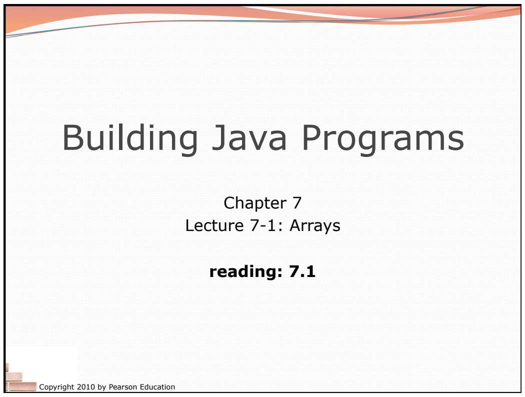
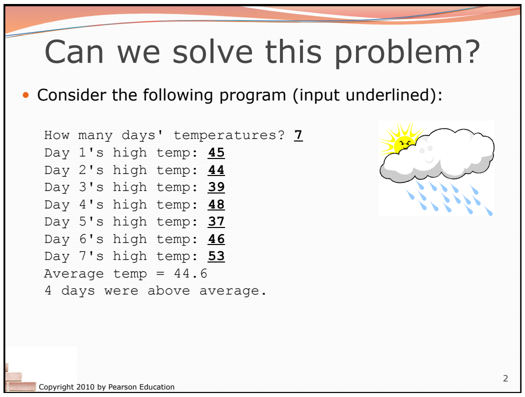


Building Java Programs

Chapter 7
Lecture 7-1: Arrays

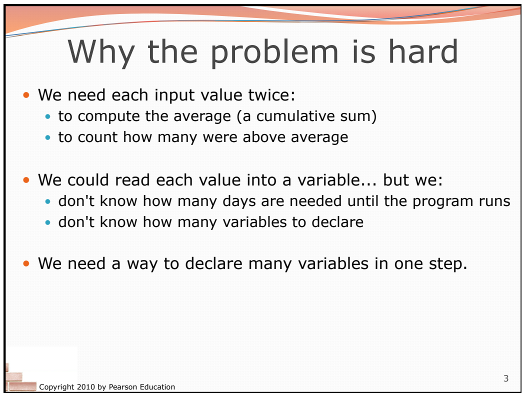
reading: 7.1



Can we solve this problem?

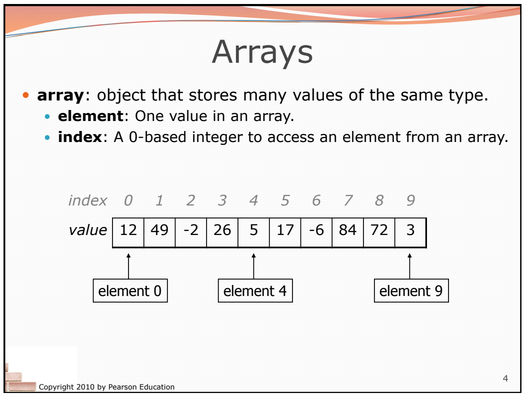
- Consider the following program (input underlined):

```
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.
```



Why the problem is hard

- We need each input value twice:
  - to compute the average (a cumulative sum)
  - to count how many were above average
- We could read each value into a variable... but we:
  - don't know how many days are needed until the program runs
  - don't know how many variables to declare
- We need a way to declare many variables in one step.

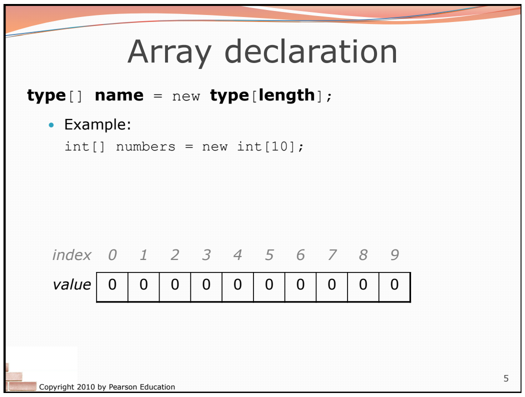


Arrays

- **array**: 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.

