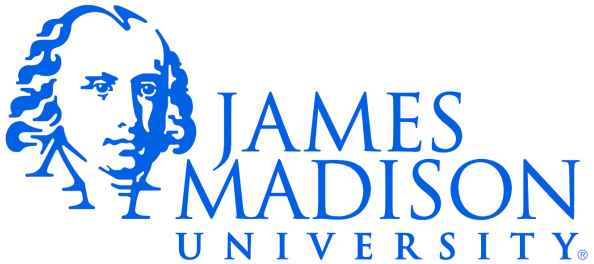
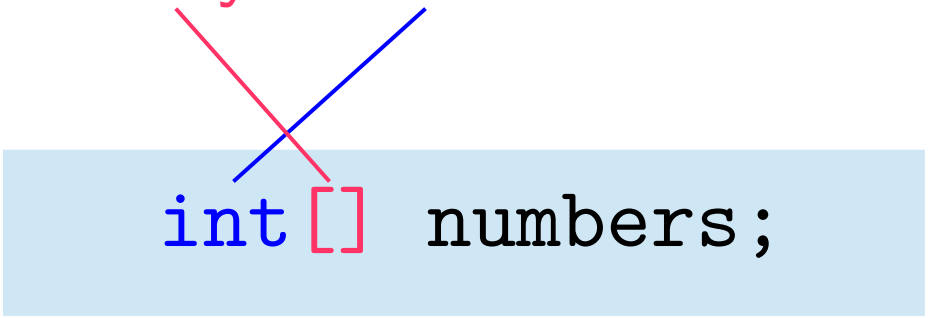
# CS159

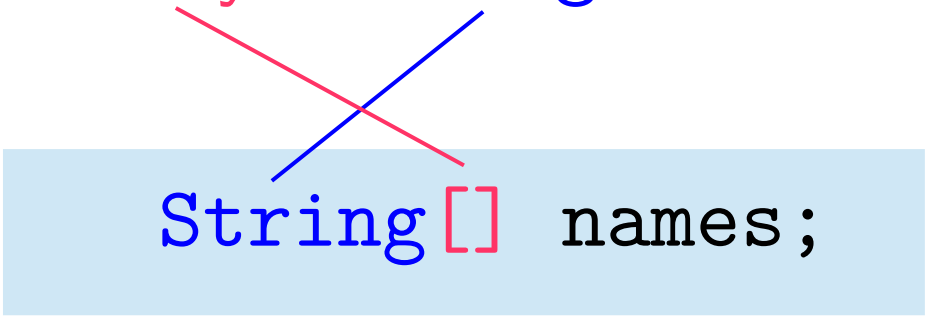


# Review: Declaring Arrays

This is an array of int entries.




This is an array of String entries.



# Review: Instantiating Arrays

```
numbers = new int[3];
names = new String[3];
```

# Review: Instantiating Arrays

- Socrative Quiz: how many ints are created by the followowing code snippet? How many Strings?

```
numbers = new int[3];
names = new String[3];
```

