ICS 111
Introduction to Computer Science I

Nikki Manuel
Leeward Community College

Arrays

Week 8
Spring 2019

Arrays

- What is an array?
- Array of primitive data types
- Array of strings
- Solving problems with arrays

Array Basics

Concept of an Array

- What if you need to read 100 numbers, compute their average, and find out how many numbers are above the average?
  - You will need 100 variables to store each number
  - You will need to write almost identical code 100 times
  - This is impractical !!!

Concept of an Array

- Java provides a *data structure* called an **array**, which stores a sequential collection of elements of the same type
    - For our problem, we can store all 100 numbers into an array, and access them through a single array variable.

Concept of an Array

- An array is used to store a collection of data
  - more specifically: variables of the same type

- Instead of declaring individual variables, such as

  `number0`, **`number1`**, … , **`number99`**

  we simply declare one array variable such as **`numbers`**, and refer to the individual numbers as

  `numbers[0]`, **`numbers[1]`**, … , **`numbers[99]`**

What is an array?

- Programming language construct (most languages have this)
- Multi-valued variable
- Used to group related data of the same type
- Organizes and manages data

When do we use arrays?

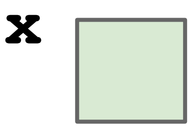
- When we have many variables of the same type that are processed in the same way
- When using single-valued variables
  - In a loop, the program doesn’t know when to move on to the next variable
  - They are disjoint

Arrays in Memory

A primitive variable:

```
int x;
```

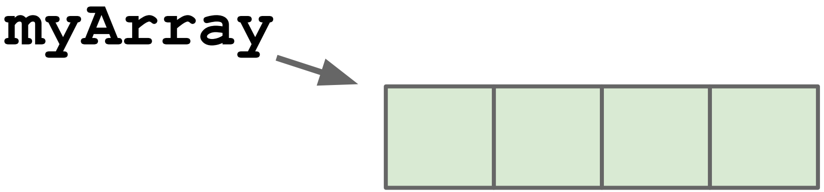
`x` is the memory location



An array (is an object):

```
int[] myArray;
```

`myArray` points to the memory location



Declaring an Array - Syntax

```
type[] arrayName;
```

- `type[]` : the data type - all values must be of the same type
- `arrayName` : the variable name to reference the array

- For example, the following code declares a variable `myArray` that references an array of integer elements:

```
    int[] myArray;
```

Declaring Arrays - Notes

- Unlike when we declare primitive data type variables, the declaration of an array variable does not allocate any space in memory for the array.
  - It only creates a storage location for the reference to an array
- You cannot assign elements to an array unless it has already been created

Creating an Array - Syntax

- After an array variable is declared, you can create an array by using the `new` operator and assign its reference to the variable with the following syntax:

```
arrayName = new type[arraySize];
```

1. An array is created using `new type[arraySize]`
2. The reference of the newly created array is assigned to the variable `arrayName`

Declaring & Creating Arrays – Syntax

```
type[] arrayName = new type[arraySize];
```

- `type[]` : the data type - all values must be of the same type
- `arrayName` : the variable name
- `arraySize` : the number of indices in the array - how many elements you want the array to be able to hold

Declaring & Create an Array

```
// Create an array with 4 blank indices
int numIndex = 4;
int[] myArray = new int[numIndex];
```

Notes on Arrays

- You must declare the size of the array
- Each position in the array is called an index
- 0-based indexing
- The size does not and will not change
- If you need to grow an array, you need to make another array and copy the contents to the new one

Getting the Value at a Specified Index

`name[index]`

- **`name`** : The name of the array
- **`index`** : The position of your desired value
  - A number from 0 to (**`size`** - 1)

