Arrays

A Collection of Data for Simple Access

Alwin Tareen

The Array Data Structure

What is a data structure?

- A data structure is a collection of data elements that are combined together under one name.
- Data structures are used for storing and organizing data so that it can be used efficiently.

What is an array?

- An array is a data structure that stores a collection of individual values that are of the same data type.
- For example, an array can contain `ints`, `doubles`, etc.

Declaring and Defining an Array

Declaring an array

- To declare an array variable, you must include square brackets `[]` between the data type and the variable name.

```
int[] tests;
```

Defining an array

- To create the array itself, we must specify its data type, and the quantity of elements that it can contain.

```
tests = new int[10];
```

- This statement creates an array that will store 10 values of type `int`.

Declaring and Defining an Array

Using a single statement

- The declaring and initializing of an array can occur in a single statement. Arrays are usually constructed in this manner.

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

- Note that in the above code statement:
  - An array variable named `nums` is declared.
  - An array object of size 5 is defined, of data type `int`.

Accessing the Elements of an Array

- To access the individual elements of an array, you must use the array variable's name followed by the number of the element, enclosed in square brackets.
- This number is referred to as the **index**.
- The following code places elements into indexes `0` and `1`.

```
nums[0] = 5;
nums[1] = 38;
```

- The following code reads from indexes `0` and `1`.

```
System.out.println(nums[0]);
System.out.println(nums[1]);
```

Initializer Lists

A programmer's shortcut

- An initializer list allows you to declare an array, and assign values to each of its elements, in a single statement.

```
double[] scores = {93.7, 86.2, 91.5, 98.3};
```

- This creates an array of size 4, and assigns each of the elements the following values:

element	93.7	86.2	91.5	98.3
index	0	1	2	3

Index Out of Bounds Exception

- If you attempt to index an element of an array that does not exist, Java will return an IndexOutOfBounds exception.

```
int[] points = new int[3];
points[5] = 99; // IndexOutOfBounds exception
```

- This usually occurs when you attempt to access an element that is located beyond the size of the array.
- This is a very common error in Java programming, and you must avoid it as much as possible.

Looping through an Array(Traversing)

- Suppose we want to create an integer array of size 20 called `nums`, and fill each element of the array with a random number in the range of 0 to 99.
- It would be inefficient to assign the random values to each element of the array in the following manner:

```
nums[0] = (int) (Math.random() * 100);
```

- A much better approach would be to use a `for` loop to iterate through all the elements of the array.

```
for (int i = 0; i < 20; i++)
{
    nums[i] = (int) (Math.random() * 100);
}
```

Looping through an Array(Traversing)

- We can also use a `for` loop to view the contents of an array:

```
for (int i = 0; i < 20; i++)
{
    System.out.println(nums[i]);
}
```

- A concise way of displaying arrays is to use the `Arrays` class. It is contained in the library `java.util`.

```
import java.util.*;
String result = Arrays.toString(nums);
System.out.println(result);
```

Determining the Quantity of Elements

Using the data member: `length`

- Java allows you to determine the quantity of elements in an array using `length`, a data member of the array object.
- In the following code statement, `samples.length` is 50.

```
double[] samples = new double[50];
```

Determining the Average

- The following Java program calculates the average from a group of random numbers. Note the use of `length`.

```
double total = 0.0;
double[] samples = new double[50];

for (int i = 0; i < samples.length; i++)
{
    samples[i] = Math.random() * 100;
    total += samples[i];
}

double average = total/samples.length;
System.out.println("Average = " + average);
```

Appropriate Use of `length`/`length()`

`String` → `length()`

- The `String` class has a **method** named `length()` that returns the quantity of characters in that `String`.

Array → `length`

- On the other hand, an array has a **variable** named `length` that contains the quantity of elements in that array.

