Exploration

Name:	

A little 2D array practice!

Start a new java file named whatever you want in IntelliJ, and in the `main` method, code up each of the following:

1.

a. Write come code that uses an **initializer list (of lists)** to create a 2D array named `seatingChart` containing the following `String` values in the following order:

Abby	**Don**	**George**	**Kim**
Brian	**Elenor**	**Harry**	**Lenny**
Cathy	**Fred**	**Jill**	**Matt**

Check

b. **New student!** Add **one** line of code to *replace* "Harry" with the new value "Paul"

Check

c. **New seats!** Add a few lines of code to *swap* "Matt" and "Abby" (use a temp variable).

Check

d. **Changing rows!** Add a few lines of code to *swap* the first and second rows (rows 0 and 1), again using a temp variable.

Check

e. Finally, write some code to print out the final 2D seating chart so the output looks like the following, which includes all updates and swaps to the seating chart (see **slide 30** for an easy way to do this; you will need to import `java.util.Arrays`):

```
 ----jGRASP exec: java Feb10
[Brian, Elenor, Paul, Lenny]
[Matt, Don, George, Kim]
[Cathy, Fred, Jill, Abby]

 ----jGRASP: operation complete.
```

Check

Copy/paste your `main` method code below for parts a-e:

PREDICT! What will the following line of code print out if you place it *after* all the other code that you wrote for problem 1 above?

```
System.out.println(seatingChart[0][2] + seatingChart[2][0]);
```

Add it to your code, then run it to **test** your prediction!

Check

---

2. You can continue coding in your `main` method, below problem 1 above.

a. Use the `new` keyword to declare and initialize *two* empty 2D arrays of `int`s named `arr1` that will hold **2 rows of 3** elements, and `arr2` that will hold **3 rows of 2** elements.

Check

b. Then, use *six* individual assignment statements to assign each of the the numbers 1 through 6 to `arr1` so that `arr1` is like this:

```
[1, 2, 3]
[4, 5, 6]
```

Include some code to test by printing out `arr1`.

Check

c. Lastly, use *six* individual assignment statements to assign each of the the numbers 1 through 6 to `arr2` so that `arr2` is like this:

```
[1, 4]
[2, 5]
[3, 6]
```

Include some code to test by printing out `arr2`.

Check

Copy/paste your code below for parts a-c:

PREDICT! What will the following line of code print out if you place it *after* all the other code that you wrote for problem 2 above?

```
System.out.println(arr1[0][2] + arr2[2][0]);
```

Add it to your code, then run it to **test** your prediction!

Check

MULTIPLE CHOICE (record your answers in the table below)

3. What would the following 2D array contain upon initialization?

```
int[][] mystery1 = new int[4][6];
```

a.
```
0  0  0  0  0  0
0  0  0  0  0  0
0  0  0  0  0  0
0  0  0  0  0  0
```

b.
```
0  0  0  0
0  0  0  0
0  0  0  0
0  0  0  0
0  0  0  0
0  0  0  0
```

c.
```
0  0
```

d. null

4. What would the following 2D array contain upon initialization?

```
boolean[][] mystery2 = new boolean[2][3];
```

a.
```
false  false  false
false  false  false
```

b.
```
true  true  true
true  true  true
```

c.
```
0  0  0
0  0  0
```

d.
```
null  null  null
null  null  null
```

5. What would the following 2D array contain upon initialization?

```
String[][] mystery3 = new String[3][1];
```

a.
```
null
null
null
```

b. (3 empty strings)

Your answers!

3	
4	
5	

Confirm your answers!

6. Quick Check!

```
arr = { { "Atlanta", "Baltimore", "Chicago", "Detroit"},
        { "Australia", "Boston", "Cincinnati", "Denver"},
        { "Austin", "Beaumont", "Corpus Christi", "Dallas"}}
```

Which of the following options would be able to change `Australia` to `Athens`?

I. `arr[2][1] = "Athens";`
II. `arr[1][0] = "Athens";`
III. `arr[arr.length -1][0] = "Athens";`

a) I only
b) II only
c) III only
d) II and III only
e) I, II, and III

My answer:	
Check my answer!	
Were you correct? If not, why not?	

Exploration continues on the next page!

7. Quick Check!

Let **`seating`** be a 2D **`String`** array that represents a seating chart in a classroom with exactly 30 seats.

