# General Quiz Information

IOA Course: AP Computer Science

Quiz Name: Unit 8 Quiz

Point Value: 40

Attempts Allowed: 3

## Associated Learning Objectives

|  |  |
| --- | --- |
| ENDURING UNDERSTANDING  VAR-2  To manage large amounts of data or complex relationships in data, programmers write code that groups the data together into a single data structure without creating individual variables for each value. | |
| LEARNING OBJECTIVE  VAR-2.F  Represent collections of related primitive or object reference data using two-dimensional (2D) array objects. | ESSENTIAL KNOWLEDGE  VAR-2.F.1  2D arrays are stored as arrays of arrays. Therefore, the way 2D arrays are created and indexed is similar to 1D array objects.X  EXCLUSION STATEMENT—(EK VAR-2.F.1):  2D array objects that are not rectangular are outside the scope of the course and AP Exam.  VAR-2.F.2  For the purposes of the exam, when accessing the element at arr[first][second], the first index is used for rows, the second index is used for columns.  VAR-2.F.3  The initializer list used to create and initialize a 2D array consists of initializer lists that represent 1D arrays.  VAR-2.F.4  The square brackets [row][col] are used to access and modify an element in a 2D array.  VAR-2.F.5  “Row-major order” refers to an ordering of 2D array elements where traversal occurs across each row, while “column-major order” traversal occurs down each c |

## Auto-Graded Question Types

### Total Points : 40

### True/False

*True must be listed prior to the False.*

Points: #1

#) An out of bounds error happens when an invalid index is used to access a 2D array element, an index that is outside the bounds of valid index values for the 2D array.

\*a. True

b. False

@a) 2D array index values for the row are from 0 through to 1 less than the length of the 2D array. Column index values are valid from 0 through the length of any one of the rows, and we typically use row 0. Any index outside these ranges will cause an out of bounds error.

### True/False

*True must be listed prior to the False.*

Points: #1

#) 2D array element access is done by using the row and the column index values.

\*a. True

b. False

@a) a 2D array, say nums, supports access of elements using valid row and column index values, as in nums[row][col].

### True/False

*True must be listed prior to the False.*

Points: #1

#) In Java, 2D arrays cannot be used to store both primitives and objects, only objects are allowed.

a. True

\*b. False

a) In Java, 2D arrays can be declared to store either objects or primitives.

### True/False

*True must be listed prior to the False.*

Points: #1

#) 2D arrays are objects, before a 2D array is used it should be ‘newed up’, or instantiated.

\*a. True

b. False

a) 2D arrays are references, and as such need to be ‘newed up’ or instantiated

### True/False

*True must be listed prior to the False.*

Points: #1

#) The maximum valid index value for a 2D array row is one less than the number of rows in the 2D array. So, if a 2D array has 7 rows, the maximum row index value is 6.

\*a. True

b. False

a) Valid index values in this case, are from 0 to 6 inclusive.

### True/False

*True must be listed prior to the False.*

Points: #1

#) The maximum valid index value for a 2D array column is the same as the number of columns in the 2D array. So, if a 2D array has 7 columns, the maximum column index value is 7.

a. True

\*b. False

b) Valid column index values in this case, are from 0 to 6 inclusive, 7 columns including the 0.

### Multiple Choice

Points: #1

#) Given the code, what is the value of variable, specialNumber, after the code executes?

int[][] specialNumbers = {{77, 99, 33, 44}, {11, 55, 88, 22}, {9, 3, 7, 5}};

int specialNumber = specialNumbers[2][3];

a. 7

b. 22

c. 88

d. 5

@d) Element at row 2, column 3 is 5

### Multiple Choice

Points: #1

#) Given the class declaration for class Candy,

***public class Candy {***

***int weightInGrams;***

***String label;***

***String chocolateType;***

***String filling;***

***}***

What is the correct declaration for a 5 by 5 array of Candy instances?

a. Candy[] boxOfChocs;

b. Candy boxOfChocs;

c. Candy[5][5] boxOfChocs;

\*d. Candy[][] boxOfChocs;

d) The class name should be used, with a double set of square brackets, indicating a 2D array.

### Multiple Choice

Points: #1

#) Given the code below, what do you predict about the outcome?

