Set 22: Two-Dimensional Arrays

Skill 22.01 Describe how Two-Dimensional Arrays Store Data

Skill 22.01 Concepts

A two-dimensional array in Java is an array inside of an array - or a list within a list. As an example, consider a list of students: Bart, Kyle, Bugs, and Marvin. Each of these students can also have grades associated with them. The grades associated with each student can be stored in a list. Below is a visual representation,

gradebook						
		0	1	2	3	4
		assignment 1	assignment 2	assignment 3	assignment 4	assignment 5
0	Bart	5	4.5	3	4	3.5
1	Kyle	2	3	2.5	4	R
2	Bugs	4	1	3.5	5	4
3	Marvin	5	2	3.5	3	4.5

The *relational database* like structure of 2D arrays provides a means of storing a lot of data at once. Representing data as a collection of rows and columns also allows for quick access to information and/or the ability to pass large amounts of data to methods wherever required.

For example, we can access any data member in the 2D array above by referencing the appropriate row and col.

location	data member
row = 1 col = 3	4
row = 0 col = 4	3.5
row = 3 col = 1	2
row = 2 col = 4	4
row = 1 col = 5	out of bounds

A *2d array* is a specific data structure that stores data as a matrix of *rows* and *columns*

Skill 22.01: Exercise 1

Skill 22.02 Declare, Initialize, and Populate a 2D Array

Skill 22.02 Concepts

Just as we saw with 1D arrays, 2D arrays in java can be declared, initialized, and populated in at least two ways. Each are described below.

Method 1

The code snippet below illustrates one method for declaring, initializing, and populating an array. This method is only useful if we know the identity of the elements we want to store.

```
String gradebook[][] = {{"5","4.5","3","4","3.5"}, {"2","3","2.5","4","R"}, {"4","1","3.5","5","4"},
{"5","2","3.5","3","4.5"}};
```

- **String** - The type of array we want to declare. This can be any type, String, int, double, etc.
- **gradebook[][]** - The name of the array is *gradebook* and the brackets, [][], following the name, tells java that this is a 2D array.
- **{...}** - Encompasses the rows in the array
- **{"5","4.5","3","4","3.5"}** - The data we want to store in each row of the array.

Method 2

The code snippet below illustrates another method for declaring an array. In the below example the array is only declared, not initialized.

```
String gradebook[][];
```

- **String** - The type of array we want to declare. This can be any type, String, int, double, etc.
- **gradebook[][]** - The name of the array is *gradebook* and the brackets, [][], following the name, tells java that this is a 2D array.

Notice that to declare a two-dimensional array we use the same notation as before but include an extra set of brackets. The first set of brackets represents the number of rows, the second set represents the number of columns. To help remember this, think “RC”, as in “RC Cola”.

To initialize the array we use the *new* keyword to tell java this is a new array. Additionally, we need to tell java the datatype the array will store, along with the number of rows and columns. The code snippet below illustrates how to initialize the array we declared above to have 4 rows and 5 columns.

```
gradebook = new String[4][5];
```

While the array above has been declared and initialized, it currently does not hold any values. The below code snippet illustrates how to populate the array with values. If no value is specified, the location will be assigned the value *null*.

```
gradebook[0] = {"5","4.5","3","4","3.5"};
gradebook[1] = {"2","3","2.5","4","R"};
gradebook[2] = {"4","1","3.5","5","4"};
gradebook[3] = {"5","2","3.5","3","4.5"};
```

- **gradebook** - The name of the array in which we want to store the item.
- **[0]** - The row ***index*** in which we want store the item.
- **{"5","4.5","3","4","3.5"}** - The ***values*** of the items in the row

The above illustrates how to populate a 2D array by assigning all the values associated with a row to each row position. The below code illustrates how to populate a specific location,

```
gradebook[0][0] = "5"; //sets position 0, 0 to "5"
gradebook[0][1] = "4.5"; //sets position 0, 1 to "4.5", and so on
gradebook[1][0] = "2";
gradebook[1][1] = "3";
gradebook[2][0] = "4";
gradebook[2][1] = "1";
```

While the above examples illustrate how to declare, initialize, and populate an array of String datatypes. 2D Arrays can store any of the data types we have learned about previously, including objects.

Skill 22.02: Exercise 1

Skill 22.03 Determine the Number of Rows and Columns in a Two-Dimensional Array

Skill 22.03 Concepts

The number of rows and columns in a two-dimensional array (sometimes called a matrix) can be determined as described below,

```
String gradebook[][] = {{"5","4.5","3","4","3.5"},
                        {"2","3","2.5","4","R"},
                        {"4","1","3.5","5","4"},
                        {"5","2","3.5","3","4.5"}};
```

For the matrix above, the number of rows can be determine as follows,

```
gradebook.length; //returns a value of 4 for the number of rows
```

```
gradebook[0].length; //returns a value of 5 for the number columns in row 0
gradebook[1].length; //returns a value of 5 for the number columns in row 1
gradebook[2].length; //returns a value of 5 for the number columns in row 2
```

Skill 22.03: Exercises 1

Skill 22.04 Iterate Over a 2D Array

Skill 22.04 Concepts

Iterating over a 2D array is the process of accessing each element of an array one by one. Accessing each element in a 2D array begins by accessing each row. The first row in the array is always located at *index = 0*, the last row of an array is located at *array.length - 1*. The first element in each row is located at *index = 0*, the last element in each row is located at *array[row].length-1*.

Consider the array below,

```
String gradebook[][] = {{"5","4.5","3","4","3.5"},
                        {"2","3","2.5","4","R"},
                        {"4","1","3.5","5","4"},
                        {"5","2","3.5","3","4.5"}};
```

The code below could be used to determine the number of elements (or length) of each row in the array.

Code	Output

`for(int row = 0; row < gradebook.length; row++){`
 	
`  System.out.println(gradebook[row].length);`
 	
`}`
	5
	5
	5
	5

By combining the above code with another loop, we now access each element in each row,

Code	Output

`for (int row = 0; row < gradebook.length; row++) {`
 	
`  for (int col = 0; col < gradebook[row].length; col++) {`
 	
`  System.out.print(gradebook[row][col] + " , ");`
 	
`  }`
 	
`  System.out.println();`
`}`
	5 , 4.5 , 3 , 4 , 3.5 ,
	2 , 3 , 2.5 , 4 , R ,
	4 , 1 , 3.5 , 5 , 4 ,
	5 , 2 , 3.5 , 3 , 4.5 ,

Skill 22.04: Exercises 1