list[minIndex]) with if (list[j] > list[maxIndex]).
* Swap largest element to the front in each iteration.

Section 7.12

▼7.12.1

Question:  
What types of array can be sorted using the java.util.Arrays.sort method? Does this sort method create a new array?

Answer:

* Can sort primitive types (int[], double[], etc.) and objects (String[], etc.) implementing Comparable.
* It does not create a new array; it sorts in place.

▼7.12.2

Question:  
To apply java.util.Arrays.binarySearch(array, key), should the array be sorted in increasing order, in decreasing order, or neither?

Answer:

* The array must be sorted in increasing order.

▼7.12.3

Question:  
Show the output of the following code:

int[] list1 = {2, 4, 7, 10};

java.util.Arrays.fill(list1, 7);

System.out.println(java.util.Arrays.toString(list1));

int[] list2 = {2, 4, 7, 10};

System.out.println(java.util.Arrays.toString(list2));

System.out.print(java.util.Arrays.equals(list1, list2));

Answer:

[7, 7, 7, 7]

[2, 4, 7, 10]

false

Section 7.13

▼7.13.1

Question:  
This book text declares the main method as public static void main(String[] args)

Can it be replaced by one of the following lines?

public static void main(String args[])

public static void main(String[] x)

public static void main(String x[])

static void main(String x[])

Answer:

* First three are valid.
* static void main(String x[]) is invalid because main must be public.

▼7.13.2

Question:  
Show the output of the following program when invoked using:

1. java Test I have a dream
2. java Test "1 2 3"
3. java Test

public class Test {

public static void main(String[] args) {

System.out.println("Number of strings is " + args.length);

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

System.out.println(args[i]);

}

}

Answer:  
1.

Number of strings is 4

I

have

a

dream

Number of strings is 1

1 2 3

Number of strings is 0

Section 8.2

▼8.2.1

Question:  
Declare an array reference variable for a two-dimensional array of int values, create a 4 × 5 int matrix, and assign it to the variable.

Answer:

int[][] matrix;

matrix = new int[4][5];

▼8.2.2

Question:  
Can the rows in a two-dimensional array have different lengths?

Answer:

* Yes, in Java 2D arrays are arrays of arrays, so rows can have different lengths.

▼8.2.3

Question:  
What is the output of the following code?

int[][] array = new int[5][6];

int[] x = {1, 2};

array[0] = x;

System.out.println("array[0][1] is " + array[0][1]);

Answer:

array[0][1] is 2

* Row 0 now points to array x.

▼8.2.4

Question:  
Which of the following statements are valid?

int[][] r = new int[2];

int[] x = new int[];

int[][] y = new int[3][];

int[][] z = {{1, 2}};

int[][] m = {{1, 2}, {2, 3}};

int[][] n = {{1, 2}, {2, 3}, };

Answer:

* Valid: int[][] y = new int[3][];, int[][] z = {{1,2}};, int[][] m = {{1,2},{2,3}};, int[][] n = {{1,2},{2,3},};
* Invalid: int[][] r = new int[2]; (should be new int[2][]), int[] x = new int[]; (size required)

Section 8.3

▼8.3.1

Question:  
Show the output of the following code:

int[][] array = {{1, 2}, {3, 4}, {5, 6}};

for (int i = array.length - 1; i >= 0; i--) {

for (int j = array[i].length - 1; j >= 0; j--)

System.out.print(array[i][j] + " ");

System.out.println();

}

Answer:

6 5

4 3

2 1

▼8.3.2

Question:  
Show the output of the following code:

int[][] array = {{1, 2}, {3, 4}, {5, 6}};

int sum = 0;

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

sum += array[i][0];

System.out.println(sum);

Answer:

1 + 3 + 5 = 9

Section 8.4

▼8.4.1

Question:  
Show the output of the following code:

public class Test {

public static void main(String[] args) {

int[][] array = {{1, 2, 3, 4}, {5, 6, 7, 8}};

System.out.println(m1(array)[0]);

System.out.println(m1(array)[1]);

}

public static int[] m1(int[][] m) {

int[] result = new int[2];

result[0] = m.length;

result[1] = m[0].length;

return result;

}

}

Answer:

2

4

* m.length = 2 (number of rows)
* m[0].length = 4 (number of columns in first row)

Section 8.5

▼8.5.1

Question:  
How do you modify the code so that it also displays the highest count and the student with the highest count?

Answer:

* Keep track of maximum count and index while looping through student counts:

int maxCount = counts[0];

int maxIndex = 0;

for (int i = 1; i < counts.length; i++) {

if (counts[i] > maxCount) {

maxCount = counts[i];

maxIndex = i;

}

}

System.out.println("Highest count is " + maxCount + " by student " + maxIndex);

Section 8.6

▼8.6.1

Question:  
What happens if the input has only one point?

Answer:

* If there is only one point, distance-based computations (like nearest neighbor or distance matrix) may fail or produce trivial results.
* Special case handling is needed to avoid array index errors.

Section 8.7

▼8.7.1

Question:  
What happens if the code in line 51 in Listing 8.4 is changed to

if (row != i && col != j && grid[row][col] == grid[i][j])

Answer:

* Original: checks all neighbors except current cell.
* New condition: only checks cells not in the same row AND not in the same column → diagonal neighbors excluded, which may break correctness for games like Sudoku or pattern checks.

Section 8.8

▼8.8.1

Question:  
Declare an array variable for a three-dimensional array, create a 4 × 6 × 5 int array, and assign its reference to the variable.

Answer:

int[][][] array3D = new int[4][6][5];

▼8.8.2

Question:  
Assume char[][][] x = new char[12][5][2], how many elements are in the array? What are x.length, x[2].length, and x[0][0].length?

Answer:

* Total elements: 12 \* 5 \* 2 = 120
* x.length = 12 (number of 2D arrays)
* x[2].length = 5 (number of rows in 3rd 2D array)
* x[0][0].length = 2 (number of columns in first row of first 2D array)

▼8.8.3

Question:  
Show the output of the following code:

int[][][] array = {{{1, 2}, {3, 4}}, {{5, 6},{7, 8}}};

System.out.println(array[0][0][0]);

System.out.println(array[1][1][1]);

Answer:

1

8