Which of the following options correctly declares the **`seating`** variable and initializes the 2D array that can be used as a seating chart for the classroom?

a) `String[][] seating = new String[30][30];`
b) `String[5][6] seating = new String[5][6];`
c) `String[10][3] seating = new String[][];`
d) `String[][] seating = new String[10][3];`
e) `String[] seating = new String[30];`

My answer:	
Check my answer!	
Were you correct? If not, why not?	

Exploration continues on the next page!

8. Quick Check!

What would be the result of trying to compile and execute the following code?

```
int[] row0nums = {3, 8, 9};
int[] row1nums = {1, 15, 4};
int[] row2nums = {6, 11, 5};
int[] row3nums = {4, 7, 21};

int[][] randNums = new int[3][4];
randNums[0] = row0nums;
randNums[1] = row1nums;
randNums[2] = row2nums;
randNums[3] = row3nums;
```

A. It would compile and execute, and result in the following 2D array stored in `randNums`:

```
3    8    9
1    15    4
6    11    5
4    7    21
```

B. It would compile and execute, and result in the following 2D array stored in `randNums`:

```
3    1    6    4
8    15    11    7
9    4    5    21
```

C. It would result in a compiler error and wouldn't execute.

D. It would execute, but result in an `ArrayIndexOutOfBoundsException`.

My answer:	
	Check my answer!
Were you correct? If not, why not?	

Exploration continues on the next page!

9. 2D Array Methods!

Download this `FunWith2DArrays` class, containing test code in the `main` method and **two** static methods that you need to implement. Implement both methods, then test using the test code.

The test code for **method 1 (`totalElements`)** should output:

```
6
6
8
9
15
14
4
1
```

The test code for **method 2 (`fourCorners`)** should output:

```
hi
go
map
mom
--------------
time
bye
bow
joy
--------------
time
time
time
time
```

Copy/paste the code for your `two` methods below:

```
public static int totalElements(int[][] numArray)
{

}

public static void fourCorners(String[][] strArray)
{

}
```

Compare my implementation

Done!

Submit in Google Classroom:

Solution (back)

1.

a. **Preferred:**

```
public static void main(String[] args)
{
  // a.
  String[][] seatingChart = {{"Abby", "Don", "George", "Kim"},
                             {"Brian", "Elenor", "Harry", "Lenny"},
                             {"Cathy", "Fred", "Jill", "Matt"}};
```

OR:

```
public static void main(String[] args)
{
  // a.
  String[][] seatingChart = {{"Abby", "Don", "George", "Kim"}, {"Brian", "Elenor", "Harry", "Lenny"}, {"Cathy", "Fred", "Jill", "Matt"}};
```

Solution (back)

b. (written after part a)

```
public static void main(String[] args)
{
  // a.
  String[][] seatingChart = {{"Abby", "Don", "George", "Kim"},
                             {"Brian", "Elenor", "Harry", "Lenny"},
                             {"Cathy", "Fred", "Jill", "Matt"}};

  // b.
  seatingChart[1][2] = "Paul";
```

Solution (back)

c. (written after part b)

```
public static void main(String[] args)
{
  // a.
  String[][] seatingChart = {{"Abby", "Don", "George", "Kim"},
                             {"Brian", "Elenor", "Harry", "Lenny"},
                             {"Cathy", "Fred", "Jill", "Matt"}};

  // b.
  seatingChart[1][2] = "Paul";

  // c.
  String temp = seatingChart[0][0]; // save Abby in temp
  seatingChart[0][0] = seatingChart[2][3]; // assign Matt to Abby's spot
  seatingChart[2][3] = temp; // assign temp (Abby) to Matt's spot
```

Solution (back)

d. (written after part c)

```
public static void main(String[] args)
{
  // a.
  String[][] seatingChart = {{"Abby", "Don", "George", "Kim"},
                             {"Brian", "Elenor", "Harry", "Lenny"},
                             {"Cathy", "Fred", "Jill", "Matt"}};

  // b.
  seatingChart[1][2] = "Paul";

  // c.
  String temp = seatingChart[0][0]; // save Abby in temp
  seatingChart[0][0] = seatingChart[2][3]; // assign Matt to Abby's spot
  seatingChart[2][3] = temp; // assign temp (Abby) to Matt's spot

  // d.
  String[] temp2 = seatingChart[0]; //save row 0 (first row) in temp
  seatingChart[0] = seatingChart[1]; // assign row 1 to row 0
  seatingChart[1] = temp2; // assign temp2 (row 0) to row 1
```

