# CS500-01 Fundamentals of Programming

# Assignment 3

Tharinda Embuldeniya

S01996041

# Question 7.17

*Code*

**import** java.util.Scanner;

**public** **class** SortStudent {

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

**int** noStdnt = 0;

**int** count =0;

Scanner input = **new** Scanner (System.***in***);

//user input:Enter number of student

System.***out***.println("Enter the number of student");

noStdnt = input.nextInt();

//based on user input define the array for student name and score

String [] studentName = **new** String [noStdnt];

**int** [] studentScores = **new** **int** [noStdnt];

//run a loop to enter student name and score

**while** (count<noStdnt) {

System.***out***.println("Enter student name");

studentName [count] = input.next();

System.***out***.println("Enter score");

studentScores [count] = input.nextInt();

count++;

}

//calling a method for sort the student scores

*sortToDec*(studentName,studentScores);

//calling a method print the student name based student score

*printStudentOder*(studentName);

}

//method for sorting scores based on descending order using selection sort

**public** **static** **void** sortToDec(String[] name, **int**[] scores) {

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

{

**int** currentMax = scores[i];

**int** currentMaxIndex = i;

// find the max value in the score array/ change the scores

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

**if** (currentMax < scores[j]) {

currentMax = scores [j];

currentMaxIndex = j;

}

}

//Swap score[i] with score[currentmaxindex] if necessary, also change name[i] with name[currentmaxindex]

**if** (currentMaxIndex != i) {

scores[currentMaxIndex] = scores[i];

scores[i] = currentMax;

String temp = name[i];

name[i] = name[currentMaxIndex];

name[currentMaxIndex] = temp;

}

}

}

//method for printing each student names

**public** **static** **void** printStudentOder(String[] name) {

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

{

System.***out***.println(name[i]);

}

}

}

*Screenshot*

**

# Question 7.31

*Code*

**import** java.util.Scanner;

**public** **class** MergerTwoList {

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

Scanner input = **new** Scanner (System.***in***);

//user input for list 1 size and content

System.***out***.println("Enter list1 size and contents");

//intilize the array

**int** n1 = input.nextInt();

**int** [] list1 = **new** **int**[n1];

//input the content into the array

**for** (**int** i = 0; i<n1; i++) {

list1[i] = input.nextInt();

}

//Using bulit in fun to sort

java.util.Arrays.*sort*(list1);

//user input for list 2 size and content

System.***out***.println("Enter list2 size and contents");

//intilize the array

**int** n2 = input.nextInt();

**int** [] list2 = **new** **int**[n2];

//input the content into the array

**for** (**int** i = 0; i<n2; i++) {

list2[i] = input.nextInt();

}

//calling method to sort list2

*SelectionSort*(list2);

//output result for merge list 1 & 2

System.***out***.println("list1 is " + java.util.Arrays.*toString*(list1));

System.***out***.println("list2 is " + java.util.Arrays.*toString*(list2));

//new array for mergelist and calling the method

**int** [] listMerge = *merge*(list1 , list2);

//sort the merge list

*SelectionSort*(listMerge);

//display the result of sorted merge list

System.***out***.println("The merged list is " + java.util.Arrays.*toString*(listMerge));

}

// method for merging two list arrays

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

//intilize merge array with adding length of list1 array and length of list2

**int** n3 = list1.length + list2.length;

**int** [] list3 = **new** **int**[n3];

//intilizing counter for value

**int** count = 0;

//add each value of list1 to list3 & add a counter

**for**(**int** i = 0; i<list1.length; i++) {

list3[i] = list1[i];

count++;

}

//add each value of list2 to list3 where above counter ends from

**for**(**int** j = 0;j<list2.length;j++) {

list3[count++] = list2[j];

