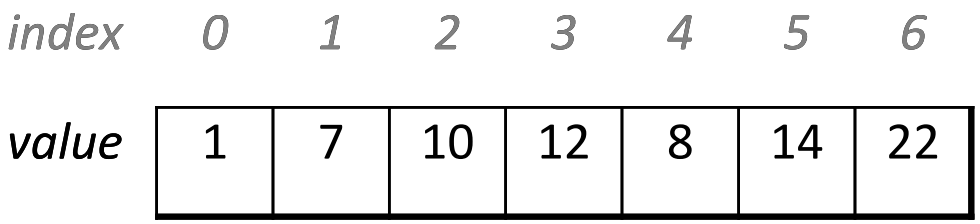
CS 106A, Lecture 15
Arrays

reading:

Art & Science of Java, 11.1 - 11.3

Lecture Outline

- Today we will learn about **arrays**.
  - An array stores many values in a single variable.
  - A powerful tool for storing and manipulating large amounts of **data**, or for grouping a lot of variables together as one variable.
  - Computer's representation of a "list" of items.



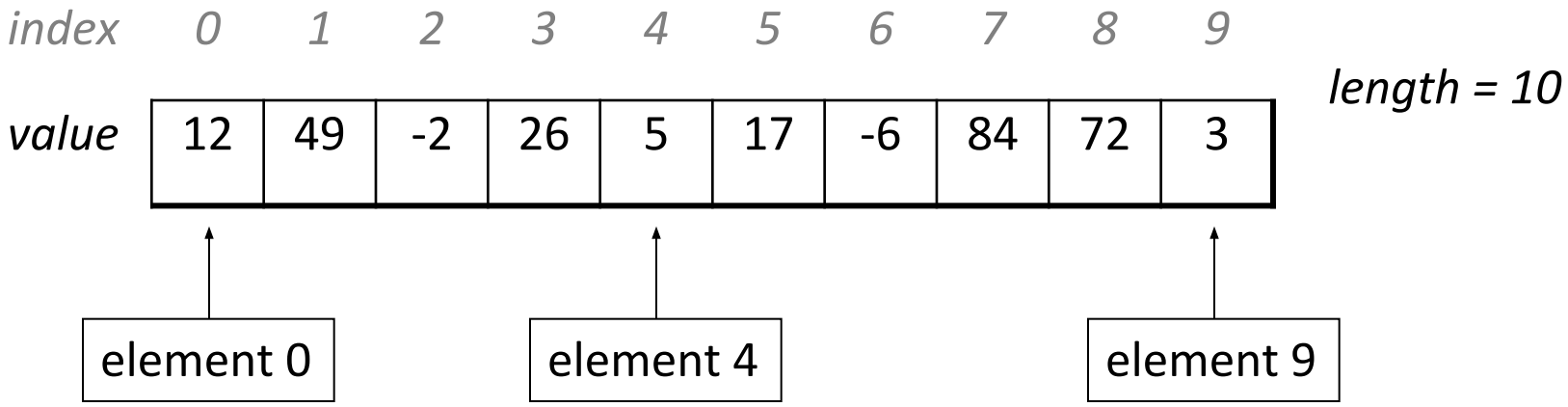
A tricky problem

- Consider the program below that prompts for Big Game margins (the amount Stanford won by) and prints averages, high/lows, etc.

```
How many years of Big Game? 7
Year 1's margin (Stanford - Cal): -14
Year 2's margin (Stanford - Cal): 34
Year 3's margin (Stanford - Cal): 3
Year 4's margin (Stanford - Cal): 18
Year 5's margin (Stanford - Cal): 50
Year 6's margin (Stanford - Cal): 21
Year 7's margin (Stanford - Cal): 13
All game margins: [-14, 34, 3, 18, 50, 21, 13]
Average margin = 17.86
Stanford won 6 games.
Two biggest wins: 50, 34
Two worst games: -14, 3
```

Arrays

- **array**: An object that stores many values of the same type.
  - **element**: One value in an array.
  - **index**: A 0-based integer to access an element from an array.
    - Same as Strings!
  - **length**: Number of elements in the array.



Array declaration

***type*[] *name* = new *type*[*length*];**

– Length explicitly provided when initializing. All elements' values are initially 0.

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

index	*0*	*1*	*2*	*3*	*4*
value	0	0	0	0	0

***type*[] *name* = {*value*, *value*, ..., *value*};**

– Infers length from number of values provided. Example:

```
int[] numbers = {12, 49, -2, 26, 5, 17, -6};
```

index	*0*	*1*	*2*	*3*	*4*	*5*	*6*
value	12	49	-2	26	5	17	-6

Accessing elements

```
name[index]                 // access
name[index] = value;        // modify
```

- Legal indexes: between **0** and the **array's length - 1**.