Solution (back)

e. (written after part d) -- don't forget the `import`!

```
1  import java.util.Arrays;
2
3  public class Feb10
4  {
5    public static void main(String[] args)
6    {
7      // a.
8      String[][] seatingChart = {{"Abby", "Don", "George", "Kim"},
9                                 {"Brian", "Elenor", "Harry", "Lenny"},
10                                {"Cathy", "Fred", "Jill", "Matt"}};
11
12     // b.
13     seatingChart[1][2] = "Paul";
14
15     // c.
16     String temp = seatingChart[0][0]; // save Abby in temp
17     seatingChart[0][0] = seatingChart[2][3]; // assign Matt to Abby's spot
18     seatingChart[2][3] = temp; // assign temp (Abby) to Matt's spot
19
20     // d.
21     String[] temp2 = seatingChart[0]; //save row 0 (first row) in temp
22     seatingChart[0] = seatingChart[1]; // assign row 1 to row 0
23     seatingChart[1] = temp2; // assign temp2 (row 0) to row 1
24
25     // e.
26     for (String[] innerArr : seatingChart)
27     {
28       System.out.println(Arrays.toString(innerArr));
29     }
30   }
31 }
```

Solution (back)

2.

a.

```
// a.
int[][] arr1 = new int[2][3];
int[][] arr2 = new int[3][2];
```

Solution (back)

2.

b.

```
// a.
int[][] arr1 = new int[2][3];
int[][] arr2 = new int[3][2];

// b.
arr1[0][0] = 1;
arr1[0][1] = 2;
arr1[0][2] = 3;
arr1[1][0] = 4;
arr1[1][1] = 5;
arr1[1][2] = 6;

for (int[] innerArr : arr1)
{
  System.out.println(Arrays.toString(innerArr));
}
```

Solution (back)

2.

c.

```
// a.
int[][] arr1 = new int[2][3];
int[][] arr2 = new int[3][2];

// b.
arr1[0][0] = 1;
arr1[0][1] = 2;
arr1[0][2] = 3;
arr1[1][0] = 4;
arr1[1][1] = 5;
arr1[1][2] = 6;

for (int[] innerArr : arr1)
{
  System.out.println(Arrays.toString(innerArr));
}

// c.
arr2[0][0] = 1;
arr2[0][1] = 4;
arr2[1][0] = 2;
arr2[1][1] = 5;
arr2[2][0] = 3;
arr2[2][1] = 6;

for (int[] innerArr : arr2)
{
  System.out.println(Arrays.toString(innerArr));
}
```

Answer (back)

PREDICT! What will the following line of code print out if you place it *after* all the other code that you wrote for problem 1 above? `System.out.println(seatingChart[0][2] + seatingChart[2][0]);`	**`PaulCathy`**

Answer (back)

PREDICT! What will the following line of code print out if you place it *after* all the other code that you wrote for problem 2 above?
 	
`System.out.println(arr1[0][2] + arr2[2][0]);`
	6
	
	`arr1[0][2] is 3`
	`arr2[2][0] is 3`
	`3 + 3 = 6!`

Note: Because these are both `ints`, and there are *no String components* included in the `println`, the two `ints` are added first and the result is printed (6), rather than **33**

If you want to "force" Java to print these as side by side numbers, **i.e. 33 instead of 6**, add an *empty String* in front:

```
System.out.println("" + arr1[0][2] + arr2[2][0]);
```
→ Now prints: `33`

Answers (back)

3. What would the following 2D array contain upon initialization?

```
int[][] mystery1 = new int[4][6];
```

This creates a 2D array with 4 rows and 6 columns, with `int` default values of 0.

a.
```
0  0  0  0  0  0
0  0  0  0  0  0
0  0  0  0  0  0
0  0  0  0  0  0
```

b.
```
0  0  0  0
0  0  0  0
0  0  0  0
0  0  0  0
0  0  0  0
0  0  0  0
0  0  0  0
0  0  0  0
```

c.
```
0  0
```

d. Null

4. What would the following 2D array contain upon initialization?

```
boolean[][] mystery2 = new boolean[2][3];
```

This creates a 2D array with 2 rows and 3 columns, with `boolean` default values of `false`.

