Java Programming

Arrays - revisited

Credit: Slides are modified with permission from Barry Wittman at Elizabethtown College

Why Arrays?

- Variables are nice
- Loops are great
- Without a way to talk about a **set** of variables, we cannot get the full potential out of a loop
- Enter: **arrays**

Definition of an Array

- An array is a set or list of data that is the same type: **int**, **double**, **String**, etc.
- The size of the array is fixed when you create it meaning it cannot adjust once it is created
- This is similar to a String if you think of how it contains a list of chars and the String is immutable

Array Syntax

Declaration of an Array

- To declare an array of a specified **type** with a given **name**:

```
type[] name;
```

- Example with a list of type **int**:

```
int[] list;
```

- Like any variable declaration, but with **[]**

Instantiation of an Array

- When you declare an array, you are creating a variable that can hold an array
- At first, it holds nothing, also know as **null**
- To use it, you have to instantiate an array, supplying a specific size:

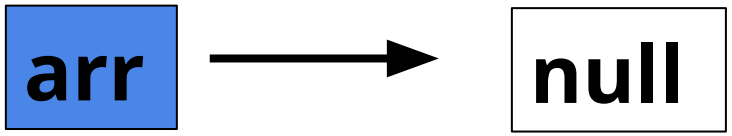
```
int[] list; // declaration - null
list = new int[10]; // instantiation
```

- This code creates an array of 10 **int**s

Arrays are Objects

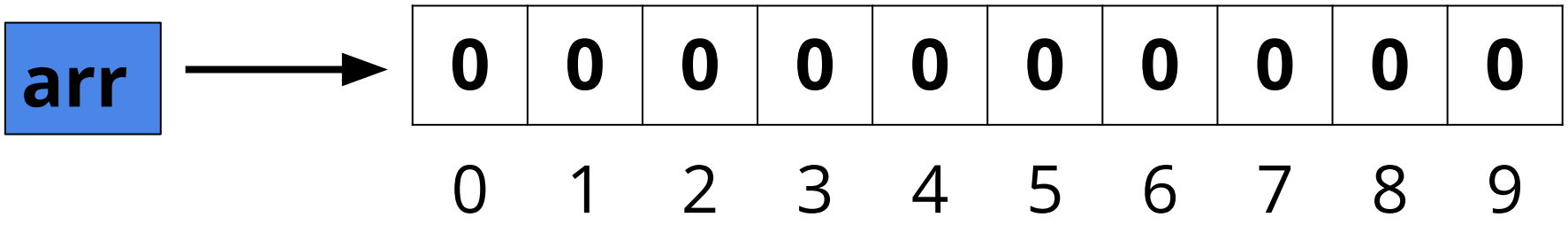
- Declares array, but assigned to null

```
int[] arr;
```



- Instantiates an existing array

```
arr = new int[10];
```



Accessing Array Elements

- You can access an element of an array by using the index inside brackets

```
list[9] = 142;
System.out.println(list[9]);
```

- Once you have indexed into an array, that variable behaves exactly like any other variable of that type
- **Indexing starts at 0 and stops at 1 less than the length**
- The index can be any number, variable, or expression that evaluates to an **integer**

Assigning an Array Element

Array Before:

0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9

```
int[] list = new int[10];
list[0] = 73;
list[4] = 2;
list[9] = 14;
System.out.println(list[0]);
System.out.println(list[4]);
System.out.println(list[7]);
System.out.println(list[9]);
```

Output
73
2
0
14

Array After:

73	0	0	0	2	0	0	0	0	14
0	1	2	3	4	5	6	7	8	9

Length of an Array

- You can use the **length** member/attribute to find out how many elements are in the array
- Please note the difference from the **length()** method for Strings and the **length** member for Arrays
- One is a method and one is a member/attribute

```
int[] list = new int[42];
int size = list.length;
System.out.println("List has " + size +
   " elements");
```

Output
42

Automatic Initialization

- When you create an **int**, **double**, **char**, or **boolean** array, the array is automatically filled with certain values

Type	Value
int	**0**
double	**0.0**
char	**'\0'**
boolean	**false**

- For other types, including **Strings**, each index in the array must be filled explicitly

Explicit Initialization

- Explicit initialization can be done with a list:

```
String[] days = {"Monday", "Tuesday",
"Wednesday", "Thursday", "Friday",
"Saturday", "Sunday"};
```

- Or, a loop could be used to set all the values:

```
String[] numbers = new String[100];
for(int i = 0; i < numbers.length; i++)
  numbers[i] = "" + (i + 1);
```

Activities on arrays

Go to codingbat.com

Start with these activities (Array 1):

- firstLast6
- commonEnd
- sum3
- makeLast
- unlucky1
- front11

Connection to for-loops

for loops + arrays = power

- Arrays are a fixed size list of a single kind of data
- A **for** loop is ideal for iterating over every item and performing some operation
- **for** loops and arrays will come up again and again

for loop going through an array

- Here is an array of **ints** called **list**
- We can use a for loop to go through the array

```
int[] list = {1, 2, 3, 4};
for( int i = 0; i < list.length; i++ ) {
  System.out.print(list[i] + " ");
}
```

Output
1 2 3 4

- Using the length parameter we do not need to know how big the array is ahead of time

for loop for summing an array

- Here is an array of **ints** called **list**
- We can use a for loop to sum up those **ints**

```
int sum = 0;
int[] list = {8, 5, 3, 7, 2};
for(int i = 0; i < list.length; i++) {

  sum += list[i];
}
System.out.println(sum);
```

Output
25

- Using the length parameter we do not need to know how big the array is ahead of time