A) 3, 3

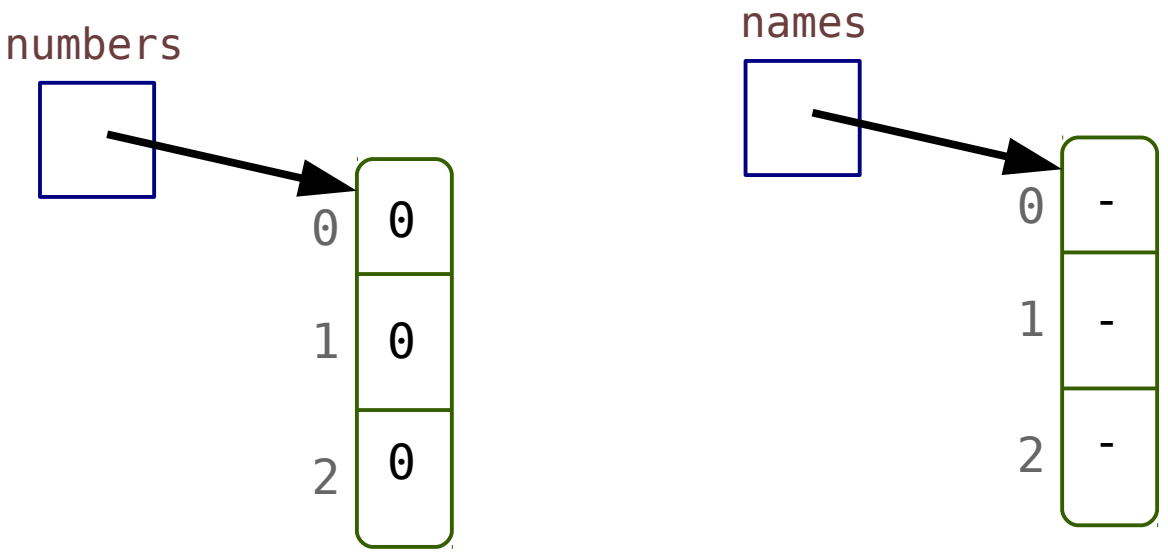
B) 2, 2

C) 3, 0

D) 0, 3

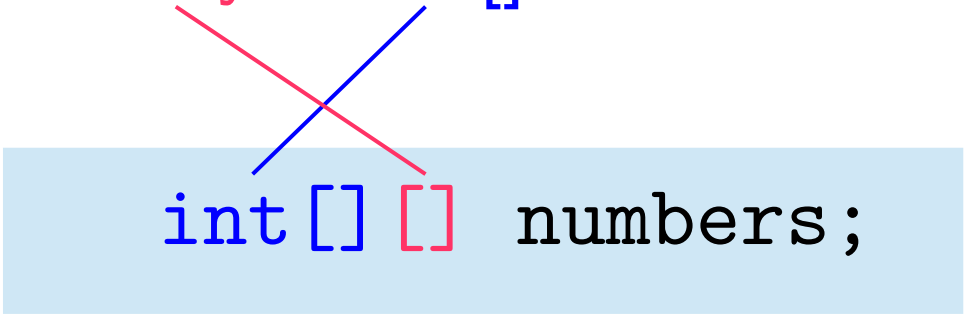
# Drawing...

```
numbers = new int[3];
names = new String[3];
```

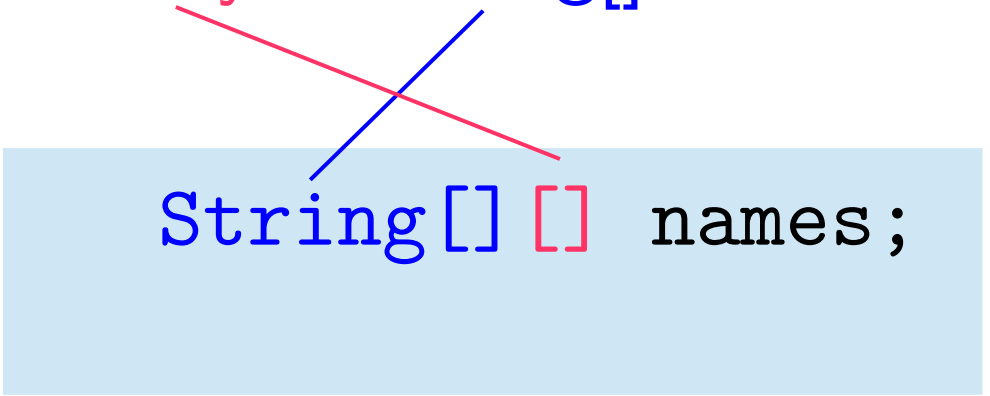


# Declaring 2D Arrays

This is an array of int[] entries.