a.
```
false  false  false
false  false  false
```

b.
```
true  true  true
true  true  true
```

c.
```
0  0  0
0  0  0
```

d.
```
null  null  null
null  null  null
```

5. What would the following 2D array contain upon initialization?

```
String[][] mystery3 = new String[3][1];
```

This creates a 2D array with 3 rows and 1 column, with default values of null (since Strings are objects!).

a.
```
null
null
null
```

b. (3 empty strings)

Correct answers

3	**a**
4	**a**
5	**a**

Answer (back)

6. Quick Check!

```
arr = { { "Atlanta", "Baltimore", "Chicago", "Detroit"},
        { "Australia", "Boston", "Cincinnati", "Denver"},
        { "Austin", "Beaumont", "Corpus Christi", "Dallas"}}
```

Which of the following options would be able to change `Australia` to `Athens`?

I. `arr[2][1] = "Athens";`
II. `arr[1][0] = "Athens";`
III. `arr[arr.length -1][0] = "Athens";`

a) I only
b) II only
c) III only
d) II and III only
e) I, II, and III

Correct answer	b

Answer (back)

7. Quick Check!

Let **`seating`** be a 2D **`String`** array that represents a seating chart in a classroom with exactly 30 seats.

Which of the following options correctly declares the **`seating`** variable and initializes the 2D array that can be used as a seating chart for the classroom?

a) `String[][] seating = new String[30][30];`
b) `String[5][6] seating = new String[5][6];`
c) `String[10][3] seating = new String[][];`
d) `String[][] seating = new String[10][3];`
e) `String[] seating = new String[30];`

Why the others are incorrect:
a would create a 2D array with 900 elements in it (30 rows of 30 each!)
b would not compile (incorrect syntax!)
c would not compile (incorrect syntax!)
e produces a 1D array!

Correct answer:	d

Solution (back)

```
// Write the totalElementsmethod below to RETURN the total
// number of elements contained in the 2D numArray,
// i.e. the row count multiplied by the column count.
public static int totalElements(int[][] numArray)
{
  int numRows = numArray.length;
  int numColumns = numArray[0].length;
  return numRows * numColumns;
}

// Write the fourCorners below to print out the elements
// in each of the four corners of strArray, one per line:
// top left, then top right, then bottom left, then bottom right
public static void fourCorners(String[][] strArray)
{
  int numRows = strArray.length;
  int numColumns = strArray[0].length;

  // print top left
  System.out.println(strArray[0][0]);

  // print top right
  System.out.println(strArray[0][numColumns - 1]);

  // print bottom left
  System.out.println(strArray[numRows - 1][0]);

  // print bottom right
  System.out.println(strArray[numRows - 1][numColumns - 1]);
}
```

Answer (back)

8. Quick Check!

What would be the result of trying to compile and execute the following code?

```
int[] row0nums = {3, 8, 9};
int[] row1nums = {1, 15, 4};
int[] row2nums = {6, 11, 5};
int[] row3nums = {4, 7, 21};

int[][] randNums = new int[3][4];
randNums[0] = row0nums;
randNums[1] = row1nums;
randNums[2] = row2nums;
randNums[3] = row3nums;
```

A. It would compile and execute, and result in the following 2D array stored in `randNums`:

```
3     8     9
1     15     4
6     11     5
4     7     21
```

B. It would compile and execute, and result in the following 2D array stored in `randNums`:

```
3     1     6     4
8     15     11     7
9     4     5     21
```

C. It would result in a compiler error and wouldn't execute.

D. It would execute, but result in an `ArrayIndexOutOfBoundsException`.

Correct answer:	**D**
	
	`new int[3][4]` creates a 2D array with **3 rows and 4 columns**, and `randNums[3] = row3nums` attempts to assign the 1D array stored in `row3nums` to the **FOURTH ROW** of `randNums` (index 3 is actually row 4).
	
	The compiler will **not** catch this error ahead of time and will compile fine (if you don't believe it, try it!). But when you execute it, it will result in the out of bounds error since `randNums` only has 3 rows!
	
	It should be noted that these three lines of code **are** valid/safe:
	`randNums[0] = row0nums;`
	`randNums[1] = row1nums;`
	`randNums[2] = row2nums;`