2023-04-24

Upcoming Schedule

Monday	Wednesday	Friday
04/24/2023 (90) • Review: Unit 8 **• AP CS Question 4: 2D Array**	**04/26/2023 (90)** **• AP CS Multiple Choice Game**	**04/28/2023 (45)** • Review: Unit 7, Unit 10 • Algorithms: Iterative/recursive binary search, selection sort, insertion sort, merge sort
05/01/2023 **• FINAL**	**05/03/2023** **• AP EXAM**	

Unit 8
AP CS FRQ 4
(2D Array)

8.1
Two-Dimensional Arrays

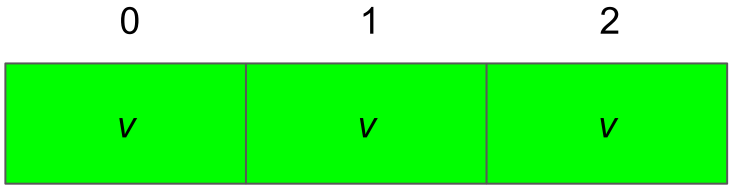
Arrays

Arrays are a zero-based indexed sequence of values of the same type. Any Java type! (Well, not void. But all other primitive or reference types.)

```
type[] name;
```

Examples:

```
boolean[] answers;
String[] questions;
int[] scores;
Student[] students;
```



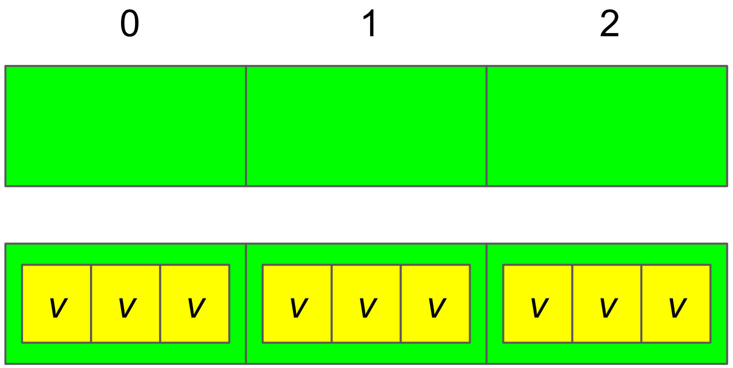
Array of Arrays

Arrays are a type, too. So, an array can be declared with arrays as its values. The result? Two-dimensional arrays!

```
type[][] name;
```

Examples:

```
boolean[][] theaterSeats;
String[][] seatingChart;
int[][] bingoCard;
Apt[][] building;
```