***int[] numbers = {{1,2,3,4}, {4,55,77,88}, {8,6,7,3}};***

***for (int[] rowOfNumbers : nums) {***

***for (int val : rowOfNumbers) {***

***if (val % 2 == 0)***

***System.out.print(val + " ");***

***}***

***}***

a. The content of the even rows are printed

b. The content of the odd rows are printed

\*c. All the even numbers are printed

d. nothing is printed

@c) The check is done, if val % 2 ==0. This is a test for an even number, so the even numbers are printed as the 2D array is traversed.

### Multiple Choice

Points: #1

#) Given the code below, and correctly defined 2D array of ints, nums, what can you say about the order in which the data is printed?

***for (int mystery1 = 0; mystery1 < nums.length; mystery1++) {***

***for (int mystery2 = 0; mystery2 < nums[0].length; mystery2++) {***

***System.out.print(nums[mystery1][mystery2] + " ");***

***}***

***}***

a. This is Column-major order

\*b. This is Row-major order.

c. This is an example of random order

d. This is 2D ordering

@b) Rows are accessed and then columns, so this is Row-major ordering.

### Multiple Choice

Points: #1

#) Given the code below, what do you predict about the outcome?

***int[] numbers = {{1,2,3,4}, {4,55,77,88}, {8,6,7,3}};***

***int mystery = 0;***

***for (int[] rowOfNumbers : nums) {***

***for (int val : rowOfNumbers) {***

***mystery += val;***

***}***

***}***

***System.out.print(mystery);***

a. All of the elements are printed

\*b. The sum of all the elements is printed.

c. Each element is assigned the value, val

d. nothing is printed

@b) This nested loop iterates through all of the elements of the 2D array, adding up these elements. The sum is printed.

### Multiple Choice

Points: #1

#) Given the code that implements a linear search in a 2D array, it has been found not to work for a specific number. The number is represented as variable, *target*, with data type, int. Examine the code and determine why there is a semantic error, and specifically what number would reveal the semantic error.

Note that a semantic error is an error in the logic of the code, whereas a syntax error is an error in the way the language is used.

***int[][] nums = {{1,2,3,4,5}, {6,7,8,9,10}, {1,3,7,9,10}, {1,2,3,11,9}};***

***int row = 0;***

***while (row < nums.length - 1) {***

***int col = 0;***

***while (col < nums[0].length) {***

***if (nums[row][col] == target )***

***System.out.println(“Found the number!);***

***col++;***

***}***

***row++;***

***}***

a. Any of the numbers in the last row would reveal the error, because the last row is not being included in the search.

b. The number, 5, would reveal the semantic error, because it is the last element in the first row. The first row is not included because the outer loop condition is that row be less than nums.length - 1

\*c. The number, 11, would reveal the semantic error, because it is in the last row. The last row is not included because the outer loop condition is that row be less than nums.length - 1

d. Any of the numbers in the last column would reveal the error. The last column is not included because the outer loop condition is that the column be less than nums.length – 1

@c) The number 11 is the only 11 in the 2D array, all the other numbers are listed in prior rows, or have duplicates before the last row. The comparison should be while (row < nums.length), so as to include the last row.

### Multiple Choice

Points: #1

#) Given the code below, what does the resulting 2D array look like?

***int[][] nums = {{1, 4, 7}, {2, 5, 8}, {3, 6, 9}};***

***for (int row = 1; row < nums.length; row++) {***

***for (int col = 0; col < nums[0].length; col++) {***

***nums[row - 1][col] = nums[row][col];***

***}***

***}***

a. {{ 1, 4, 7}, { 2, 5, 8}, { 3, 6, 9}} - no changes

b. {{ 2, 5, 8}, { 3, 6, 9},{ 1, 4, 7}}

\*c. {{ 2, 5, 8}, { 3, 6, 9}, { 3, 6, 9}}

d. {{ 2, 5, 8}, { 3, 6, 9}, { 2, 5, 8}}

@c) The prior row is assigned the current row’s values, to the second last one.

### Multiple Choice

Points: #1

#) Given the code below, what do you predict about the outcome?

***double[] numbers = {{1,2,3,4}, {4,55,77,88}, {8,6,7,3}};***

***double mystery = 0;***

***for (int[] rowOfNumbers : nums) {***

***for (int val : rowOfNumbers) {***

***mystery += val;***

***}***

***}***

***System.out.print(mystery/(numbers.length\*numbers[0].length));***

a. All of the elements are printed

b. The sum of all the elements is printed.

c. Each element is assigned the value, val

\*d. The average value of the elements is printed.