index	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
value	12	49	-2	26	5	17	-6	84	72	3

element 0
element 4
element 9



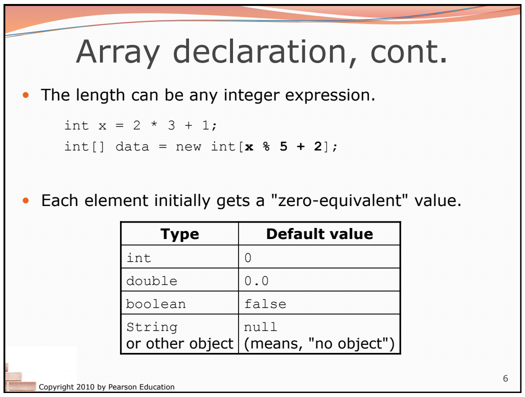
Array declaration

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

- Example:

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

index	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
value	0	0	0	0	0	0	0	0	0	0



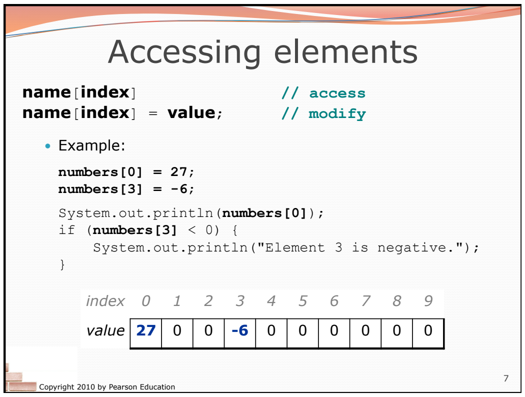
Array declaration, cont.

- The length can be any integer expression.

```
int x = 2 * 3 + 1;
int[] data = new int[x % 5 + 2];
```

- Each element initially gets a "zero-equivalent" value.

Type	Default value
`int`	`0`
`double`	`0.0`
`boolean`	`false`
`String` or other object	`null` (means, "no object")



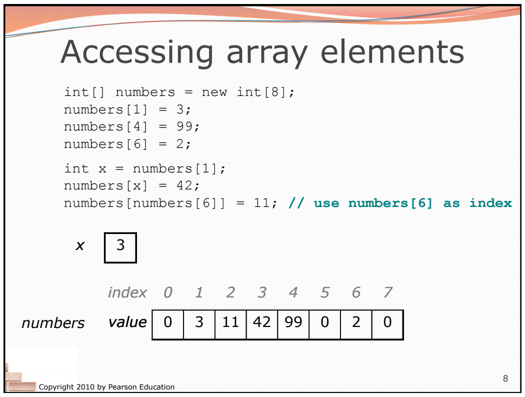
Accessing elements

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

- Example:

```
numbers[0] = 27;
numbers[3] = -6;

System.out.println(numbers[0]);
if (numbers[3] < 0) {
    System.out.println("Element 3 is negative.");
}
```

index	0	1	2	3	4	5	6	7	8	9
value	**27**	0	0	**-6**	0	0	0	0	0	0

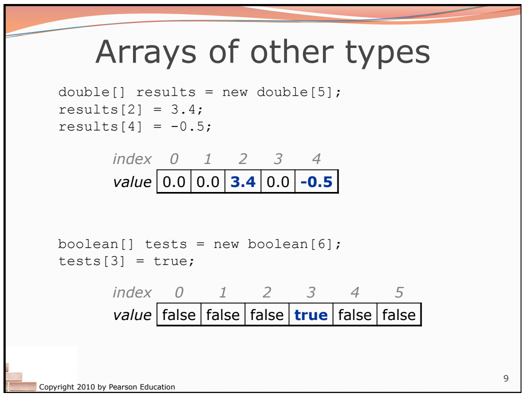


Accessing array elements

```
int[] numbers = new int[8];
numbers[1] = 3;
numbers[4] = 99;
numbers[6] = 2;

int x = numbers[1];
numbers[x] = 42;
numbers[numbers[6]] = 11; // use numbers[6] as index
```