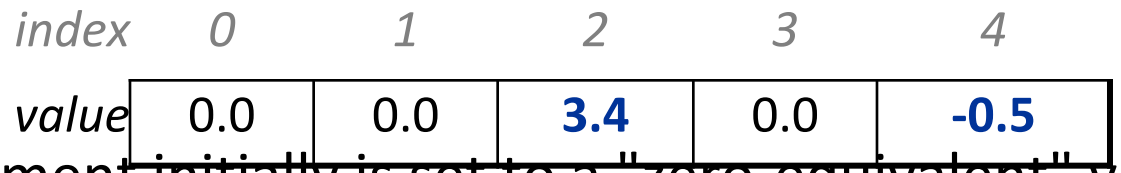
```
int[] numbers = {12, 49, -2, 26, 5, 17, -6};
 numbers[3] = 88;
for (int i = 0; i < 7; i++) {
    print(numbers[i] + " ");
}
println(numbers[-1]);  // exception
println(numbers[7]);   // exception
```

index	*0*	*1*	*2*	*3*	*4*	*5*	*6*
value	12	49	-2	**88**	5	17	-6

Arrays of other types

- You can also create arrays of other types. For example:

```
double[] results = new double[5];
results[2] = 3.4;
results[4] = -0.5;
```



- Each element initially is set to a "zero-equivalent" value.
  - `int` to `0`, `double` to `0.0`, `char` to `'\0'`
  - `String` and other objects to `null`

Accessing elements

arrayElements1

Q: What are the contents of `numbers` after this code?

```
int[] numbers = new int[8];
numbers[1] = 3;
numbers[4] = 7;
numbers[6] = 5;

int x = numbers[1];
numbers[x] = 2;
numbers[numbers[4]] = 9;
```

Arrays and for loops

- It is common to use `for` loops to access array elements.

```
int[] numbers = {0, 3, 0, 2, 7, 0, 5, 9};
for (int i = 0; i < 8; i++) {
   print(numbers[i] + " ");
}
println();                    // output: 0 3 0 2 7 0 5 9
```

- Sometimes we assign each element a value in a loop.

```
for (int i = 0; i < 8; i++) {
    numbers[i] = 2 * i;
}
```

index	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
value	0	2	4	6	8	10	12	14

The length field

- An array's `length` field stores its number of elements.

```
name.length

for (int i = 0; i < numbers.length; i++) {
    print(numbers[i] + " ");
}
// output: 0 2 4 6 8 10 12 14
```

- Note: NO parentheses like String's `.length()`
- What expressions refer to:
  - The last element of any array?
  - The middle element?

The length field

- An array's `length` field stores its number of elements.

```
name.length

for (int i = 0; i < numbers.length; i++) {
    print(numbers[i] + " ");
}
// output: 0 2 4 6 8 10 12 14
```

- Note: NO parentheses like String's `.length()`
- What expressions refer to:
  - The last element of any array? `name[name.length - 1]`
  - The middle element?

The length field

- An array's `length` field stores its number of elements.

```
name.length

for (int i = 0; i < numbers.length; i++) {
    print(numbers[i] + " ");
}
// output: 0 2 4 6 8 10 12 14
```

- Note: NO parentheses like String's `.length()`
- What expressions refer to:
  - The last element of any array? `name[name.length - 1]`
  - The middle element? `name[name.length / 2]`

"Array mystery" problem

arrayMystery1

- What values are stored in the array after the code below?
  - **traversal**: An examination of each element of an array.

```
int[] a = {1, 7, 5, 6, 4, 14, 11};
for (int i = 0; i < a.length - 1; i++) {
    if (a[i] > a[i + 1]) {
        a[i + 1] = a[i + 1] * 2;
    }
}
```

Limitations of arrays

- You cannot resize an existing array:

```
int[] a = new int[4];
a.length = 10;                          // error
```

- You cannot compare arrays with == or `equals` :

```
int[] a1 = {42, -7, 1, 15};
int[] a2 = {42, -7, 1, 15};
if (a1 == a2) {  ... }                  // false!
if (a1.equals(a2)) {  ... }             // false!
```

- An array does not know how to print itself:

```
println(a1);                            // [I@98f8c4]
```

Arrays.toString

- `Arrays.toString` accepts an array as a parameter and returns a string representation of its elements.
  - This is the preferred way to print the contents of an array.

```
int[] e = {0, 2, 4, 6, 8};
e[1] = e[3] + e[4];
println("e is " + Arrays.toString(e));
```

Output:

```
e is [0, 14, 4, 6, 8]
```

The Arrays class

- Class `Arrays` in package `java.util` has useful methods for manipulating arrays:

Method name	Description
`Arrays.binarySearch(`***`array, value`***`)`	returns the index of the given value in a *sorted* array (or < 0 if not found)
`Arrays.copyOf(`***`array, length`***`)`	returns a new copy of array of given length
`Arrays.equals(`***`array1, array2`***`)`	returns `true` if the two arrays contain same elements in the same order
`Arrays.fill(`***`array, value`***`);`	sets every element to the given value
`Arrays.sort(`***`array`***`);`	arranges the elements into sorted order
`Arrays.toString(`***`array`***`)`	returns a string representing the array, such as `"[10, 30, -25, 17]"`

Big Game exercise

- Write a **`Big Game`** program that prompts the user to enter daily temperatures, and uses an array to produce this output:

```
How many years of Big Game? 7
Year 1's margin (Stanford - Cal): -14
Year 2's margin (Stanford - Cal): 34
Year 3's margin (Stanford - Cal): 3
Year 4's margin (Stanford - Cal): 18
Year 5's margin (Stanford - Cal): 50
Year 6's margin (Stanford - Cal): 21
Year 7's margin (Stanford - Cal): 13

All game margins: [-14, 34, 3, 18, 50, 21, 13]
Average margin = 44.6
Stanford won 6 games.
Two biggest wins: 50, 34
Two worst games: -14, 3
```

Array param/return

```
public void methodName(type[] name) {    // parameter
public type[] methodName(params) {       // return
```

– Example:

```
public int sum(int[] a) {
    int result = 0;
    for (int i = 0; i < a.length; i++) {
        result += a[i];
    }
    return result;
}
```

- Call:

```
int[] numbers = {2, -1, 4, 7};
int total = sum(numbers);          // 12
```