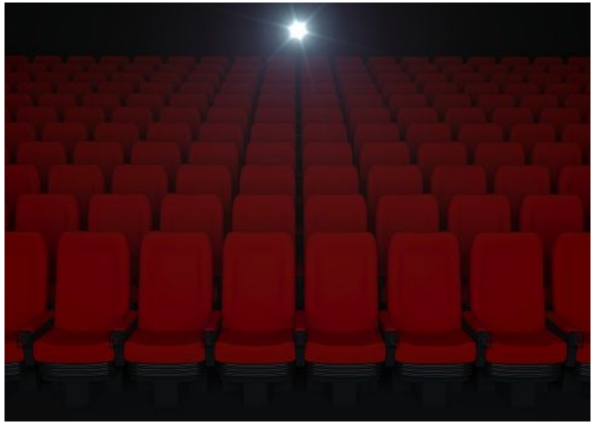
2D Arrays

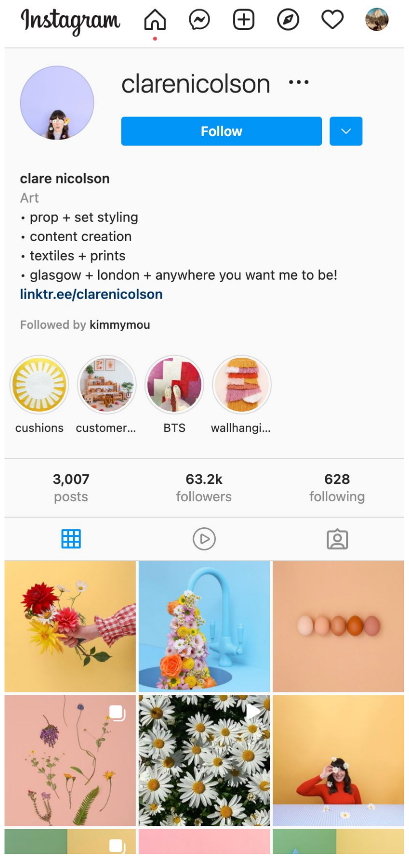
In Java, a two-dimensional array is really a one-dimensional array of one-dimensional arrays.

Conceptually, though, such arrays are used to model two-dimensional concepts, and we think of it as a single table with rows and columns.

At the Java level, it's actually an outer array where each element is a reference to a nested inner array.

What are 2D arrays good for?





Instagram

clarenicolson ···

Follow

clare nicolson
Art
• prop + set styling
• content creation
• textiles + prints
• glasgow + london + anywhere you want me to be!
linktr.ee/clarenicolson

Followed by kimmymou

cushions customer... BTS wallhangi...

3,007 posts | 63.2k followers | 628 following

BINGO
BINGO
BINGO

Here's the conventional 2D array approach to the ASCII Art Canvas.

Use rows and cols directly to access the underlying data.

Since row is the outer dimension, this is called row-major order.

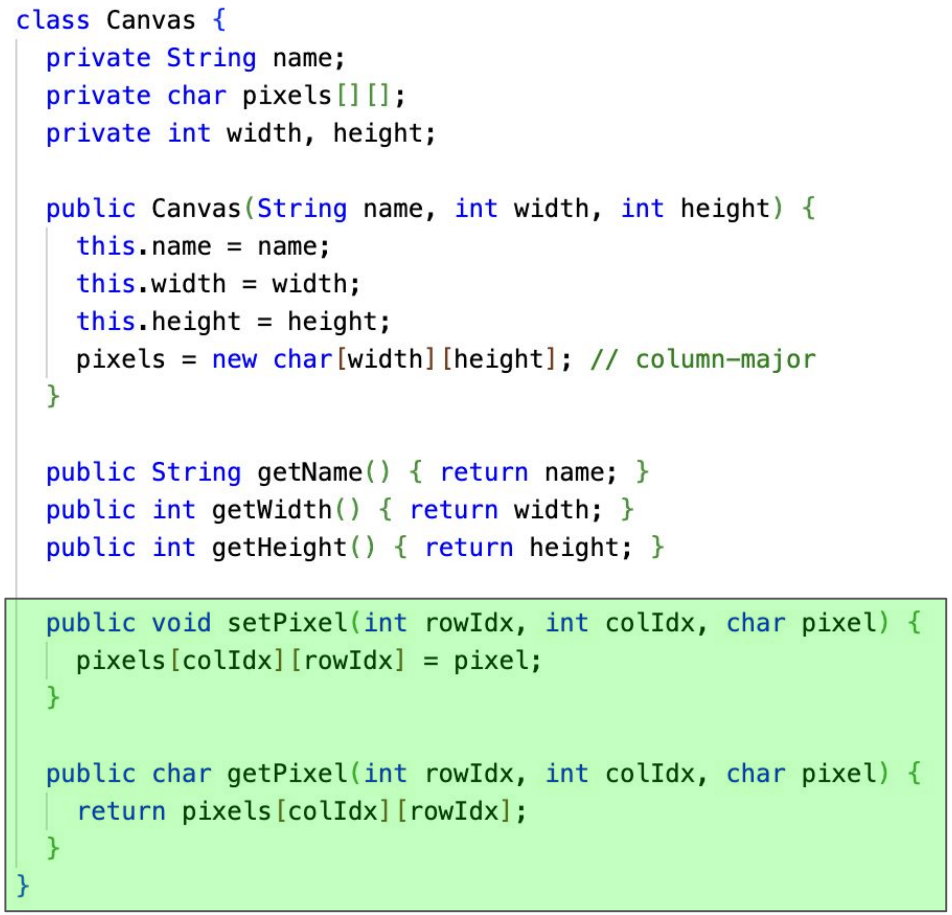
```java
class Canvas {
  private String name;
  private char pixels[][];
  private int width, height;

  public Canvas(String name, int width, int height) {
    this.name = name;
    this.width = width;
    this.height = height;
    pixels = new char[height][width];
  }

  public String getName() { return name; }
  public int getWidth() { return width; }
  public int getHeight() { return height; }

  public void setPixel(int rowIdx, int colIdx, char pixel) {
    pixels[rowIdx][colIdx] = pixel;
  }

  public Character getPixel(int rowIdx, int colIdx) {
    return pixels[rowIdx][colIdx];
  }
}
```

