Multidimensional Arrays

Forest Agostinelli
University of South Carolina

Outline

- Creating multidimensional arrays
- Example

Creating Multidimensional Arrays

- Arrays are a collection of variables of the same type
- Foundational Data Structure
- Contiguous Block of Memory
  - The size of the Array must be specified initially
  - Arrays cannot be resized
- In Java, Arrays are considered a special kind of Object
  - Container Object
  - Identifiers contain only the reference to its contents
  - The reference *points* to contents
  - "==" Does not check the contents of the array

Creating an Array Syntax

```
//Declaring an Array
<<type>>[] <<id>>;
//Initializing an Array]
<<id>> = new <<type>>[<<size>>];
//or
<<type>>[] <<id>> = new <<type>>[size];
```

Example

```
//Creates an array of 5 integers
int[] array = new int[5];
```

Creating Multidimensional Arrays

- Arrays may have multiple dimensions
  - More square brackets ("[]") means more dimensions
- Java creates an "Array of Arrays" for multidimensional arrays
  - Arrays are considered container objects
  - The identifier contains a reference to the first array
  - Then Arrays contain memory addresses to other arrays

Creating an 2D Array Syntax

```
//Declaring a 2D Array
<<type>>[][] <<id>> = new <<type>>[<<size01>>][<<size02>>];
```

Example

```
//Creates a 2 x 3 2D Array of integers
int[][] array = new int[2][3];
```

Creating Multidimensional Arrays

- If the values are known, it is possible to both construct the array and initialize the values at the same time.
- Values are put inside of curly braces ("{}")
- Each value is separated by a comma (",")
- For each dimension include additional curly braces ("{}")

Creating an 2D Array Syntax

```
//Declaring an Array and Initializing its Values
<<type>>[][] <<id>> = {{<<00>>,<<01>>…},{<<10>>,<<11>>,…}};
```

Example

```
//Creates a 2 x 3 2D Array of integers
int[][] array = {{0,1,2},{1,2,3}};
```

Creating Multidimensional Arrays

```
//Creates a 2 x 3 2D Array of integers
int[][] array = new int[2][3];
```

Memory

Identifier	Contents	Byte Address
…	…	…
…	…	…

Creating Multidimensional Arrays

```
//Creates a 2 x 3 2D Array of integers
int[][] array = new int[2][3];
```