}

**return** list3;

}

//based on Book Listing 7.8 and modified the code

// this method for sorting the array

**public** **static** **void** SelectionSort(**int**[] list) {

// go through each element in list

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

// find the minimum value from the list

**int** currentMin = list[i];

**int** currentMinIndex = i;

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

**if** (currentMin > list[j]) {

currentMin = list[j];

currentMinIndex = j;

}

}

// change the currentMinindex , if not equal

**if**(currentMinIndex !=i) {

list[currentMinIndex] = list[i];

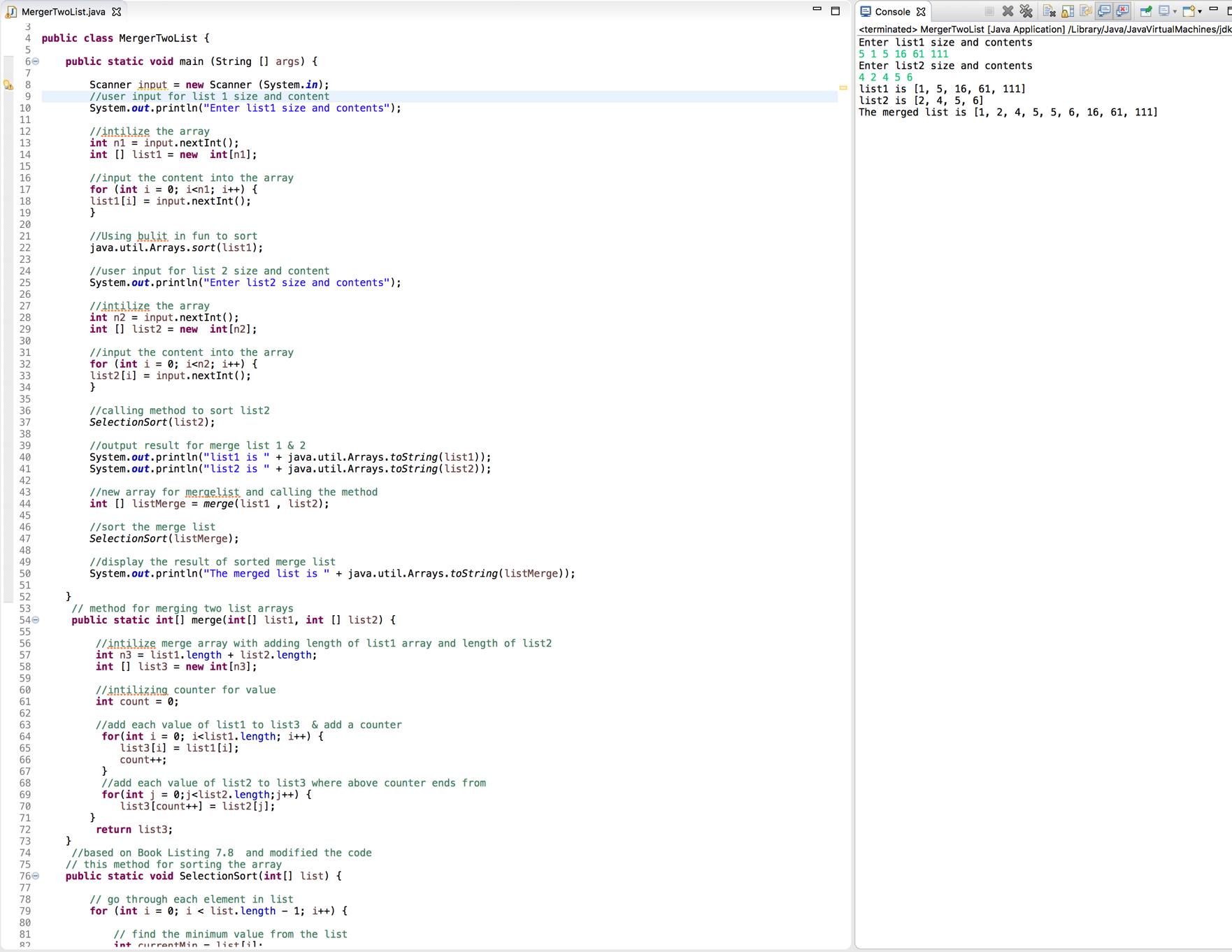
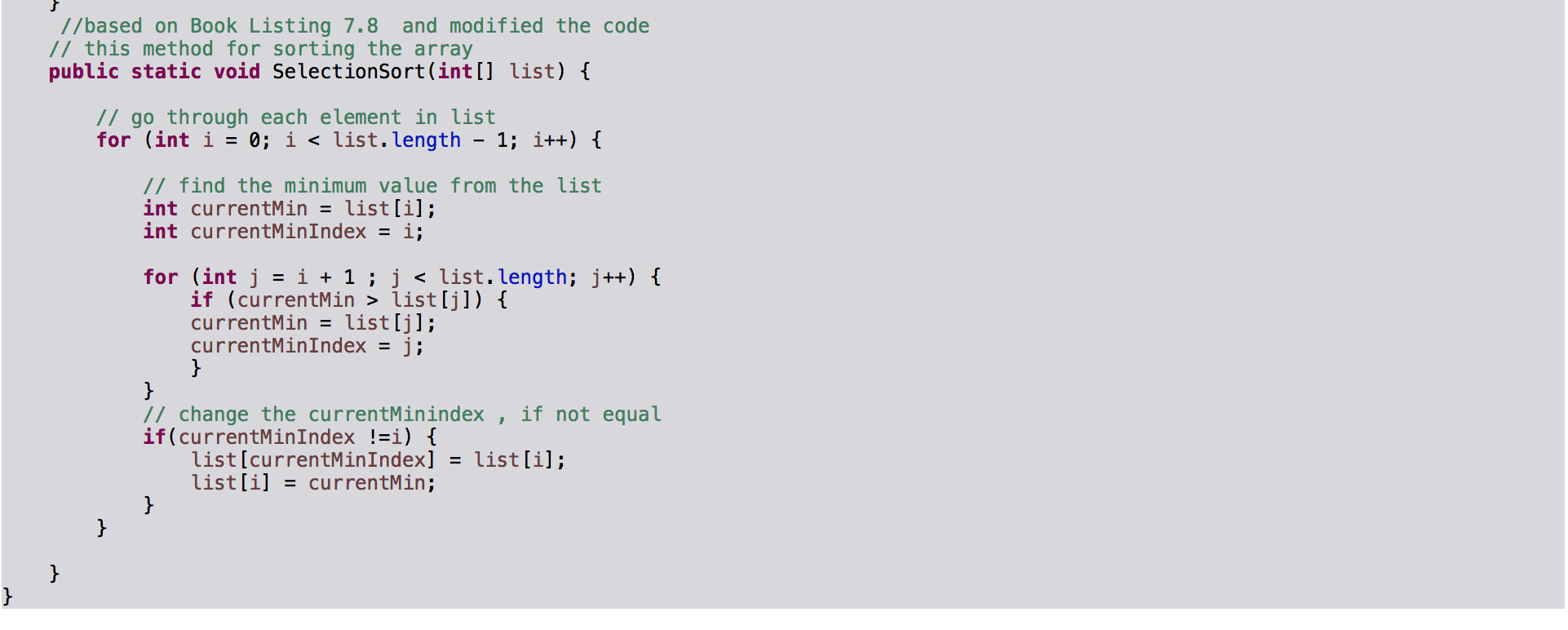
list[i] = currentMin;

}

}

}

}

*Screenshot*

**Question 8.13**

*Code*

**import** java.util.Scanner;

**public** **class** LocateTheLargest {

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

// user input for the size for 2-d array

Scanner input = **new** Scanner (System.***in***);

System.***out***.println("Enter the number of rows and columns of the array");