It is also possible to implement the ASCII Art Canvas in **column-major order.**

Here, the outer array represents the columns, and the inner arrays are the rows!

Row-major is more common, and on the AP exam, **assume row-major unless it says otherwise.**

IRL whether row-major or column-major is used depends on what the array is being used for, as well as the programmer's mental model.



```java
class Canvas {
  private String name;
  private char pixels[][];
  private int width, height;

  public Canvas(String name, int width, int height) {
    this.name = name;
    this.width = width;
    this.height = height;
    pixels = new char[width][height]; // column-major
  }

  public String getName() { return name; }
  public int getWidth() { return width; }
  public int getHeight() { return height; }

  public void setPixel(int rowIdx, int colIdx, char pixel) {
    pixels[colIdx][rowIdx] = pixel;
  }

  public char getPixel(int rowIdx, int colIdx, char pixel) {
    return pixels[colIdx][rowIdx];
  }
}
```

Array types of array types

Every type T in Java has a related array type T[]

Examples: int and int[], String and String[]

But... this is ALSO true for array types themselves!

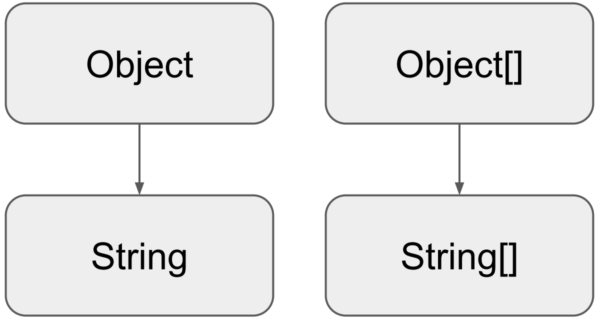
T[] has a related array type T[][]

T[][] is an array of T[]

```
type[][] name;
```

- Examples

```
boolean[][] theaterSeats; <- array of boolean[]
String[][] seatingChart; <- array of String[]
int[][] bingoCard; <- array of int[]
Apt[][] building; <- array of Apt[]
```



Multi-Dimensional Arrays

You don't have to stop at two dimensions...

The type `T[][][]` is an array of `T[][]`

Example: A 3D printer prints "voxels", three-dimensional pixels
Software for a 3D printer might store a model in memory in a 3-dimensional array.

```
Voxel[][][] voxels;
```

A Minecraft-like game might store an animation as frames of 3D voxels, but over time. So, a 4-dimensional array:

```
Voxel[][][][] voxelAnimationFrames;
```

Two-dimensional arrays are the most common multi-dimensional arrays, though. (And it's the most dimensions you'll see on the AP Exam.)