Activities on arrays/loops

Go to codingbat.com

Start with these activities (Array 2):

- countEvens
- Sum28
- more14
- has22
- matchUp
- shiftLeft
- bigDiff

for each loop

for each loop

- There is a variation of the for loop called the for each loop
- This loop goes through some list of items
- In this case the variable x stores the actual value of an array element

```
int[] list = {1, 2, 3, 4};

for (int x : list) {
   System.out.print(x + " ");
}
```

Output
1 2 3 4

for each loop

- Inside the loop setup you need to define a temporary variable, in this case it is **x**
- The data type for the variable must match the type of data stored in the array
- Then you have a **:** followed by the array you want to go through

```
double[] list = {2.0, 5.0, 3.0};

for (double x : list) {
   System.out.print(x + " ");
}
```

Output
2.0 5.0 3.0

for each loop

- To use the for each loop to go through each character in a String, you need a method to convert the String to an array of chars
- The toCharArray() method does this

```
String word = "hello";
for(char x : word.toCharArray()) {
   System.out.print(x + " ");
}
```

Output
h e l l o

Array Examples

Array Swap

- Swapping the values of two variables is a fundamental operation in programming
- It is going to become more important in arrays because now the order of variables has become important
- The simplest way to swap two variables involves using a third variable as a temporary location

Swap Code

- Here is an example of swapping two **ints** in an array of **ints** called **arr**

```
int[] arr = {8, 3, 6};
int temp;
temp = arr[0];
arr[0] = arr[2];
arr[2] = temp;
System.out.println(arr[0]);
System.out.println(arr[2]);
```

Output
6 8

Why the Third Variable?

- Why do we need the temporary variable?
- What would the output be from the code below?

```
int[] arr = {8, 3, 6};
arr[0] = arr[2];
arr[2] = arr[0];
System.out.println(arr[0]);
System.out.println(arr[2]);
```

Output
6
6

- Without the temporary variable we lose the value of one of the array elements

Shuffling Cards

- Using the swap code, we can do a random shuffling of a deck of cards
- To do so, we go through each element of the array, and randomly swap it with any of the later elements

```
int n = 52;
for(int i = 0; i < n; i++) {
  exchange = i+(int)(Math.random()*(n-i));
  temp = deck[i];
  deck[i] = deck[exchange];
  deck[exchange] = temp;
}
```

Searching

- Searching through an array is an important operation
- The simplest way to do so is a linear search: check every element in the array
- Searching and sorting are really key to all kinds of problems

Searching Example

- This example goes through the array and finds the first occurrence of 4
- You could find the last occurrence by starting at the length-1 and going to 0

```
int[] arr = {8, 3, 4, 6};
for(int x = 0; x < arr.length; x++) {
  if(arr[x] == 4)
    return x;
}
```

Output
2

Counting Occurrences

- This example goes through and counts the number of 4's located in **arr**

```
int[] arr = {8, 3, 4, 6, 4, 9, 4};
int count = 0;
for(int x = 0; x < arr.length; x++) {
  if(arr[x] == 4)
    count++;
}
System.out.println(count);
```

Output
3

Removing Occurrences

- This example goes through and removes any occurrence of 4

```
int[] arr = {8, 3, 4, 6, 4, 9, 4};
int count = 0;
for(int x = 0; x < arr.length; x++){
  if(arr[x] == 4)
    count++;
}
int[] newarr = new int[arr.length-count];
int i = 0;
for(int x = 0; x < arr.length; x++){
  if(arr[x] != 4){
    newarr[i] = arr[x];
    i++;
  }
}
System.out.println(Arrays.toString(newarr));
```

Remember the array cannot be resized so we need to create a new array

Output

[8, 3, 6, 9]

Arrays.sort()

- Here is how you can sort an array
- You will need to add this import statement

```
import java.util.Arrays;
```

```
int[] list = {9, 4, 7, 2};
Arrays.sort(list);
System.out.println(Arrays.toString(list));
```

Output

[2, 4, 7, 9]

Arrays.toString()

- You have already seen toString() throughout the presentation
- The Arrays.toString() method returns a string of the array that is passed in as an argument
- You will again need this import statement

```
import java.util.Arrays;
```

```
int[] list = {9, 4, 7, 2};
System.out.println(Arrays.toString(list));
```

Output

[9, 4, 7, 2]

Common Pitfalls with Arrays

Array Index Errors

- Accessing an element of an array that does not exist will kill your program

```
int[] numbers = new int[100];

numbers[103] = 5;    //crash
System.out.println( numbers[-3]); //crash
System.out.println( numbers[99]);//okay
for( int i = 0; i <= 100; i++ )
  numbers[i] = i;  //crash when i == 100
```

Un-initialized Arrays

- Remember, you get no initialization with arrays of **Strings**
- If you try to access an non-existent element, the world will explode

```
String[] array = new String[50];

String s = array[42]; // works fine
int size = s.length(); // destroys world
```

Weather Lab

Using an array to store the temperatures. Write a program that prompts the user for several days' temperatures and prints out the average and the number of days over the average temperature. Use appropriate prompts and comments in the program.

```
How many days' temperatures? 7
Day 1's high temp: 45
Day 2's high temp: 44
Day 3's high temp: 39
Day 4's high temp: 48
Day 5's high temp: 37
Day 6's high temp: 46
Day 7's high temp: 53
Average temp = 44.6
4 days were above average.
```

Extension to Weather lab…

Keep track of the coldest and hottest two days. See below example:

```
Temperatures: [45, 44, 39, 48, 37, 46, 53]
Two coldest days: 37, 39
Two hottest days: 53, 48
```