**int** row = input.nextInt();

**int** col = input.nextInt();

**double** [] [] values = **new** **double** [row][col];

// user input for the array data using for loop

System.***out***.println("Enter the array");

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

**for** (**int** j = 0 ;j < values[i].length; j++) {

values [i][j] = input.nextDouble();

}

}

//pass the user created 2-d array to locateLarget array

// 1-d array variable to hold the data from method locatelLargest

**int** [] locValue = *locateLargest*(values);

//output the result

System.***out***.println("The location of the largest element is at (" + locValue[0]+ " , "+ locValue [1] + " )");

}

// method for finding the largest value and returning a 1-d array

**public** **static** **int**[] locateLargest(**double**[][] a) {

// inside method create 1-d array with 2 elements

**int** [] rowCol = **new** **int** [2];

// set the current max value as 0

**double** currentMax =0;

// looping through each row of the 2-d array

**for** (**int** row = 0 ; row < a.length; row++) {

// looping through each column of the 2-d array

**for** (**int** column = 0; column < a[row].length; column ++) {

// check the largest value with current Max compare to 2-d array's value

**if** (currentMax < a [row] [column])

{

//place the location inside the 1-d array

rowCol[0] = row;

rowCol[1] = column;

//swap or update the current Max

currentMax = a [row] [column];