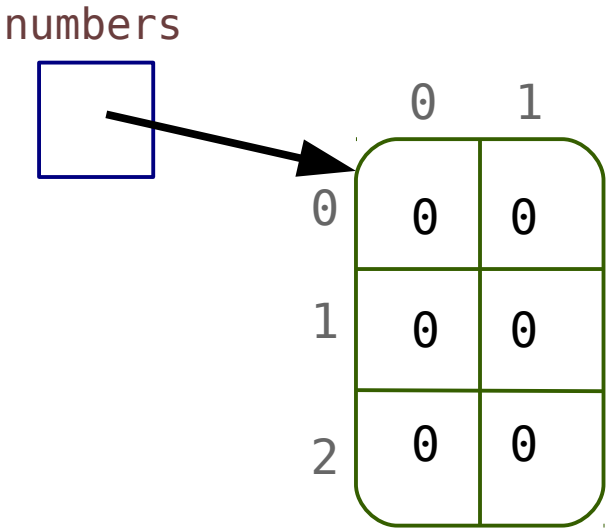

This is an array of String[] entries.



# Instantiating 2D Arrays

Rows Columns

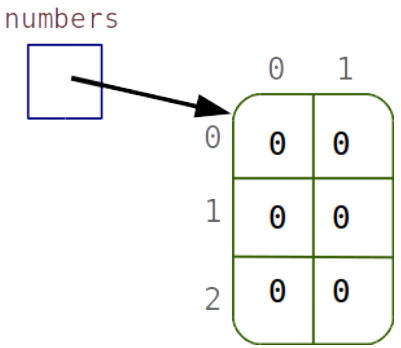
```
numbers = new int[3][2];
```



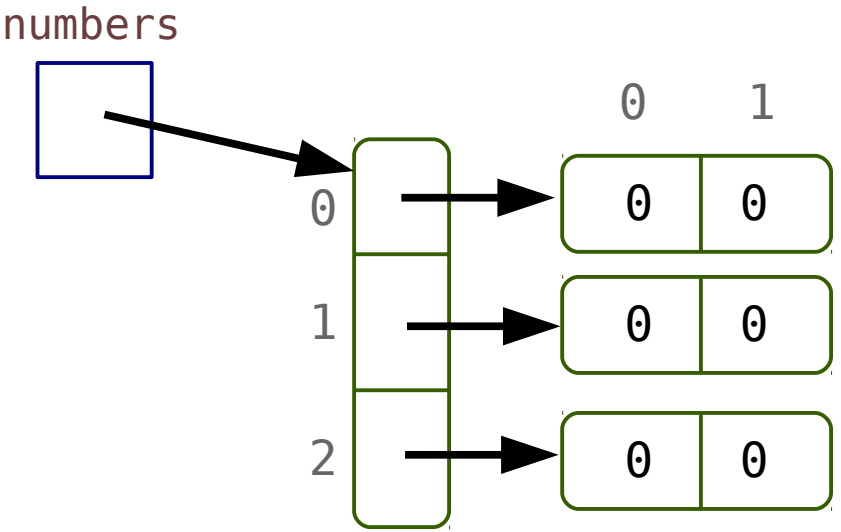
# Instantiating 2D Arrays

Rows Columns

```
numbers = new int[3][2];
```




This is more accurate...



# Socrative Quiz

- Draw the contents of memory. What will be printed?

```
int[][] numbers = new int[2][3];
numbers[0][1] = 7;
numbers[1] = new int[4];

for (int i = 0; i < 2; i++)
{
    System.out.println("val: " + numbers[i][1]);
    System.out.println("len: " + numbers[i].length);
}
```