`String` method

```
int qty = word.length();
```

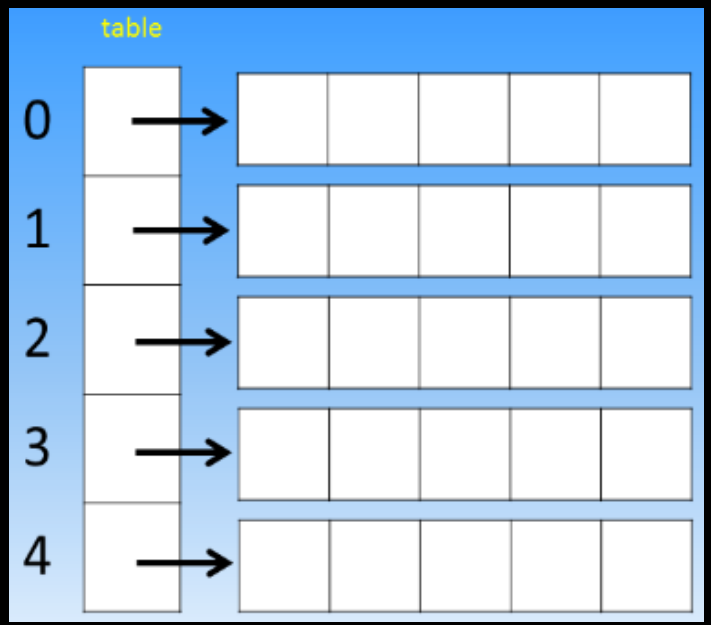
Array variable

```
int qty = nums.length;
```

Two-Dimensional Arrays

The concept of a 2-D array

- Suppose I declare an array of size `5`, named `table`.
- Instead of filling the array with integers or `Strings`, I place an array in each cell.
- In this case, I have created an array of arrays, also known as a two-dimensional array.



Two-Dimensional Arrays

Initializing a 2-D array

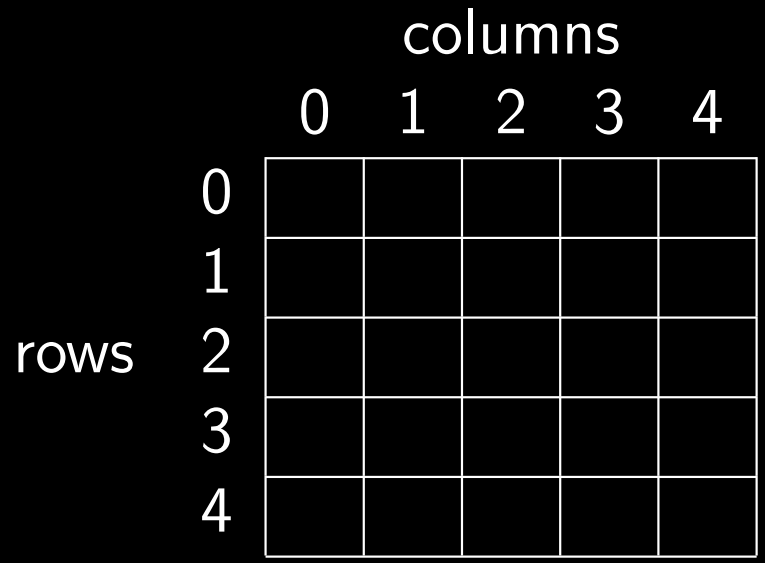
- In Java, two-dimensional arrays are defined using the following notation:

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

- Note the double square brackets. This indicates the dimension, which in this case is 2.
- Symbolically, it is easier to think of a 2-dimensional array as a grid, with rows and columns.
  - The first 5 declares the number of rows.
  - The second 5 declares the number of columns.

Two-Dimensional Arrays

A grid representation of a 2-D array



- Location [1] [3] would be 2 rows down and 4 columns across. Remember that rows and columns start with 0.
- Similarly, location [4] [2] would be 5 rows down and 3 rows across.