Assign a Value to a Specified Index

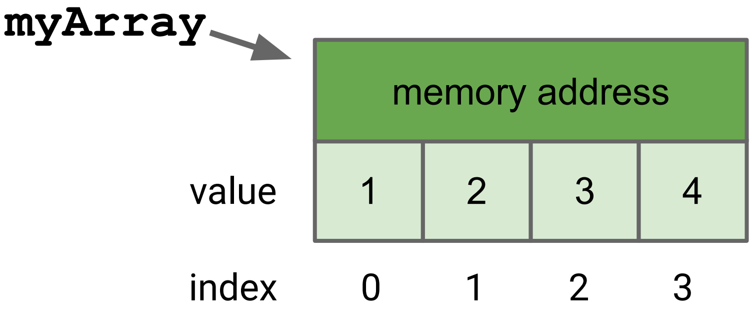
```
name[index] = value;
```

- Use an assignment statement
- `index` : The position of your desired value
  - A number from 0 to (`size` - 1)
- `value` : Make sure the value is the same type as the array
  - Or a value that can be promoted (int -> double)

Declaring, Creating, & Populating an Array #1

```
// Create an array with 4 blank indices
int numIndex = 4;
int[] myArray = new int[numIndex];

// Create and populate the array with values
myArray[0] = 1;
myArray[1] = 2;
myArray[2] = 3;
myArray[3] = 4;
```



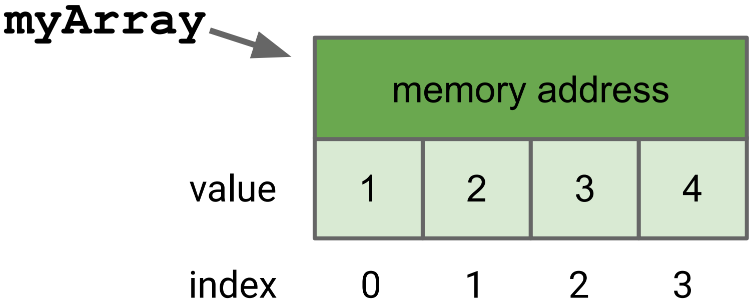
myArray

memory address

value	1	2	3	4
index	0	1	2	3

Declaring, Creating, & Populating an Array #2

```
// Declare, create, initialize all in one line:
int[] myArray = {1, 2, 3, 4};
```



myArray

memory address

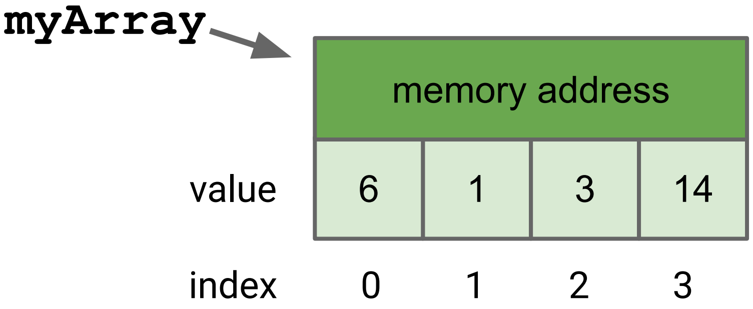
value	1	2	3	4
index	0	1	2	3

Array Properties : Indices & Array Length

- If **`N`** is how many indices are in the array:
  - It will have positions 0 to **`N`** – 1
- To find the size of the array, we use **`length`**
  - **`length`** is a property of the array, not a method

Finding Array Size

```
int[] myArray = {6, 1, 3, 14};
int arrayLen = myArray.length;   // No parentheses!
System.out.println(arrayLen);    // 4
```



myArray

memory address

value	6	1	3	14
index	0	1	2	3

The array **`myArray`** contains 4 values. Therefore, we say it is of length 4.