Array of Arrays - Definition

```
boolean[][] theaterSeats = new boolean[numRows][numSeatsPerRow];
String[][] seatingChart = new String[numRows][numSeatsPerRow];
int[][] bingoCard = new int[5][5];
Apt[][] building = new Apt[numFloors][numAptsPerFloor];
```

**** NOTE 1 ****
Two-Dimensional Arrays can have non-equal dimensions!

```
boolean[][] theaterSeats = new boolean[75][25];
String[][] seatingChart = new String[5][10];
int[][] multiplicationTable = new int[100][500];
Apt[][] building = new Apt[5][10];
```

Array of Arrays - Definition

```
boolean[][] theaterSeats = new boolean[numRows][numSeatsPerRow];
String[][] seatingChart = new String[numRows][numSeatsPerRow];
int[][] bingoCard = new int[5][5];
Apt[][] building = new Apt[numFloors][numAptsPerFloor];
```

**** NOTE 2 ****
Just like Arrays - Two-Dimensional Arrays initialize their values to "reasonable" defaults

- **`0 for numeric types`**
- **`null for Object types`**
- **`false for boolean types`**

Array of Arrays - Definition

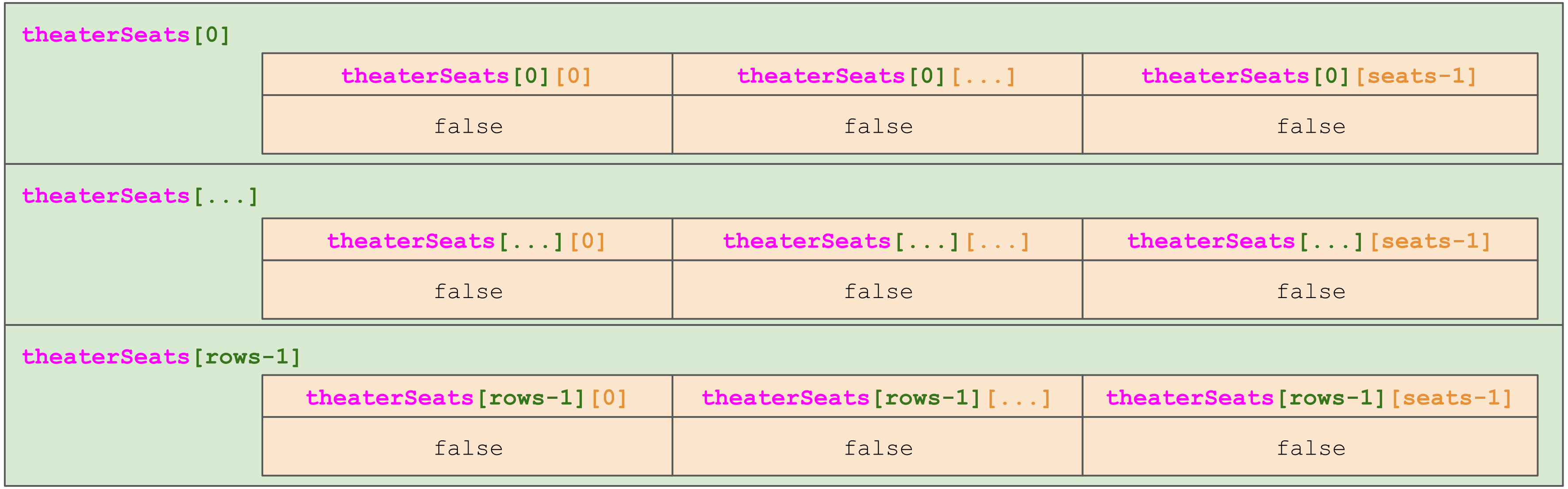
```
int[][] bingoCard = new int[5][5];

                      Alternatively...

int[][] bingoCard = new int[5][]; // omit internal array size
bingoCard[0] = new int[5];
bingoCard[1] = new int[5];
bingoCard[2] = new int[5];
bingoCard[3] = new int[5];
bingoCard[4] = new int[5];
```

Array of Arrays - Definition

```
boolean[][] theaterSeats = new boolean[numRows][numSeatsPerRow];

                         Alternatively...

boolean[][] theaterSeats = new boolean[numRows][];
for (int rowIdx = 0 ; rowIdx < numRows ; rowIdx++ {
  theaterSeats[rowIdx] = new boolean[numSeatsPerRow];
}
```

Arrays of Arrays - Example

```
boolean[][] theaterSeats = new boolean[rows][seats];
```



```
Write: theaterSeats[0][0] = true;
Read : System.out.println(theaterSeats[rows-1][seats-1]);
```