}

}

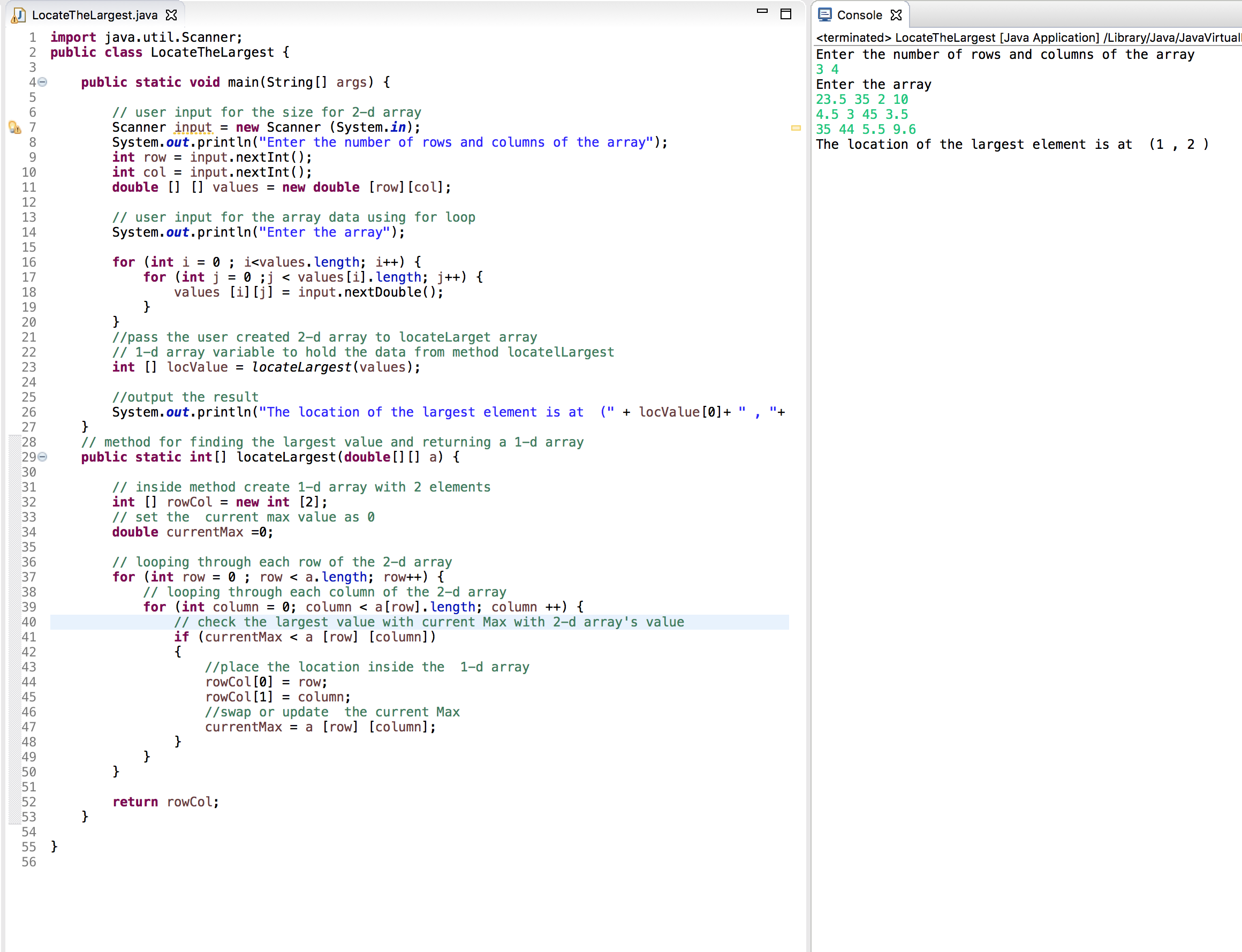
}

**return** rowCol;

}

}

*Screenshot*

****

**Question 8.31**

*Code*

**import** java.util.Scanner;

**public** **class** IntersectingPoint {

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

Scanner input = **new** Scanner (System.***in***);

**double** [] [] points = **new** **double** [4] [2];

//user input and assign points to 2-d arrays using for loop

System.***out***.println("Enter x1,y1,x2,y2,x3,y3,x4,y4:");

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

**for** (**int** j = 0; j < points[i].length; j++) {

points [i] [j] = input.nextDouble();

}

}

// create global 1-d array to store the result from the reference value from getIntersectingPoint

**double** [] result = *getIntersectiongPoint*(points);

//print method for the output

*printResult*(result);

}

// calling method for printing result by the passing the value

**public** **static** **void** printResult(**double**[] result) {

// array value are null then there are parallel

**if** (result == **null**) {

System.***out***.println("The two lines are parallel");

}

// result of the intersecting point

**else** {

System.***out***.println("The intersectiong point is at (" + result [0] + " , " +

result [1] + " )");

}

}

// calling method for cal intersection point

**public** **static** **double**[] getIntersectiongPoint(**double**[][] points) {

// define 1-d array

**double** [] result = **new** **double** [2] ;

//assign the values for each point from 2-d array

**double** x1,y1,x2,y2,x3,y3,x4,y4;

x1 = points [0][0] ;

y1 = points [0][1] ;

x2 = points [1][0] ;

y2 = points [1][1] ;

x3 = points [2][0] ;

y3 = points [2][1] ;

x4 = points [3][0] ;

y4 = points [3][1] ;

/\*simplification based on exercise 3 .25

/(y1 - y2)\* x -(x1 - x2) \* y = (y1 -y2)\* x1 - (x1-x2)\*y1

(y3 - y4)\* x - (x3- x4) \* y = (y3 -y4)\* x3 - (x3- x4)\*y3 \*/

**double** a = y1-y2;

**double** b = -1\*(x1-x2);

**double** c = y3 - y4;

**double** d = -1 \* (x3 - x4);

**double** e = (y1-y2)\* x1 -(x1-x2)\* y1;

**double** f = (y3-y4) \* x3 - (x3-x4)\* y3;

//check for no intersection

**if** (a \* d - b \* c == 0)

{

**return** **null**;

}

**else**

{

// find the intersecting point

result [0] = (e \* d - b \* f) / (a \* d - b \* c);

result [1] = (a \* f - e \* c) / (a \* d - b \* c);