Memory

Identifier	Contents	Byte Address
…	…	…
array	NULL	28
…	…	…
…	…	…

Creating Multidimensional Arrays

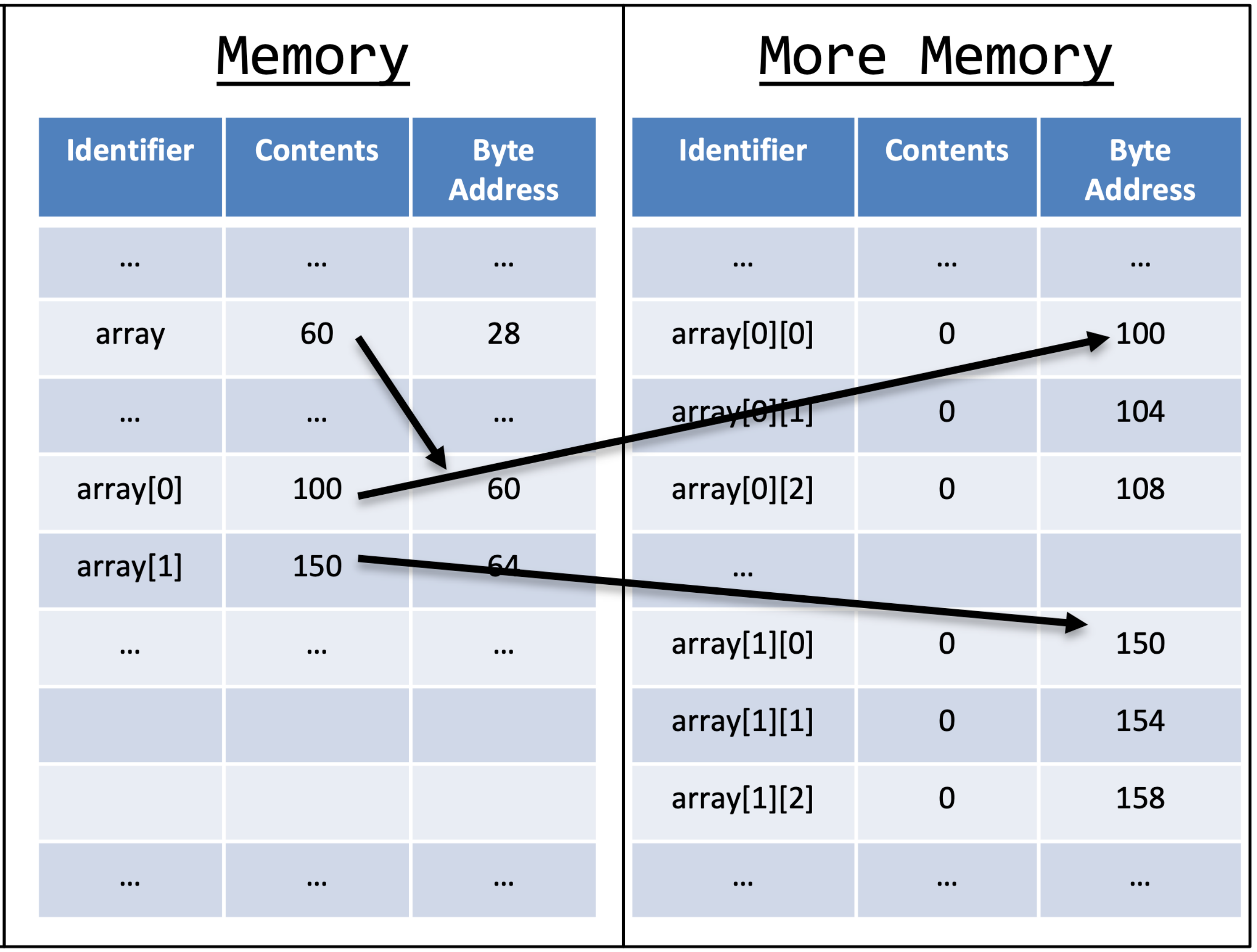
```
//Creates a 2 x 3 2D Array of integers
int[][] array = new int[2][3];
```

Memory

Identifier	Contents	Byte Address
…	…	…
array	NULL	28
…	…	…
array[0]	NULL	60
array[1]	NULL	64
…	…	…
…	…	…

Creating Multidimensional Arrays

```
//Creates a 2 x 3 2D Array of integers
int[][] array = new int[2][3];
```



Memory

Identifier	Contents	Byte Address
…	…	…
array	60	28
…	…	…
array[0]	100	60
array[1]	150	64
…	…	…
…	…	…

More Memory

Identifier	Contents	Byte Address
…	…	…
array[0][0]	0	100
array[0][1]	0	104
array[0][2]	0	108
…		
array[1][0]	0	150
array[1][1]	0	154
array[1][2]	0	158
…	…	…

Creating Multidimensional Arrays

- Indices still work the same way
  - Indices start at 0
  - Indices End at Size-1 (or Length-1)
  - Need an index for each dimension
- The size of each dimension can be access through the property ".length"
- Nested For-Loops are the multidimensional arrays "best friend"
  - Counting variables can be used for indexing
  - Using the property ".length" can be used in the Boolean expression

Indexing Syntax

```
//Accessing Data
<<id>>[<<index01>>][<<index02>>];
//Modifying Data
<<id>>[<<index01>>][<<index02>>] = <<value>>;
//Using .length property
<<id>>.length;//Outside dimension
<<id>>[<<index>>].length;//Inside dimension
```

Example

```
//Assigning some values
array[0][0] = 1;
array[1][1] = 5;
//Accesses and adds the assigned values
int added = array[0][0] + array[1][1];
```

Creating Multidimensional Arrays

```
//Creates a 2 x 3 2D Array of integers
int[][] array = new int[2][3];
for(int i=0;i<array.length;i++)
{
     for(int j=0;j<array[i].length;j++)
     {
          array[i][j] = i+j;
     }
}
```