Arrays of Arrays - Initializer Lists

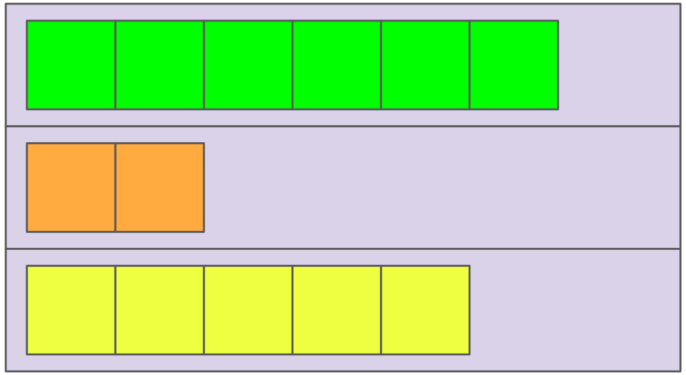
- You can initialize the values of a Two-Dimensional Array when you create it (and the sizes will be automatically calculated)

```
int[][] ticketInfo = { {25,20,25}, {25,20,25} };
ticketInfo.length     => 2
ticketInfo[0].length) => 3
ticketInfo[1].length) => 3


boolean[][] jaggedTable = { {false, true, true}, {false}, {true} };
jaggedTable.length     => 3
jaggedTable[0].length) => 3
jaggedTable[1].length) => 1
jaggedTable[2].length) => 1
```

Arrays of Arrays - Jagged Arrays

- On the exam all inner arrays will have the same length - **However** it is possible in Java to have inner arrays of different lengths. These are called Jagged (or Ragged) Arrays. **You can assume 2D arrays on the exam are NOT jagged.**

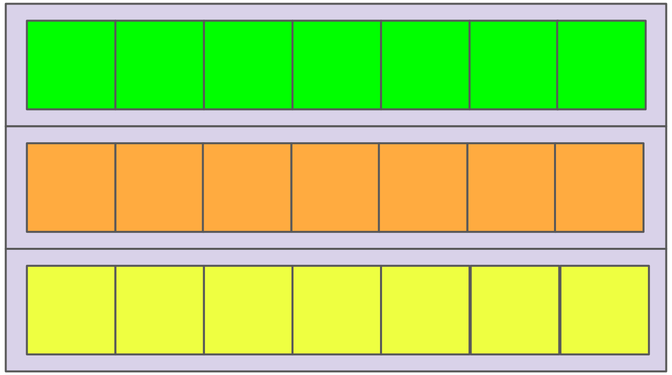
```
int jaggedTable[][] = new int[3][]; // omit the inner array size

jaggedTable[0] = new int[6];
jaggedTable[1] = new int[2];
jaggedTable[2] = new int[5];

jaggedTable.length       => 3
jaggedTable[0].length    => 6
jaggedTable[1].length    => 2
jaggedTable[2].length    => 5
```



2D array dimensions

On the AP exam, all inner arrays will have the same length.
So, you can use the length of the first row, if any, to determine the number of columns in a 2D array.
You may safely assume the rest of the rows will have the same number of columns.

```
int grid[][] = new int[3][7];
...

int numRows = grid.length;
> 3

int numCols = grid[0].length;
> 7
```



8.2
Traversing
Two-Dimensional Arrays

Review: Traversing Arrays with `for` loops

- Remember that the range of valid Array indexes (for non-empty Arrays) is `0` to `Array.length - 1`

```
int scores[] = {95, 100, 91, 85 };
for (int idx = 0; idx < scores.length; idx++)
{
     System.out.println(scores[ idx]);
}
```

✓

```
int scores[] = {95, 100, 91, 85 };
for (int idx = 1; idx <= scores.length;
idx++) {
     System.out.println(scores[ idx]);
}
```