@b) Average is defined as the sum of the numbers divided by the number of numbers, the average of the numbers is printed.

### Multiple Choice

Points: #1

#) Given the code below, and correctly defined 2D array of ints, nums, what can you say about the order in which the data is printed?

***for (int mystery1 = 0; mystery1 < nums[0].length; mystery1++) {***

***for (int mystery2 = 0; mystery2 < nums.length; mystery2++) {***

***System.out.print(nums[mystery2][mystery1] + " ");***

***}***

***}***

\*a. This is Column-major order

b. This is Row-major order.

c. This is an example of random order

d. This is 2D ordering

@b) Data is read down the columns, so this is Column-major ordering.

### Multiple Choice

Points: #1

#) Given the code below, what does the resulting 2D array look like?

***int[][] nums = {{1, 4, 7}, {2, 5, 8}, {3, 6, 9}};***

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

***for (int col = 0; col < nums[0].length; col++) {***

***nums[row][col] = nums[row][col] - (row%2);***

***}***

***}***

a. {{1, 4, 7}, {2, 5, 8}, {3, 6, 9}}, no changes

b. {{2, 5, 8}, {3, 6, 9},{1, 4, 7}}

c. {{ 2 5 8}, { 3 6 9}, { 1, 5, 8}}

\*d. {{1,4,7},{1,4,7},{3,6,9}}

@d) Evaluate row 1, as rows 0 and 2 give 0 with % 2.

### ============================================

### Multiple Select

Points: #3

#) The 2D array, mathArr,

***int[][] mathArr***

***= {{1, 2, 3, 4, 5}, {6, 7, 8, 9, 10}, {11, 12, 13, 14, 15},{16, 17, 18, 19, 20}};***

should be printed in row major form like this,

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

Select 3 code snippets that accomplish this.

\*a. for (int[] row : mathArr) {

for (int item : row) {

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

}

System.out.println("");

}

\*b. for (int i = 0; i < mathArr.length; i++) {

for (int j = 0; j < mathArr[0].length; j++) {

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

}

System.out.println("");

}

\*c. int i = 0;

while (i < mathArr.length) {

int j = 0;

while (j < mathArr[0].length) {

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

j++;

}

i++;

System.out.println("");

}

d. for (i = 0; i < mathArr.length; i++) {

for (int j = 0; j < mathArr[0].length; j++) {

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

}

System.out.println("");

}

e. for (int row : mathArr) {

for (int item : row) {

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

}

System.out.println("");

}

f. for (int i = 0; i < mathArr.size(); i++) {

for (int j = 0; j < mathArr[0].length; j++) {

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

}

System.out.println("");

}

@a,b,c) These correctly print the content of the row, followed by the next row. Watch for improper use of size(), and swapping the column and row index values.

### Multiple Choice

Points: #1

#) A 2D array, chances, has an odd number of rows and an odd number of columns, with the number of rows the same as the number of columns. How would you get the element at the center of the rows and columns?

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | X |  |
|  |  |  |

a. chances[chances.length][chances[0].length]

\*b. chances[chances.length/2][chances.length/2]

c. chances[chances.length/2, chances.length/2]

d. chances[chances.length/2]

@b) The index for the middle of the rows is chances.length/2, and since the rows and columns are the same length, the index for the middle of the columns is also chances.length

### Multiple Choice

Points: #1

#)

What is printed by the code,

***int[][] mathArr***

***= {{1, 2, 3, 4, 5},***

***{6, 7, 8, 9, 10},***

***{11, 12, 13, 14, 15},***

***{16, 17, 18, 19, 20}};***

***int sum = 0;***

***int j = 0;***

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

***if (j <= mathArr[0].length)***

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

***j += 2;***

***}***

a. 1 6 11 16

\*b. 1 8 15

c. 6 13 20

d. 1 8 13 20

@b) i iterates 0 through 4, but j is incremented by 2 each time through the loop and the element is only printed if j is less than the size of the row.

### Multiple Select

Points: #2

#) A 2D array,

Question[][] questions;

is populated with Question objects. The question class is defined below.