Another Perspective

j i	0	1	2
0			
1			

Creating Multidimensional Arrays

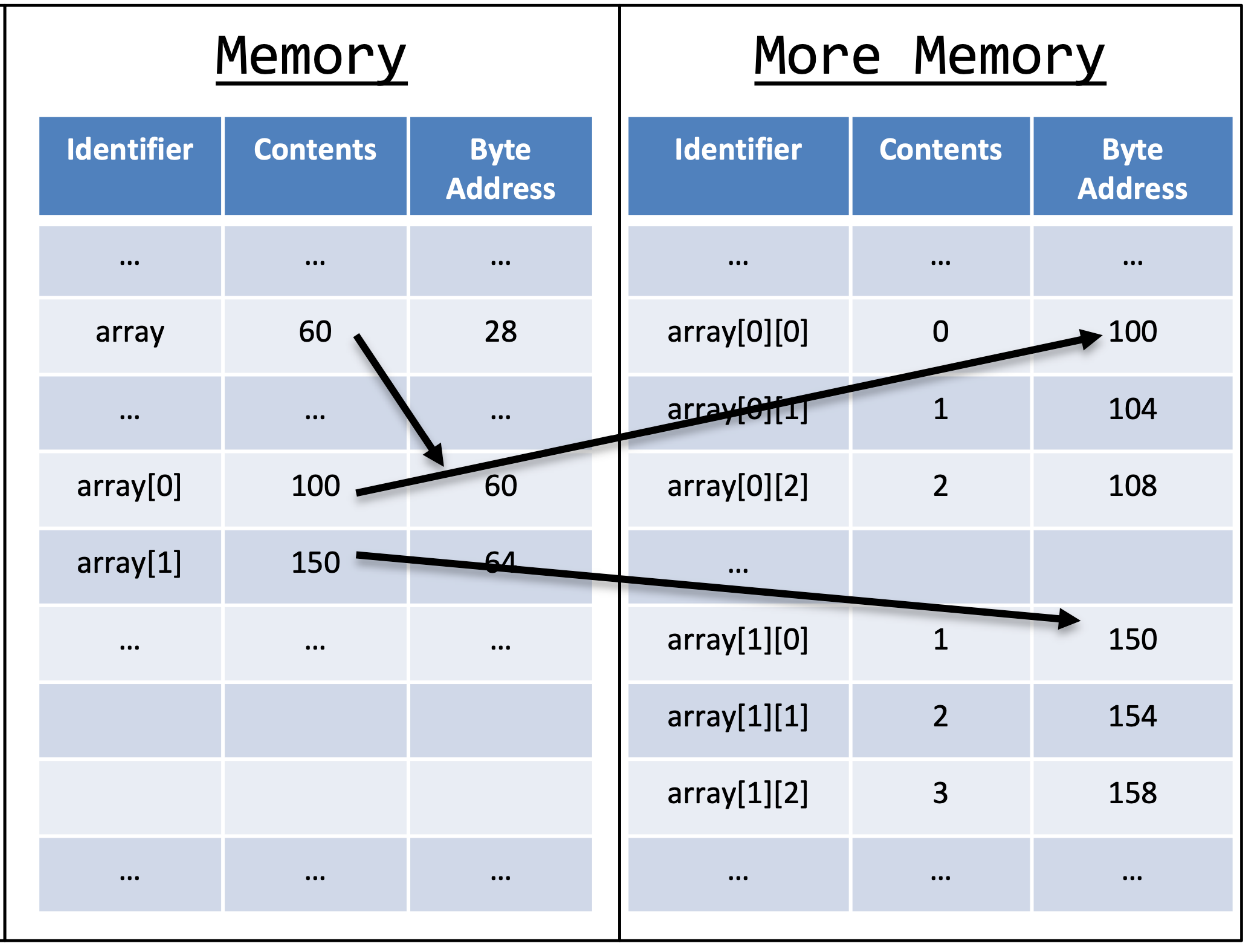
```
//Creates a 2 x 3 2D Array of integers
int[][] array = new int[2][3];
for(int i=0;i<array.length;i++)
{
      for(int j=0;j<array[i].length;j++)
      {
            array[i][j] = i+j;
      }
}
```

Another Perspective

i \ j	0	1	2
0	0	1	2
1	1	2	3

Creating Multidimensional Arrays

```
//Creates a 2 x 3 2D Array of integers
int[][] array = new int[2][3];
for(int i=0;i<array.length;i++)
{
      for(int j=0;j<array[i].length;j++)
      {
            array[i][j] = i+j;
      }
}
```



