# CS159

#### Nathan Sprague

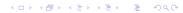
April 24, 2015

◆□ ▶ < 圖 ▶ < 圖 ▶ < 圖 ▶ < 圖 • 의 Q @</p>



#### Bernstein Slides ∧

Some content here is borrowed from those slides.



#### Review of Arrays

#### Declaration:

- int[] numbers;
- String[] words;

### Review of Arrays

#### Declaration:

- int[] numbers;
- String[] words;

#### Instantiation:

- numbers = new int[4];
- words = new String[4];

◆□▶ ◆□▶ ◆三▶ ◆三▶ 三三 のへぐ

### Review of Arrays

#### Declaration:

- int[] numbers;
- String[] words;

Instantiation:

- numbers = new int[4];
- words = new String[4];
- How many strings have been created? How many ints?

Let's draw a picture of memory...

# Multiple Arrays...

#### Does this seem OK?

int[]	donationsWeek0	=	{10,	75};
int[]	donationsWeek1	=	{15,	20, 20};
int[]	donationsWeek2	=	{100,	, 63, 10};
//				

## Multiple Arrays...



No reason we can't create an array of arrays.

▲□▶ ▲□▶ ▲□▶ ▲□▶ ▲□ ● ● ●

Declaration:

int[][] donations;

◆□▶ ◆□▶ ◆臣▶ ◆臣▶ 臣 の�?

String[][] words;

#### Declaration:

- int[][] donations;
- String[][] words;

Instantiation:

```
numbers = new int[4][3];
```

words = new String[4][3];

◆□▶ ◆□▶ ◆三▶ ◆三▶ 三三 のへぐ

#### Declaration:

- int[][] donations;
- String[][] words;

Instantiation:

```
numbers = new int[4][3];
```

- words = new String[4][3];
- How many strings have been created? How many ints?

Let's draw a picture of numbers...

#### Declaration:

- int[][] donations;
- String[][] words;

Instantiation:

```
numbers = new int[4][3];
```

- words = new String[4][3];
- How many strings have been created? How many ints?

- Let's draw a picture of numbers...
- How would this change the picture:

```
numbers[1][2] = 7;
```

#### Declaration:

- int[][] donations;
- String[][] words;

Instantiation:

```
numbers = new int[4][3];
```

- words = new String[4][3];
- How many strings have been created? How many ints?

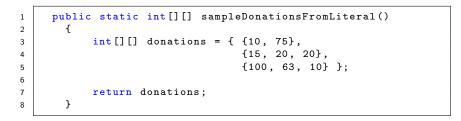
- Let's draw a picture of numbers...
- How would this change the picture:

numbers[1][2] = 7;

How would this change the picture:

```
numbers[2] = new int[5];
```

#### **MD-Array Literals**



### **MD-Array Example**

1 2

3 4

5 6

7

8

9 10

11

12

13

14 15

16

17

18 19

20 21

```
public int[][] sampleDonations()
£
    int[][] donations;
    donations = new int[3][]; // 3 rows, no columns
    donations[0] = new int[2];
    donations [0][0] = 10:
    donations [0][1] = 75:
    donations[1] = new int[3];
    donations [1][0] = 15:
    donations [1][1] = 20;
    donations [1][2] = 20;
    donations[2] = new int[3];
    donations [2][0] = 100:
    donations [2][1] = 63;
    donations [2][2] = 10;
    return donations;
}
```

#### Rectangular MD-Array

```
public int[][] sampleDonationsRectangular()
1
 2
         ſ
3
             int rows;
4
             int columns:
5
             int[][] donations;
6
7
             rows = 3:
8
             columns = 4:
9
             donations = new int[rows][columns];
10
11
             for (int row = 0: row < rows: row++)</pre>
12
             ſ
13
                  for (int col = 0; col < columns; col++)</pre>
14
                  ſ
15
                      donations[row][col] = row * col;
16
                  3
17
             }
18
19
             return donations;
20
         }
```

◆□▶ ◆□▶ ◆臣▶ ◆臣▶ 三臣 - のへで

<□ > < @ > < E > < E > E のQ @

```
1
 2
         public static int maximum(int[][] values, int rows, int columns)
 3
 4
              int maxVal = Integer.MIN_VALUE;
5
6
              for (int row = 0: row < rows: row++)</pre>
7
8
                  for (int col = 0: col < columns: col++)</pre>
9
                       if (values[row][col] > maxVal)
10
11
12
                           maxVal = values[row][col]:
13
                       3
14
                  3
15
16
              return maxVal:
17
         3
```

Only works for rectangular array. Why require the caller to provide the array sizes?

```
public static int maximum1(int[][] values)
1
        ſ
            int maxVal = Integer.MIN_VALUE;
            for (int row = 0; row < values.length; row++)</pre>
            ſ
                 for (int col = 0; col < values[row].length; col++)</pre>
                 ł
                     if (values[row][col] > maxVal)
                     Ł
                         maxVal = values[row][col];
                     }
                 3
            3
            return maxVal:
        }
```

▲□▶ ▲□▶ ▲□▶ ▲□▶ □ のQ@

1

4 5

6

7 8

9

10

11

12

13

14

15

16

```
public static int maximum1(int[][] values)
2
3
        ſ
            int maxVal = Integer.MIN_VALUE;
            for (int row = 0; row < values.length; row++)</pre>
             ſ
                 for (int col = 0; col < values[row].length; col++)</pre>
                     if (values[row][col] > maxVal)
                     ſ
                         maxVal = values[row][col];
                     }
                 3
             3
            return maxVal:
        }
```

Better! What if we are worried about unitialized arrays?

```
public static int maximum3(int[][] values)
    int maxVal = Integer.MIN_VALUE;
    if (values != null) // Make sure values is initialized.
    ſ
        for (int row = 0; row < values.length; row++)</pre>
        ſ
            if (values[row] != null) // Make sure the row is initialized.
            ſ
                 for (int col = 0; col < values[row].length; col++)</pre>
                 Ł
                     if (values[row][col] > maxVal)
                         maxVal = values[row][col];
                     3
                }
            }
        3
    3
   return maxVal;
3
```

▲ロト ▲園ト ▲ヨト ▲ヨト ニヨー のへ(で)

#### One more:

#### One more:

```
public static int maximum4(int[][] values)
1
2
             int maxVal = Integer.MIN_VALUE;
             for (int[] row : values)
             ſ
                 for (int value : row)
                 Ł
                     if (value > maxVal)
                     ſ
                         maxVal = value;
                     }
                 3
             return maxVal;
16
         }
```

I think this is less clear than the indexed approach, but it works.