2D Arrays

CS 1400 2021
Cody Henrichsen

1

2D Structure

▷ Data organized into a 2D grid or matrix
▷ In Java it is really an array of arrays
▷ Two [ ] are the "keyword" since an array is hipster
▷ T/type [ ][ ] variableName = new T/type [rows][cols];
▷ Rows are always the first set of [] columns the second!

2

2D Addresses

arrayName [0][0]	arrayName [0][1]	arrayName [0][2]	arrayName [0][3]	arrayName [0][4]
arrayName [1][0]	arrayName [1][1]	arrayName [1][2]	arrayName [1][3]	arrayName [1][4]
arrayName [2][0]	arrayName [2][1]	arrayName [2][2]	arrayName [2][3]	arrayName [2][4]
arrayName [3][0]	arrayName [3][1]	arrayName [3][2]	arrayName [3][3]	arrayName [3][4]

3

2D Info

▷ Just like an array the placement of the assignment operator tells you what is happening

   ▷ Right of the arrayName and address is assigning a value to the location

   ▷ Left of the arrayName and address is retrieving a value from the array

4

Columns

01 length = Columns

5

2D Initialization

▷ The default value for the data type of the array fills each slot

   ▷ 0 for primitives except boolean (false)

   ▷ null for all Objects

▷ Can also initialize with values separated by commas surrounded with squiggles for each row in the array

6

2D Initialization Demo

```java
private void twoDimensionalDemo()
{
    int [][] grid = new int [4] [5]; // This is 20 zeroes!

    DebugDuck [][] boringArray = new DebugDuck [4][4];  // This is 16 null values

    String [][] words = {
                            {"These", "words", "are"},
                            {"in", "a", "2D"},
                            {"array", "of", "String"}
                        };
}
```

7

Iteration: Standard For

▷ Proper iteration of a 2D array can be either row major or column major
  ▷ AKA loop over rows first or columns first
▷ Regardless of style you will want to nest the loops
▷ If you need addresses you will need the standard for loop
  ▷ Replacing data AKA assigning values
▷ You can iterate over parts of the 2D array and vary the steps
  ▷ Backwards
  ▷ Half
  ▷ Every third/seventh spot…

8

Iteration – Standard For Demo

```java
for (int row = 0; row < grid.length; row++)
{
    for (int col = 0; col < grid[0].length; col++)
    {
        System.out.println("Current value is: " + grid[row][col]);
    }
}

for (int row = 0; row < grid.length; row++)
{
    for (int col = 0; col < grid[0].length; col++)
    {
        grid [row][col] = (int)(Math.random() * 1234);
    }
}
```

9

Iteration: For Each

▷ Great for output or evaluating values
▷ Can't replace values in a for-each!
▷ Objects can have mutator methods called to update state!
▷ Row major only

10

Iteration – For Each Demo

```java
for (int [] row : grid)
{
    for (int value : row)
    {
        System.out.println("Current value is: " + value);
    }
}
```

```java
for (int [] row : grid)
{
    for (int value : row)
    {
        value = 7;
    }
}
```

11

Iteration – For Each Demo 2

```java
DebugDuck [][] duckburg = new DebugDuck [5][6];

for (int row = 0; row < duckburg.length; row++)
{
    for (int col = 0; col < duckburg[0].length; col++)
    {
        duckburg[row][col] = new DebugDuck("Duck at row: " + row + ", col: " + col );
    }
}

for (DebugDuck [] row : duckburg)
{
    for(DebugDuck currentDuck : row)
    {
        int helpfulness = (int) (Math.random() * 13);
        currentDuck.setQuestionCount(helpfulness);
        view.displayMessage("My name is: "+ currentDuck.getName());
    }
}
```

12

Bounds Checking

▷ Be sure to check the correct bounds before doing any logic inside your 2D array
▷ All row indices are >= 0 and < arrayName.length
▷ All column indices are >= 0 and < arrayName[0].length
▷ Do the bounds check BEFORE any other logic or access!!!
  ▷ You want to avoid ArrayIndexOutOfBoundsException

13

Still the hipster of Java

▷ No methods belong to the type
▷ No constructor or () anywhere in sight
▷ Two public data members
  ▷ arrayName.length for the number of rows
  ▷ arrayName[0].length for the number of columns

14

Jagged/Ragged Array

▷ A non rectangular 2D array
▷ Each row can have a different amount of columns
▷ Will NEVER be used in class
▷ Traversal must NOT use arrayName[0].length since each row's column count is independent!

15

Jagged Demo

```java
private void jaggedDemo()
{
    int [][] jagged = new int [5][];
    jagged [0] = new int [5];
    jagged [1] = new int [1];
    jagged [2] = new int [3];
    jagged [3] = new int [2];
    jagged [4] = new int [5];

}
```

jagged [0] [0]	jagged [0] [1]	jagged [0] [2]	jagged [0] [3]	jagged [0] [4]
jagged [1] [0]				
jagged [2] [0]	jagged [2] [1]	jagged [2] [2]		
jagged [3] [0]	jagged [3] [1]			
jagged [4] [0]	jagged [4] [1]	jagged [4] [2]	jagged [4] [3]	jagged [4] [4]

0	0	0	0	0
0				
0	0	0		
0	0			
0	0	0	0	0

16