`arrayLen` is 4

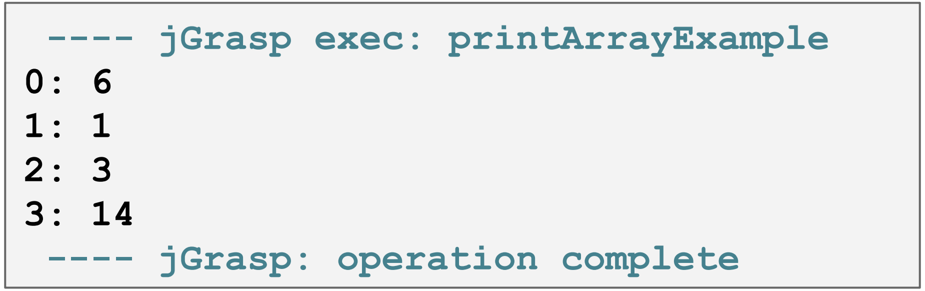
Printing Arrays

- Use a loop to print the array
- The loop should start from index 0 to the last index of the array

Printing Arrays

```java
int[] myArray = {6, 1, 3, 14};
int arrayLen = myArray.length;

for (int i = 0; i < arrayLen; i++) {
   System.out.println(i + ": " + myArray[i]);
}
```

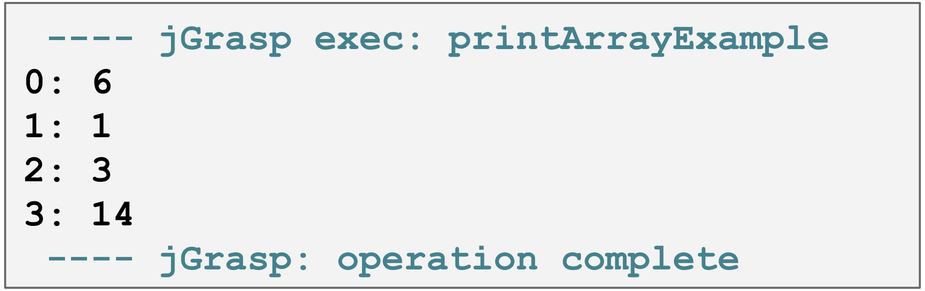


```
 ---- jGrasp exec: printArrayExample
0: 6
1: 1
2: 3
3: 14
 ---- jGrasp: operation complete
```

Printing Arrays - An Alternative

```java
int[] myArray = {6, 1, 3, 14};

for (int i = 0; i < myArray.length; i++) {
   System.out.println(i + ": " + myArray[i]);
}
```



```
 ---- jGrasp exec: printArrayExample
0:  6
1:  1
2:  3
3:  14
 ---- jGrasp: operation complete
```

Using Other Data Types in Arrays

Dealing with Other Types

- `double`
- `boolean`
- `char`
- `String`

Declaring Arrays

```
final int ARRAY_SIZE = 20;

double[] midtermGrades = new double[ARRAY_SIZE];

boolean[] booleanArray = new boolean[ARRAY_SIZE];

String[] helloWords = new String[ARRAY_SIZE];
```

Declaring & Initializing Arrays

```
double[] midtermGrades = {78.0, 92.5, 0, 99.9};

boolean[] booleanArray = {true, false, false};

String[] helloWords = {"hello", "hi", "hey"};
```

Empty Arrays?

- An array of a given size and type is never empty
- It will always contain "something"
- So what is in the uninitialized array?
  - It depends!

~~Empty~~ Non-Empty Arrays!

- An array of **`int/double`** will contain : **`0`**
- An array of **`char`** will contain: **`\u0000`** (the min value of a char)
- An array of **`boolean`** will contain: **`false`**
- An array of **`String`** will contain: **`null`**
  - More generally, an array of objects is **`null`**

Populating an Array of doubles

```java
double[] midtermGrades = new double[20];
Scanner reader = new Scanner(System.in);

for (int i = 0; i < midtermGrades.length; i++) {
   midtermGrades[i] = reader.nextDouble();
}
```

Populating an Array of Strings

```
String[] helloWords = new String[20];
Scanner reader = new Scanner(System.in);

for (int i = 0; i < helloWords.length; i++) {
   helloWords[i] = reader.nextLine();
}
```