***public class Question {***

***private int points;***

***private String text;***

***public Question(int points, String text) {..}***

***Question() {..}***

***public int getPoints() {..}***

***public void setPoints(int points) {..}***

***public String getText() {..}***

***public void setText(String text) {..}***

***}***

What are correct ways to access the question on the first row and in the first column in the questions 2D array? Select 2!

a. String questionText = questions[0][0].text;

\*b. String questionText = questions[0][0].getText();

c. String questionText = questions[1][1].text.getText();

d. String questionText = questions[1][0].getText();

e. Question question = questions[0][0].getText();

\*f. Question question = questions[0][0];

String questionText = question.getText();

### Multiple Select

Points: #3

#) Which 3 of the following code snippets prints the first column of data in the 2D array, without error?

***int[][] mathArr***

***= {{1, 2, 3, 4, 5},***

***{6, 7, 8, 9, 10},***

***{11, 12, 13, 14, 15},***

***{16, 17, 18, 19, 20}};***

**\*a. *for (int i = 0; i < mathArr.length; i++) {***

***System.out.println(mathArr[i][0]);***

***}***

**b. *int index = 0;***

***while (index < mathArr.length) {***

***System.out.println(mathArr[index][0]);***

***}***

**\*c*. for (int[] arr : mathArr)***

***System.out.println(arr[0]);***

**d*. for (int[] arr : mathArr[])***

***System.out.println(arr[0]);***

**\*e. *index = 0;***

***while (index < mathArr.length) {***

***System.out.println(mathArr[index][0]);***

***index++;***

***}***

@a,c,e) These access column 0 in each row correctly, and print the first element.

===================================================================

### Multiple Select

Points: #2

#) What can you state about the declaration below? Pick 2. *You can assume that the code is accurate and functional, and class SoccerPlayer and method, team, are correctly defined and functional*.

***SoccerPlayer[][] allPlayers = team();***

a. *team* is a 2D array of SoccerPlayer objects

b. SoccerPlayer is a 2D array of objects

\*c. allPlayers is a 2D array of SoccerPlayer objects

\*d. Method team returns a 2D array of SoccerPlayer objects

e. allPlayers is a 1D array of SoccerPlayer objects

f. allPlayers is definitely null

@c) and d) allPlayers is declared as a 2D array of SoccerPlayer objects, it might be null, but it is set up to receive the 2D array. The code is accurate and error free, so method team musy return a valid 2D array of SoccerPlayer objects.

### 

### Multiple Select

Points: #2

#) Although the code below may look correct, is it the solid code expected from a professional developer? What, if any, are potential problem with the code? Pick 2!

***int[][] allPlayerNumbers = {{9,10,32,12,14}, {11,33,24,19,7}, {33,15,8,4,3}};***

***for (int rowIndex = 0; rowIndex < 3; rowIndex++)***

***for (int colIndex = 0; colIndex < 5; colIndex++)***

***print(allPlayerNumbers[rowIndex][colIndex]);***

a. This code is good, no obvious issues.

b. This code is more efficient, as it uses an enahnced for loop.

c. The square brackets should come after the name of the 2D array, as in *int playerNumbers[][];*

d. There are 3 rows and 5 columns, in Java the number of columns and the numbers of rows must be equal.

\*e. Instead of using a numeric literal for the number of rows, the coder should use allPlayerNumbers.length.

\*f. Instead of using a numeric literal for the number of columns, the coder should use allPlayerNumbers[0].length.

@e) and f) allPlayerNumbers is declared as a 2D array of ints, any traversal should use the length property for both the rows and the columns. You would do this in case the 2D array dimensions are changed.

========================================================

### Essay Question

*Provide the suggested answer to assist with grading by adding an “a.” Only the online teacher can see the suggested answer.*

Points: #10

Type: E

#) Student grades for 5 subject exams are kept in a 2D array. Each column represents the points a student scored in a subject exam. Here is a sample score matrix for a student,

{{71, 82, 93, 74, 75},

{67, 72, 82, 91, 100},

{81, 82, 93, 74, 85},

{86, 87, 78, 89, 100}};

Your job is to complete the method that accepts the 2D array as a parameter, and totals the columns. The method should return a 1D array containig the totals for each column. For instance, in the array above, the totals single dimensional array would store 305 in totals[0], because 71 + 67+ 81 + 86 sums to 305.

Here is the method header, and an outline of the method.

***public static int[] totalPoints(int[][] examScores) {***

***int[] totals = new int[examScores[0].length];***

***// Your code here***

***return totals;***

***}***

***@ Solution)***

***public static int[] totalPoints(int[][] examScores) {***

***int[] totals = new int[examScores[0].length];***

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

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

***totals[j] += examScores[i][j];***

***}***

***}***

***return totals;***

***}***

============================================================