✗

Review: Traversing Arrays with `for` loops

- Remember that the range of valid Array indexes (for non-empty Arrays) is `0` to `Array.length - 1`

```
int scores[] = {95, 100, 91, 85 };
for (int idx = 0; idx < scores.length; idx++)
{
    System.out.println(scores[ idx]);
}
```

✓

```
int scores[] = {95, 100, 91, 85 };
for (int idx = 1; idx <= scores.length;
idx++) {
    System.out.println(scores[ idx]);
}
```

✗

***Note:** Passing an out of range index will cause a `ArrayIndexOutOfBoundsException`!*

Review: Traversing Arrays with `for` loops

- Remember that the range of valid Array indexes (for non-empty Arrays) is `0` to `Array.length - 1`

```
int scores[] = {95, 100, 91, 85 };
for (int idx = 0; idx < scores.length; idx++)
{
    System.out.println(scores[ idx]);
}
```

```
int scores[] = {95, 100, 91, 85 };
for (int idx = 1; idx <= scores.length;
idx++) {
    System.out.println(scores[ idx]);
}
```

This loop also skips the first element in the Array!

***Note:** Passing an out of range index will cause a `ArrayIndexOutOfBoundsException`!*

Review: Traversing Arrays with `for` loops

- You can use a `for` loop to traverse an Array from back to front!

```
int scores[] = {95, 100, 91, 85 };
for (int idx = scores.length - 1; idx >= 0; idx--) {
    System.out.println(scores[ idx]);
}
```

- ...or to traverse any arbitrary range of elements

```
int scores[] = {95, 100, 91, 85 };
for (int idx = 1; idx <= 2; idx++) {
    System.out.println(scores[ idx]);
}
```

Traversing Two-Dimensional Arrays with `for` loops

- Traversing Two-Dimensional Arrays is very similar

```
int scores[ ][ ] = { {10,20,30},{40,50,60} };
for (int idx = 0; idx < scores.length; idx++) {
  for (int jdx = 0; jdx < scores[idx].length; jdx++) {
    System.out.println(scores[idx][jdx]);
  }
}
```

Typically start at index = 0
& don't exceed `length-1`

Note: *Passing an out of range index will cause a `ArrayIndexOutOfBoundsException`!*

Review: Traversing Arrays with `for-each` loops

```
for (type arrayItemVariable : arrayVariable) {
    arrayItemVariable resolves to arrayVariable[...]
}
```

```
String[] colors = {"red", "orange", "purple"};
```

```
System.out.println("begin");
for(String color: colors){
  System.out.println(" " + color);
}
System.out.println("end");
```

Review: Traversing Arrays with `for-each` loops

- The type of the `for-each` variable MUST match the type of the values stored in the Array

```
String colors[] = {"red", "orange", "purple"};

for(int color: colors){
  System.out.println(" " + color);
}
```

Note: *color must be of type String since colors is an Array that contains Strings*

Traversing Two-Dimensional Arrays with for-each loops

- Remember during the introduction of Two-Dimensional Arrays - We said: "***Arrays are a type*** *- Which means you can easily create an* ***Array that contains Arrays*** *- often called Two-Dimensional Arrays*"
- That means we can use `for-each` to traverse a Two-Dimensional Array almost exactly like a One-Dimensional Array (**we just have to be careful how we declare the types**)

Reminder: `for-each` loops can be super-useful; but you are unable to make use of an index or change the underlying Array while looping

Traversing Two-Dimensional Arrays with for-each loops

Given this Two-Dimensional Array

```
int scores[][] = {{10,20,30},{40,50,60}};
```

And this general description of for-each

```
for (type arrayItemVariable : arrayVariable) {
    arrayItemVariable resolves to Array[...]
}
```

Q: What is the type of the outer array in scores?

scores[]   -> an array of int[]

Q: What is the type of the inner array in scores?

scores[][]   -> an array of int

Traversing Two-Dimensional Arrays with for-each loops

```
int scores[][] = {{10,20,30},{40,50,60}};
```

```
scores[]   -> an array of int[]
scores[][] -> an array of int
```

