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 determines the length 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.Arrays;`

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
```

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]);
    }
}
```