Memory

Identifier	Contents	Byte Address
…	…	…
array	60	28
…	…	…
array[0]	100	60
array[1]	150	64
…	…	…
…	…	…

More Memory

Identifier	Contents	Byte Address
…	…	…
array[0][0]	0	100
array[0][1]	1	104
array[0][2]	2	108
…		
array[1][0]	1	150
array[1][1]	2	154
array[1][2]	3	158
…	…	…

Outline

- Creating multidimensional arrays
- Example

Example

```java
/*
 * Written by JJ Shepherd
 */
import java.util.Scanner;
import java.util.Random;
public class HideAndSeek {

    public static final int BOARD_SIZE = 10;
    public static final int COLD_DIST = (BOARD_SIZE/2)*(BOARD_SIZE/2);
    public static final int WARM_DIST = (BOARD_SIZE/4)*(BOARD_SIZE/4);

    public static final char EMPTY = '_';
    public static final char PLAYER = 'X';
    public static final char WALKED_PATH = '#';
    public static final char GOAL = '_';

    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);
        Random r = new Random();

        int pX = 0;
        int pY = 0;

        int gX = r.nextInt(BOARD_SIZE);
        int gY = r.nextInt(BOARD_SIZE);

        char[][] board = new char[BOARD_SIZE][BOARD_SIZE];
        for(int i=0;i<board.length;i++)
        {
            for(int j=0;j<board[i].length;j++)
            {
                board[i][j] = EMPTY;
            }
        }

        board[pY][pX] = PLAYER;
        board[gY][gX] = GOAL;

        System.out.println("Welcome to hide and seek!");
        boolean gameOver = false;
        while(!gameOver)
        {
            for(int i=0;i<board.length;i++)
            {
                for(int j=0;j<board[i].length;j++)
                {
                    System.out.print(board[i][j]);
                }
                System.out.println();
            }
            int distance = (pX-gX)*(pX-gX)+(pY-gY)*(pY-gY);
            if(distance > COLD_DIST)
            {
                System.out.println("You are getting colder");
            }
            else if(distance > WARM_DIST)
            {
```

Example

```
                System.out.println("You are getting warmer");
            }
            else
            {
                System.out.println("You are getting hotter!");
            }
            System.out.println("Enter either -1, 0, or 1 to move in the x");
            int dX = keyboard.nextInt();
            System.out.println("Enter either -1, 0, or 1 to move in the y");
            int dY = keyboard.nextInt();
            if(dX < -1 || dX > 1)
            {
                System.out.println("That is invalid");
                dX = 0;
            }
            if(dY < -1 || dY > 1)
            {
                System.out.println("That is invalid");
                dY = 0;
            }
            board[pY][pX] = WALKED_PATH;
            pX += dX;
            pY += dY;

            if(pX < 0)
            {
                pX = 0;
            }
            else if(pX > BOARD_SIZE-1)
            {
                pX = BOARD_SIZE-1;
            }
            if(pY < 0)
            {
                pY = 0;
            }
            else if(pY > BOARD_SIZE-1)
            {
                pY = BOARD_SIZE-1;
            }
            board[pY][pX] = PLAYER;
            if(pX == gX && pY == gY)
            {
                System.out.println("You win!");
                gameOver = true;
            }
        }
    }

}
```

Example

X									
						O			

Example

#									
	#	#							
		X							

Example

#									
	#	#							
			#						
			#						
	#	#							
	#					X			
	#					#			
		#				#			
			#		#				
				#					

Ragged Arrays

- Java allows multidimensional arrays to have different sizes for each dimension
  - Referred to as "Ragged Arrays"
  - Important to use <<id>>[<<index>>].length to ensure the correct size
- Not all programming languages allow this

Creating a Ragged 2D Array Syntax

```
//Declaring a Ragged Array
<<type>>[][] <<id>>;
<<id>> = new <<type>>[<<size for outside array>>];
<<id>>[<<index0>>] = new <<type>>[<<size at index 0>>];
<<id>>[<<index1>>] = new <<type>>[<<size at index 1>>];
…
```

Example

```
//Declare the array
int[][] a;
//Construct outside array
a = new int[3];
//Construct internal arrays
a[0] = new int[5];//First row has 5 elements
a[1] = new int[8];//Second row has 8 elements
a[2] = new int[2];//Third row has 2 elements
```