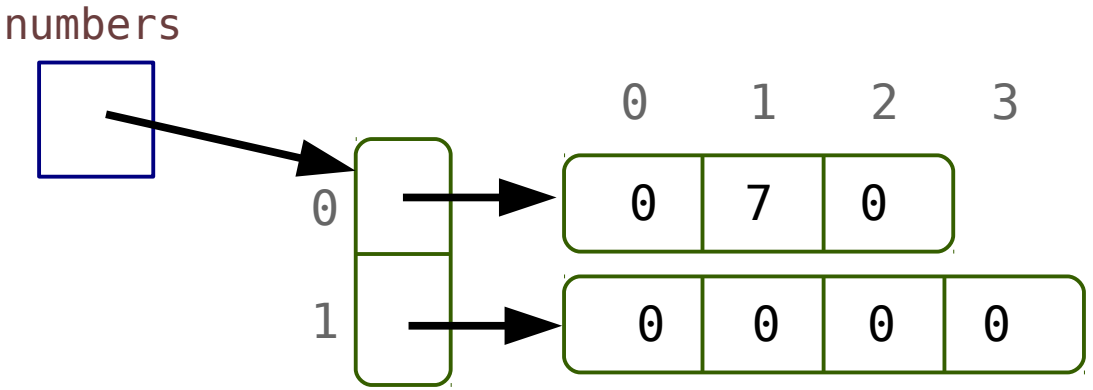
A)
```
val: 0
len: 3
val: 0
len: 4
```

B)
```
val: 7
len: 2
val: 0
len: 2
```

C)
```
val: 7
len: 3
val: 0
len: 4
```

D)
```
val: 7
len: 3
val: 0
len: 3
```

# Solution...

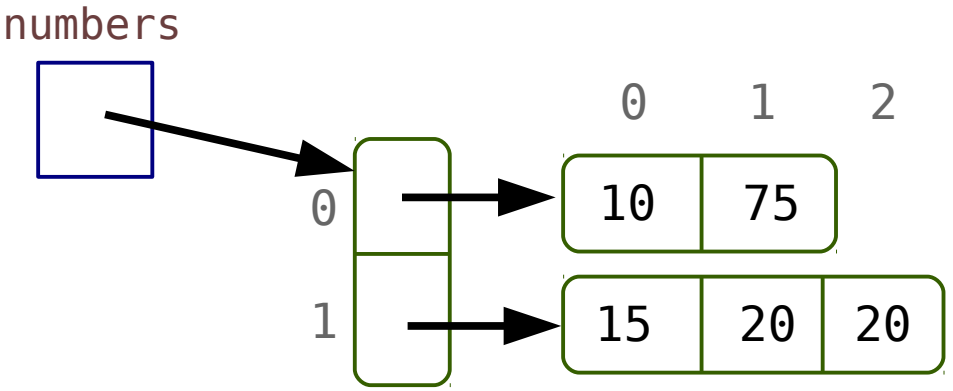



```
val: 7
len: 3
val: 0
len: 4
```

(This is a “ragged” array)

# 2D Array Initialization...

```
int[][] donations = { {10, 75},
                      {15, 20, 20} };
```



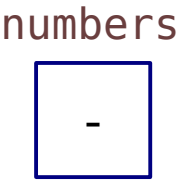
# 2D Array Initialization

```
int[][] donations;

// 2 rows, no columns
donations = new int[2][];

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;
```



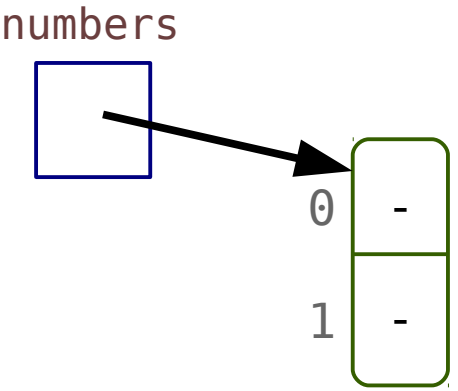
# 2D Array Initialization

```
int[][] donations;

// 2 rows, no columns
donations = new int[2][];

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;
```



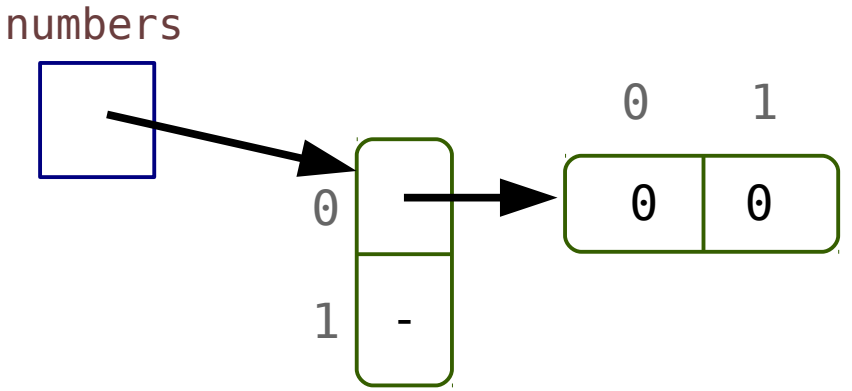
# 2D Array Initialization

```
int[][] donations;

// 2 rows, no columns
donations = new int[2][];

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;
```



