Your Tasks

-  Declare a two-dimensional array
-  Initialize a two-dimensional array
-  Determine the number of rows and columns in a two-dimensional array
-  Declare an initialize a “Ragged” array
-  Indicate the default initialization value for different types of arrays
-  Apply the array class to two-dimensional arrays
-  Practice with two-dimensional arrays
-  Complete Assignment 22 and have Ms. Pluska mark it complete

## Declare a two-dimensional array

Consider the following array of numbers (3 rows, 2 columns):

```
22          23
24          25
26          27
```

Let’s declare our array as follows:

```
int a[ ][ ] = new int[3][2]
```

Notice that to declare a two-dimensional array we use the same notation as before but include an extra set of brackets. The number between the first set, “3”, represents the rows. The number between the second set, “2”, represents the columns. This is the same conversion as in mathematics.

To help remember this, think “RC”, as in “RC Cola”.

## Initialize a two-dimensional array

Now let’s initialize the a array declared above (i.e., store values in the various positions). This can be done in three ways:

**The first way:**

```
int a[ ][ ] = new int[3][2]; //declaration
a[0][0] = 22;  //initializes position 0, 0 to 22
a[0][1] = 23;  //initializes position 0, 1 to 23, and so on
a[1][0] = 24;
a[1][1] = 25;
a[2][0] = 26;
a[2][1] = 27;
```

**The second way:**

```
int a[ ][ ] = { {22, 23}, {24, 25}, {26, 27} }; //notice that declaration and
                                                //and initialization must both
                                                //take place on the same line
```

**The third way:**

```
int a[ ][ ] = new int[ ][ ] { {22, 23}, {24, 25}, {26, 27} }; //notice that declaration and
                                                              //and initialization must both
```

```
                                                    //take place on the same line
```

Determine the number of rows and columns in a two-dimensional array

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

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

```
a.length; //returns a value of 3 for the number of rows
```

The number of columns in each row can be determined as follows,

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

Declare and initialize a “Ragged” array

The previous discussion seems redundant, since all the rows have 2 columns. What if each row had a different number of columns? Such an array is called a “Ragged” array.

Suppose, for example, we wanted the following array structure,

```
X    X    X    X
X    X
X    X    X
```

Below is the code that would declare such an array:

```
int a[ ][ ] = new int[3][ ]; //array has 3 rows, but unspecified number of columns
a[0] = new int[4]; //row 0 has 4 columns
a[1] = new int[2]; //row 1 has 2 columns
a[2] = new int[3]; //row 2 has 3 columns
```

Incidentally, the first line of code above ( int a[ ][ ] = new int[3][ ]; ) could be equivalently replaced with the following; however, the former is preferred:

```
int a[ ][ ] = new int[3][0]; //3 rows, unspecified number of columns
```

While on the subject of working with a single row of a two-dimensional array, consider an array a of three rows declared as was done above. How would we pass a single row of this array to a method called myMethod in which each element would be initialized?

```
a[2] = new int[3]; //Row 2 has 3 columns.  Before passing a[2] below,
                   //we must have specified the number of columns

myMethod(a[2]); //Call the method and pass the row with index 2

…

public void myMethod(int[ ] x) //notice how we receive the row
{
        x[0] = 36; //Initialize 2, 0 to 36
        x[1] = 101; //Initialize 2, 1 to 101
        x[2] = -45; //Initialize 2, 2 to -45
    }
```

Indicate the default initialization value for different types of arrays

As with all one-dimensional numeric arrays, all elements of two-dimensional arrays are also automatically initialized to 0 by default. For example:

```
    int abc[ ][ ] = new int[20][30];
    System.out.println(abc[5][22]); //prints 0
```

The default value for boolean arrays is "false". For example:

```
    boolean abc[ ][ ] = new boolean[20][30];
    System.out.println(abc[5][22]); //prints false
```

The default value for reference arrays (i.e., String, objects) is "null". For example,

```
    String abc[ ][ ] = new int[20][30];
    System.out.println(abc[5][22]); //prints null
```

Apply the array class to two-dimensional arrays

In a previous lesson on one-dimensional arrays, we used several methods from the Array class. Below are their equivalents as used with two-dimensional arrays,

```
    int a[ ][ ] = { {3, 9, 2, 1},
                    {5, 7, 6, 0} };

    int b[ ][ ] = { {0, 2, 8, 4},
                    {3, 9, 2, 1} };

    System.out.println(Arrays.equals(a, b)); //always false… won't compare entire
                                             //two-dimensional arrays

    System.out.println(Arrays.equals(a[0], b[1])); //true, compares row 0 of a to row 1 of b

    Arrays.sort(a); //illegal (run-time exception), cannot sort entire two-dimensional array
    Arryas.sort(a[0]); //sorts the 0 row of the a array.  Row 0 is now {1, 2, 3, 9}

    System.out.println(Arryas.binarySearch(a[0], 9)); //returns 3, the index of the 9 in row 0
                                                      //this row must have been sorted first

    Arrays.fill(a, 22); //illegal, cannot fill an entire two-dimensional array
    Arryas.fill(a[1], 22); //fills this row with 22 in each position
```

Practice with two-dimensional arrays

Consider the following a matrix for problems 1-11:

| | | | |
|---|---|---|---|
| 5 | -16 | 9 | 21 |
| 22 | 19 | -101 | 36 |
| 18 | 17 | 64 | -2 |

1. Write a single line of code that will create the above integer array. Call the array a.

2. Write a line of code that will printout the array element occupied by -101

3. The above table can be described as
   (a) an Array (b) a Matrix
   (c) numbers that could represented as subscripted variables
   (d) a, b, and c (e) none of these

4. Write a line of code that will print the number of rows in matrix a

5. Write a line of code that will print number of columns (in matrix a) in the row given by index 2.

6. What is printed by System.out.println(a[1][3]); ?

7. What will the following output:

```
for(int row = 0; row < a.length; row++)
{
        for(int col = 0; col < a[row].length; col++)
        {
                System.out.println(a[row][col] + "\t");
        }
        System.out.println("");
}
```

8. What is printed by the following:

```
Arrays.sort(a[0]);
System.out.println(Arrays.binarySearch(a[0], 5));
```

9. What is printed by the following:

```
Arrays.sort(a[0]);
System.out.println(Arrays.binarySearch(a[0], 0));
```

10. Show what the matrix a would look like after following code executes:

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

11. Show what the matrix a would look like after the following code executes:
    Arrays.fill(a[2], -156);

12. Must all two-dimensional arrays have the same number of columns in each row?

In the remaining problems, some of the code might not compile or will give a run-time exception. If that is the case then state, "Won't compile" or "Run-time exception".

13. What is printed by the following?

```
double d[ ][ ] = new double[8][25];
```

```
System.out.println(d[4][2]);
```

14. If x and y both represent two-dimensional int arrays and have identically the same elements, what does the following print?

    ```
    System.out.println( Arrays.equals(x, y));
    ```

15. Is it possible to sort z (a two-dimensional array) with Arrays.sort(z);?

16. Is it possible to use on of the sort methods of the Arrays class to sort a single row (index 3) of the two-dimensional matrix g? If so, show the code that will accomplish this.

## Assignment 22

Create a new project in Netbeans called GridMaker

Locate the the GridMaker activity guide

Complete challenges 1 thru 5