x	3

	index	0	1	2	3	4	5	6	7
numbers	value	0	3	11	42	99	0	2	0



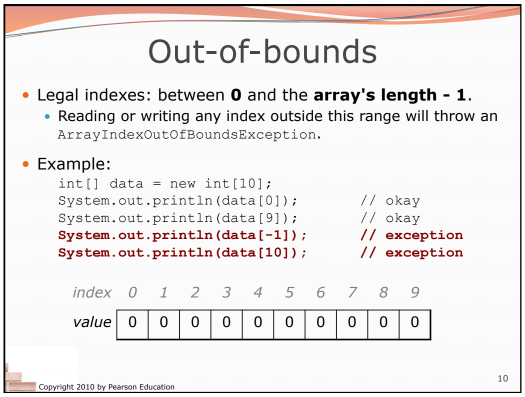
Arrays of other types

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

index	0	1	2	3	4
value	0.0	0.0	**3.4**	0.0	**-0.5**

```
boolean[] tests = new boolean[6];
tests[3] = true;
```

index	0	1	2	3	4	5
value	false	false	false	**true**	false	false

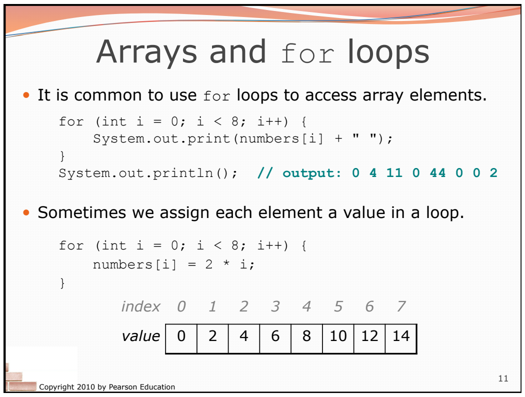


Out-of-bounds

- Legal indexes: between **0** and the **array's length - 1**.
  - Reading or writing any index outside this range will throw an `ArrayIndexOutOfBoundsException`.
- Example:

```
int[] data = new int[10];
System.out.println(data[0]);       // okay
System.out.println(data[9]);       // okay
System.out.println(data[-1]);      // exception
System.out.println(data[10]);      // exception
```

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



Arrays and `for` loops

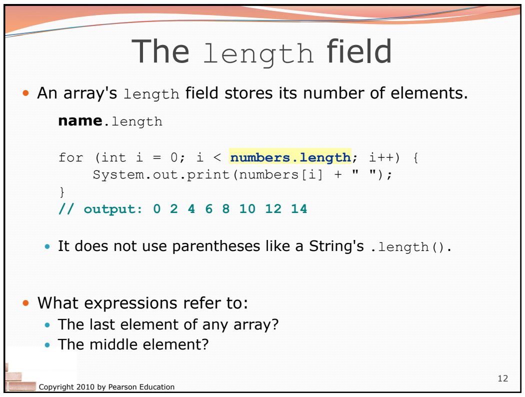
- It is common to use `for` loops to access array elements.

```
for (int i = 0; i < 8; i++) {
    System.out.print(numbers[i] + " ");
}
System.out.println();  // output: 0 4 11 0 44 0 0 2
```