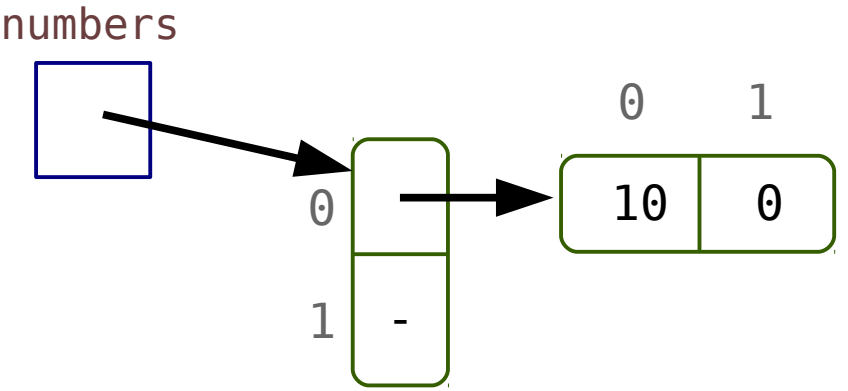
# 2D Array Initialization

```
int[][] donations;

// 2 rows, no columns
donations = new int[2][];

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;
```



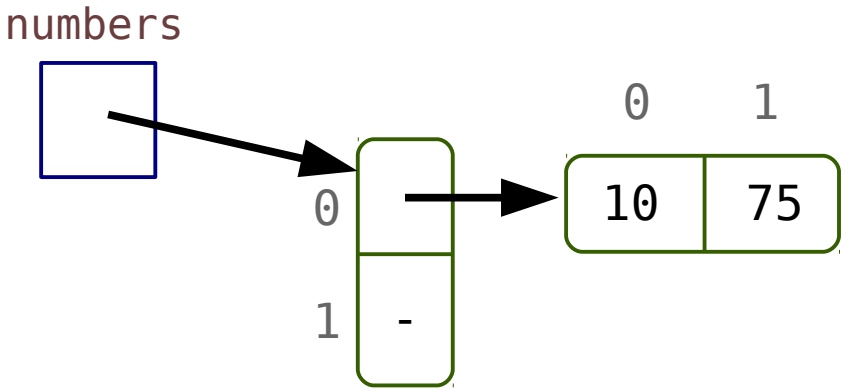
# 2D Array Initialization

```
int[][] donations;

// 2 rows, no columns
donations = new int[2][];

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;
```



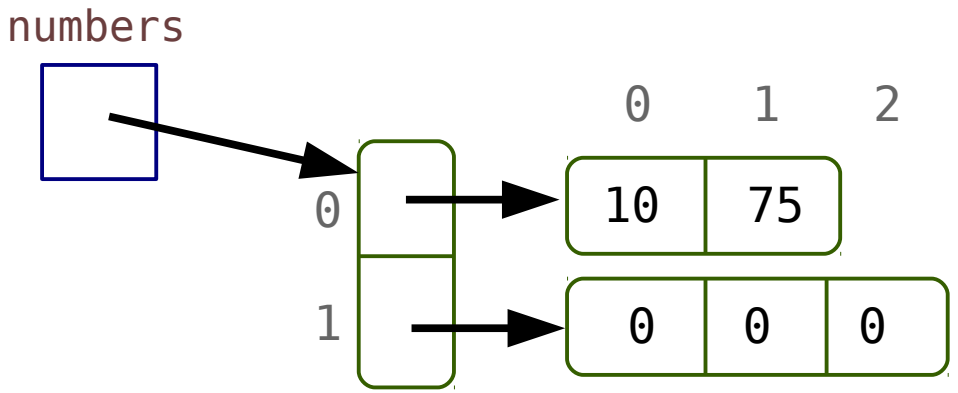
# 2D Array Initialization

```
int[][] donations;

// 2 rows, no columns
donations = new int[2][];

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;
```