So we can use `for-each` to traverse the Two-Dimensional Array like this

```
for (int[] outer : scores) {
  for (int inner : outer) {
    System.out.println(inner);
  }
}
```

Note: The inner array is a One-Dimensional Array - So we use the same `for-each` that we used previously for One-Dimensional Arrays

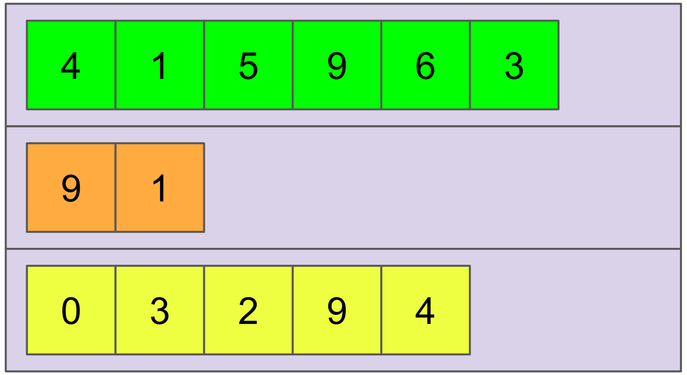
```
for (type arrayItemVariable : arrayVariable) {
  arrayItemVariable resolves to arrayVariable[...]
}
```

No indices available for use!

Traversal of Jagged Arrays

- Traversal of Two-Dimensional Jagged Arrays works the same!

```
int jaggedTable[][] = new int[3][];

jaggedTable[0] = new int[]{4,1,5,9,6,3};
jaggedTable[1] = new int[]{9,1};
jaggedTable[2] = new int[]{0,3,2,9,4};

for (int idx = 0; idx < jaggedTable.length; idx++) {
  for (int jdx = 0; jdx < jaggedTable[idx].length; jdx++) {
    System.out.print(jaggedTable[idx][jdx] + " ");
  }
  System.out.println();
}
```



Example: Basketball Scores (Row-Major)

```
int NUM_PLAYERS = 5, NUM_GAMES = 3;
int scores[][] = new int[NUM_PLAYERS][NUM_GAMES];
```

	Game 1	Game 2	Game 3
Player 1	14	19	22
Player 2	24	13	5
Player 3	5	26	31
Player 4	0	18	40
Player 5	15	9	46

`scores[][]`

Example: Basketball Scores (Row-Major)

```
int NUM_PLAYERS = 5, NUM_GAMES = 3;
int scores[][] = new int[NUM_PLAYERS][NUM_GAMES];
```

	Game 1	Game 2	Game 3
Player 1	14	19	22
Player 2	24	13	5
Player 3	5	26	31
Player 4	0	18	40
Player 5	15	9	46

```
scores[3][0], scores[3][1], scores[3][2]
```

Example: Basketball Scores (Row-Major)

```
int NUM_PLAYERS = 5, NUM_GAMES = 3;
int scores[][] = new int[NUM_PLAYERS][NUM_GAMES];
```

	Game 1	Game 2	Game 3
Player 1	14	19	22
Player 2	24	13	5
Player 3	5	26	31
Player 4	0	18	40
Player 5	15	9	46

```
scores[0][1], scores[1][1], scores[2][1], scores[3][1], scores[4][1]
```

Example: Basketball Scores (Row-Major)

```
int NUM_PLAYERS = 5, NUM_GAMES = 3;
int scores[][] = new int[NUM_PLAYERS][NUM_GAMES];
```

	Game 1	Game 2	Game 3
Player 1	14	19	22
Player 2	24	13	5
Player 3	5	26	31
Player 4	0	18	40
Player 5	15	9	46

Q1: *How would you determine the total points scored by all 5 players in all 3 games?*
Q2: *How would you determine which player had the highest number of points in Game 3?*
Q3: *How would you determine the average points scored by Player 3 in all 3 games?*

AP CS FRQ 4

(25 minutes)

2022 AP Computer Science A - Free-Response Questions

Complete (4.A) and (4.B)

AP CS FRQ 4 - Review

(15 minutes)

Sample Responses and Scoring Commentary - FRQ-4