- 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



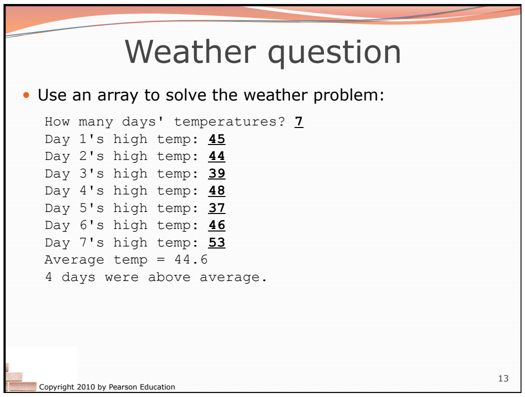
The `length` field

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

```
name.length

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

  - It does not use parentheses like a String's `.length()`.

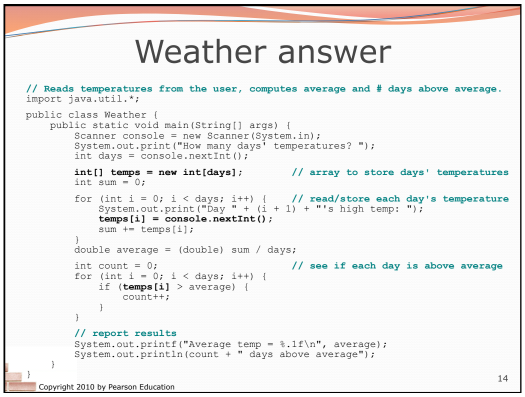
- What expressions refer to:
  - The last element of any array?
  - The middle element?



Weather question

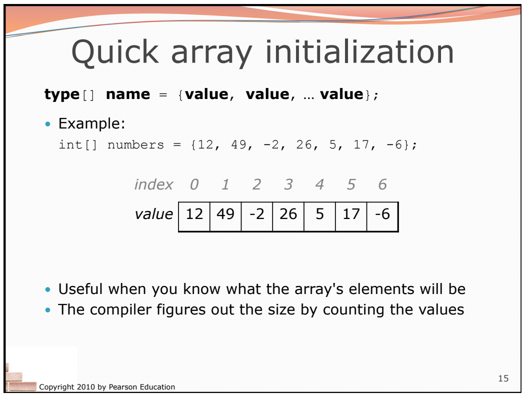
- Use an array to solve the weather problem:

```
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.
```



Weather answer

```
// Reads temperatures from the user, computes average and # days above average.
import java.util.*;

public class Weather {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        System.out.print("How many days' temperatures? ");
        int days = console.nextInt();

        int[] temps = new int[days];        // array to store days' temperatures
        int sum = 0;

        for (int i = 0; i < days; i++) {    // read/store each day's temperature
            System.out.print("Day " + (i + 1) + "'s high temp: ");
            temps[i] = console.nextInt();
            sum += temps[i];
        }
        double average = (double) sum / days;

        int count = 0;                      // see if each day is above average
        for (int i = 0; i < days; i++) {
            if (temps[i] > average) {
                count++;
            }
        }

        // report results
        System.out.printf("Average temp = %.1f\n", average);
        System.out.println(count + " days above average");
    }
}
```



Quick array initialization

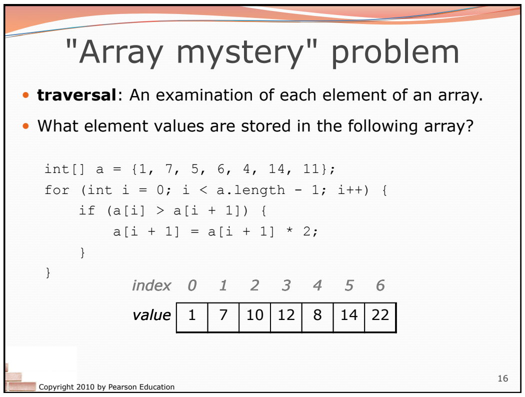
type`[]` **name** `= {`**value**`,` **value**`, …` **value**`};`

- 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

- Useful when you know what the array's elements will be
- The compiler figures out the size by counting the values

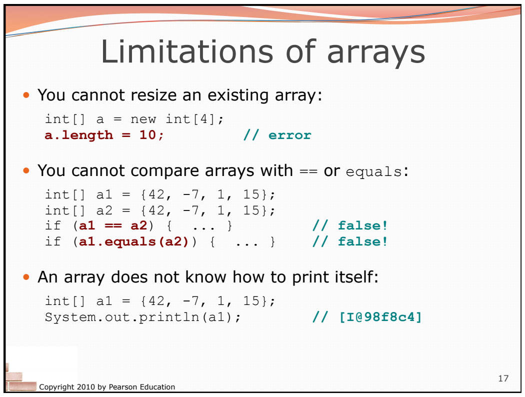


"Array mystery" problem

- **traversal**: An examination of each element of an array.
- What element values are stored in the following 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;
    }
}
```

index	0	1	2	3	4	5	6
value	1	7	10	12	8	14	22



Limitations of arrays

- You cannot resize an existing array:

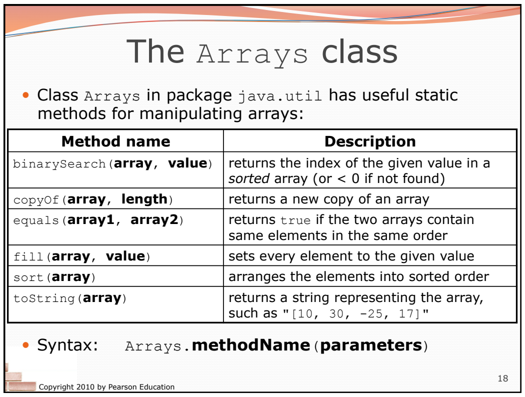
```
int[] a = new int[4];
a.length = 10;          // error
```

- 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:

```
int[] a1 = {42, -7, 1, 15};
System.out.println(a1);         // [I@98f8c4]
```

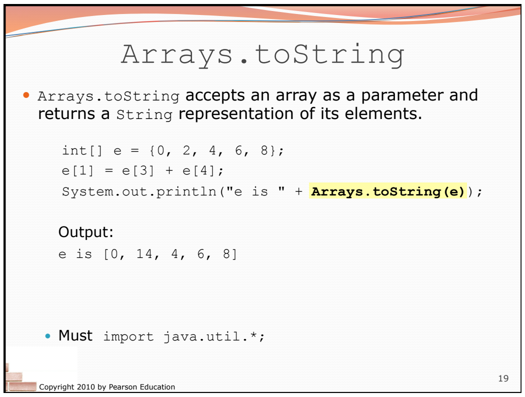


The `Arrays` class

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

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

- Syntax: `Arrays.`**methodName**`(`**parameters**`)`



Arrays.toString

- `Arrays.toString` accepts an array as a parameter and returns a `String` representation of its elements.

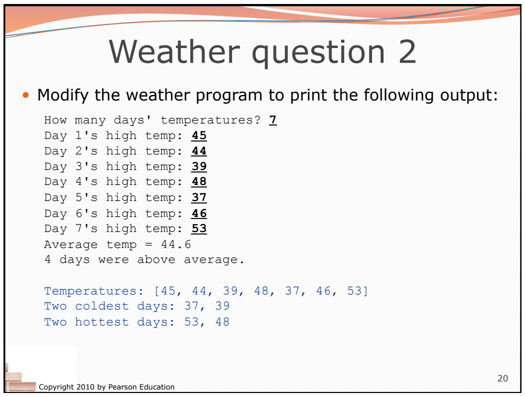
```
int[] e = {0, 2, 4, 6, 8};
e[1] = e[3] + e[4];
System.out.println("e is " + Arrays.toString(e));
```

Output:

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

- Must `import java.util.*;`

19

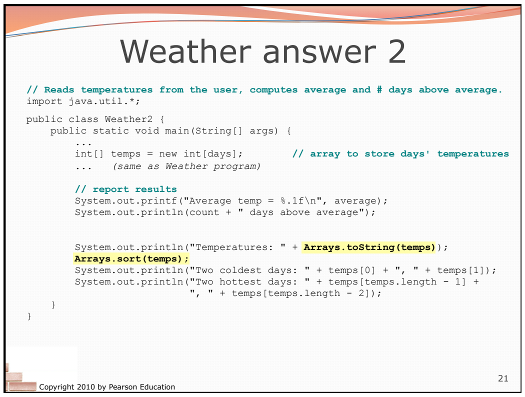


Weather question 2

- Modify the weather program to print the following output:

```
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.

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

20



Weather answer 2

```
// Reads temperatures from the user, computes average and # days above average.
import java.util.*;

public class Weather2 {
    public static void main(String[] args) {
        ...
        int[] temps = new int[days];        // array to store days' temperatures
        ...    (same as Weather program)

        // report results
        System.out.printf("Average temp = %.1f\n", average);
        System.out.println(count + " days above average");


        System.out.println("Temperatures: " + Arrays.toString(temps));
        Arrays.sort(temps);
        System.out.println("Two coldest days: " + temps[0] + ", " + temps[1]);
        System.out.println("Two hottest days: " + temps[temps.length - 1] +
                           ", " + temps[temps.length - 2]);
    }
}
```

21