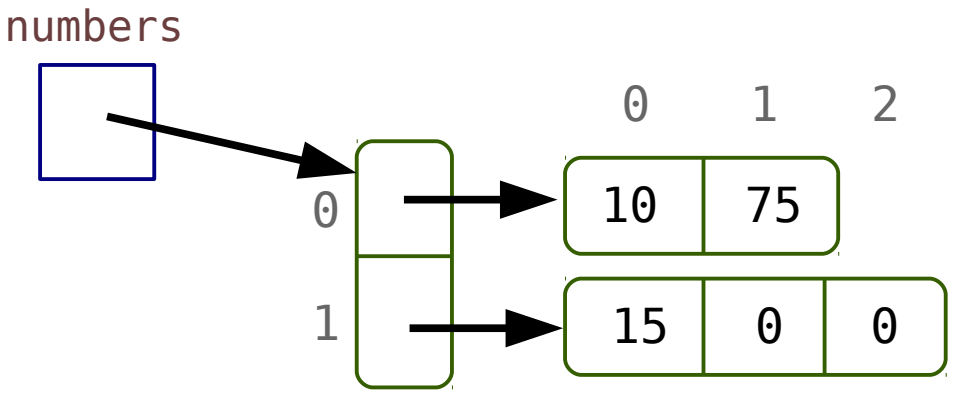
# 2D Array Initialization

```
int[][] donations;

// 2 rows, no columns
donations = new int[2][];

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;
```



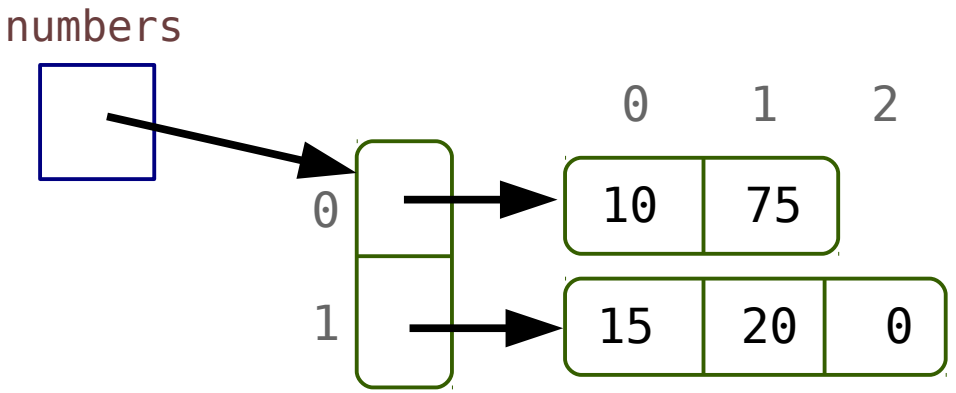
# 2D Array Initialization

```
int[][] donations;

// 2 rows, no columns
donations = new int[2][];

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;
```



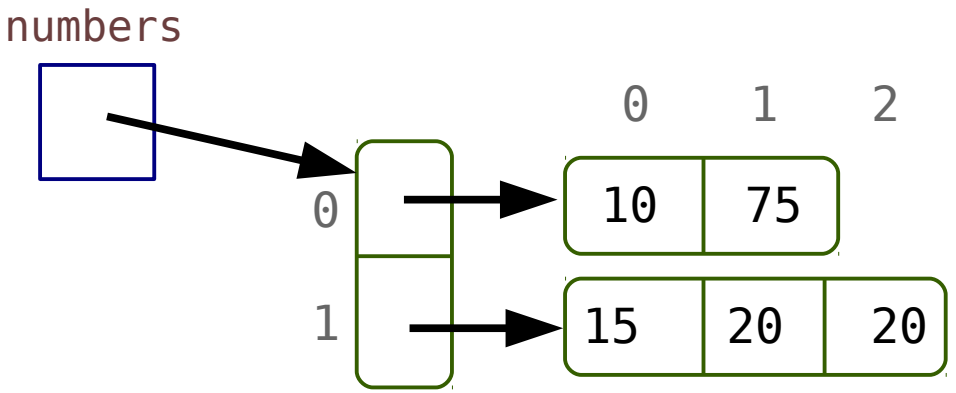
# 2D Array Initialization

```
int[][] donations;

// 2 rows, no columns
donations = new int[2][];

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;
```