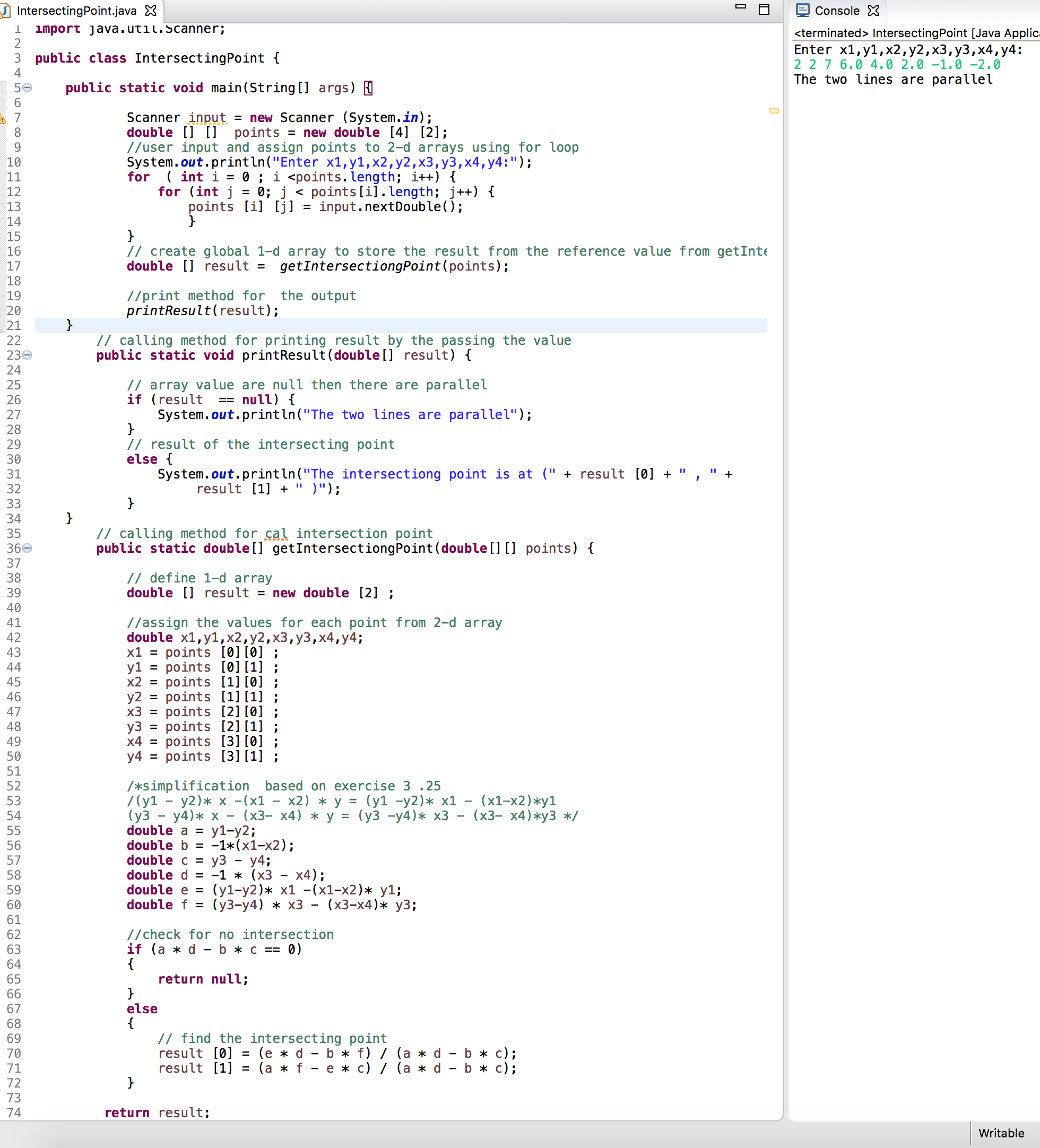
}

**return** result;

}

}

*Screenshot*

**