Two-Dimensional Arrays

Determining the number of rows and columns

- The number of rows in a 2-D array is found in:
  `table.length`
- The number of columns in a 2-D array is found in:
  `table[0].length`

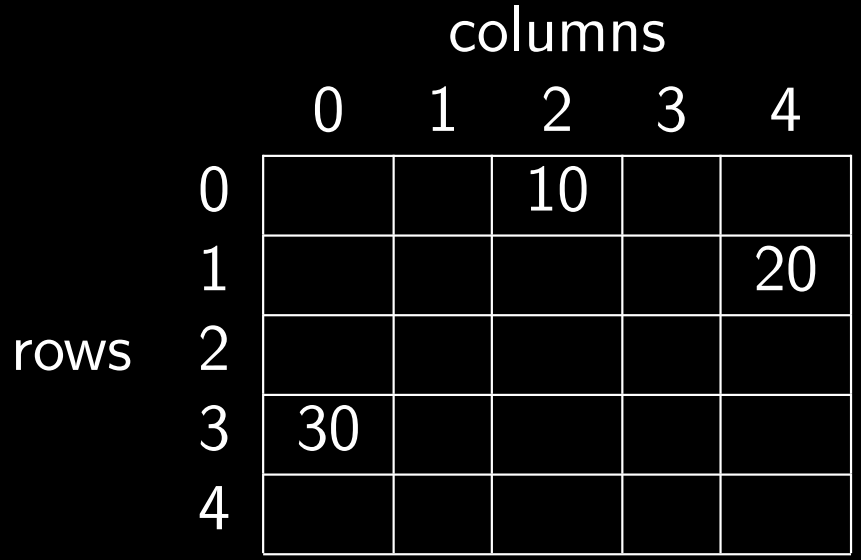
Additional details

- All the elements of a 2-dimensional array must be of the same data type(`int`, `double`, etc.).
- The first index of an array initialization statement always represents the row, and the second index represents the column.
- All 2-dimensional arrays in the AP exam are guaranteed to be square or rectangular(no ragged edge arrays).

Two-Dimensional Arrays

- The following code indicates how a 2-dimensional array assigns values to its elements.

```
int[][] matrix = new int[5][5];
matrix[0][2] = 10;
matrix[1][4] = 20;
matrix[3][0] = 30;
```



Two-Dimensional Arrays

Initializer lists

- A 2-dimensional array can be established with an initializer list.
- The number of inner lists determines the number of rows, and the size of each inner list determines the number of columns in that particular row.

```
int[][] matrix = {{0, 1, 2, 3, 4}, // row 0
                  {10, 11, 12, 13, 14}, // row 1
                  {20, 21, 22, 23, 24}, // row 2
                  {30, 31, 32, 33, 34}}; // row 3
```

Two-Dimensional Arrays

Displaying a 2-D array

- Generally, you would use two `for` loops to display a 2-dimensional array.
- A better way is to use the `Arrays` class from the `java.util` library.

```
import java.util.*;
String display = "";

for (int row = 0; row < table.length; row++)
{
    display += Arrays.toString(table[row]) + "\n";
}
System.out.println(display);
```

Looping Through a 2-D Array(Traversing)

- The easiest way to manipulate a 2-dimensional array is to use nested `for` loops.
- The following code sums all of the numbers in the 2-dimensional array called `nums`.
- The outer loop iterates 4 times, and moves down the rows.
- Each time through the outer loop, the inner loop iterates 5 times and moves across the columns of the current row.

```
for (int row = 0; row < 4; row++)
{
    for (int col = 0; col < 5; col++)
    {
        total += nums[row][col];
    }
}
```

Looping Through a 2-D Array(Traversing)

- In the previous case, we used a limit of 4 for the number of rows, and a limit of 5 for the number of columns.
- However, there are cases in which we won't know the quantity of rows and columns in the 2-D array.
- Therefore, it is much more practical to use the `length` variable instead of literal numbers.
- Recall that `nums.length` is the number of rows, and `nums[0].length` is the number of columns.

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

Arrays: End of Notes