# 2D Array Example...

```
public static int maximum1(int[][] values, int rows, int columns)
{
    int maxVal = Integer.MIN_VALUE;

    for (int row = 0; row < rows; row++)
    {
        for (int col = 0; col < columns; col++)
        {
            if (values[row][col] > maxVal)
            {
                maxVal = values[row][col];
            }
        }
    }
    return maxVal;
}
```

Problems with this code?

# 2D Array Example...

```
public static int maximum1(int[][] values, int rows, int columns)
{
    int maxVal = Integer.MIN_VALUE;

    for (int row = 0; row < rows; row++)
    {
        for (int col = 0; col < columns; col++)
        {
            if (values[row][col] > maxVal)
            {
                maxVal = values[row][col];
            }
        }
    }
    return maxVal;
}
```

Problems with this code? 1) Only works with rectangular arrays.

2) No need to pass dimensions.

# 2D Array Example (V2)

```
public static int maximum2(int[][] values)
{
    int maxVal = Integer.MIN_VALUE;

    for (int row = 0; row < values.length; row++)
    {
        for (int col = 0; col < values[row].length; col++)
        {
            if (values[row][col] > maxVal)
            {
                maxVal = values[row][col];
            }
        }
    }
    return maxVal;
}
```

# 2D Array Example (V2)

```
public static int maximum2(int[][] values)
{
    int maxVal = Integer.MIN_VALUE;

    for (int row = 0; row < values.length; row++)
    {
        for (int col = 0; col < values[row].length; col++)
        {
            if (values[row][col] > maxVal)
            {
                maxVal = values[row][col];
            }
        }
    }
    return maxVal;
}
```

Better! What if we are worried about nulls?

# 2D Array Example (V3)

```
public static int maximum3(int[][] values)
{
    int maxVal = Integer.MIN_VALUE;

    if (values != null) // Make sure values is initialized.
    {
        for (int row = 0; row < values.length; row++)
        {
            if (values[row] != null) // Make sure the row is initialized.
            {
                for (int col = 0; col < values[row].length; col++)
                {
                    if (values[row][col] > maxVal)
                    {
                        maxVal = values[row][col];
                    }
                }
            }
        }
    }

    return maxVal;
}
```

# 2D Array Example (V3)

```
public static int maximum3(int[][] values)
{
    int maxVal = Integer.MIN_VALUE;

    if (values != null) // Make sure values is initialized.
    {
        for (int row = 0; row < values.length; row++)
        {
            if (values[row] != null) // Make sure the row is initialized.
            {
                for (int col = 0; col < values[row].length; col++)
                {
                    if (values[row][col] > maxVal)
                    {
                        maxVal = values[row][col];
                    }
                }
            }
        }
    }

    return maxVal;
}
```

Note: This isn't really *better* than V2.

It probably makes more sense to require the caller to provide non-broken arrays.

# 2D Array Example (V4)

```
public static int maximum4(int[][] values)
{
    int maxVal = Integer.MIN_VALUE;

    for (int[] row : values)
    {
        for (int value : row)
        {
            if (value > maxVal)
            {
                maxVal = value;
            }
        }
    }

    return maxVal;
}
```

# 2D Array Example (V4)

```
public static int maximum4(int[][] values)
{
    int maxVal = Integer.MIN_VALUE;

    for (int[] row : values)
    {
        for (int value : row)
        {
            if (value > maxVal)
            {
                maxVal = value;
            }
        }
    }

    return maxVal;
}
```

Personally, I find this less clear than previous versions, but it works.