



Module 1

Arrays

Arrays

Arrays are a collection of variables of the same data type. To declare an array, we can use the following:

```
int[] arr1; //declaration form #1  
arr1 = new int[100]; //instantiation
```

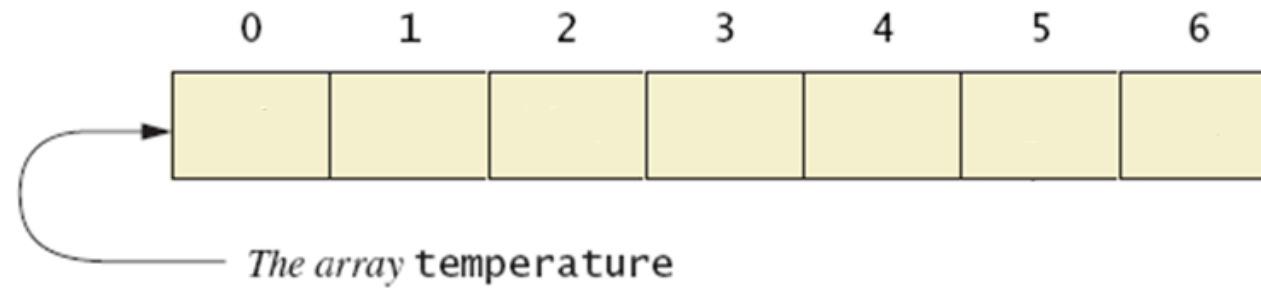
```
int arr2[]; //declaration form #2  
arr2 = new int[100]; //instantiation
```

```
int[] arr3 = new int[20]; //declaration and instantiation form #3
```

```
int arr4[] = new int[20]; //declaration and instantiation form #4
```

Visualizing Arrays

```
double[] temperature = new double[7];
```



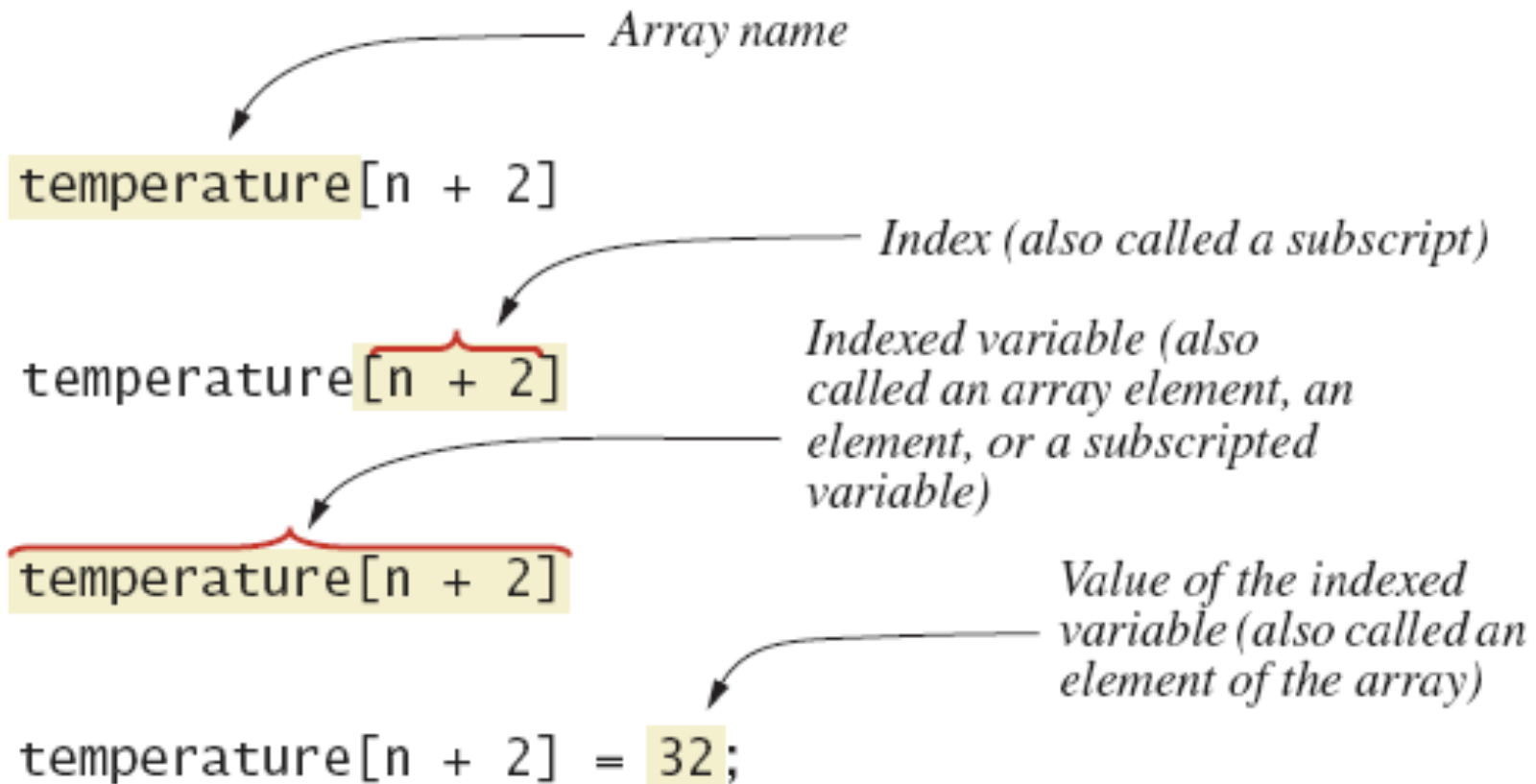
Syntax

All array positions have to be of the same type. We cannot mix Strings, integers or doubles on the same array.





```
data_type[] array_name = new data_type[array_size];  
data_type array_name[] = new data_type[array_size];
```

The array size needs to be a positive number greater than or equal to 0.

Nomenclature



Brackets

Brackets			
			
Round brackets	Square brackets	Curly brackets	Angle brackets
or	or	or	or
parentheses	brackets	braces	chevrons

Accessing array positions

To access the element of an array, we use square brackets [].

The first position of an array will always be index 0, and the last one will be `array_name.length - 1`.

```
double[] temperature = new double[7];
temperature[0] = 6.9; //first element index 0
temperature[1] = 30;
temperature[2] = 25.7;
temperature[3] = 26;
temperature[4] = 34;
temperature[5] = 31.5;
temperature[6] = 29; //last element index (temperature.length-1)
//temperature[temperature.length-1] = 29;
```

Accessing array positions

We can initialize an array in two ways:

1. Instantiating the array with the `new` operator, then filling each position.
2. Implicit instantiation with all array values

1

```
double[] temperature = new double[7];  
temperature[0] = 6.9;  
temperature[1] = 30;  
temperature[2] = 25.7;  
temperature[3] = 26;  
temperature[4] = 34;  
temperature[5] = 31.5;  
temperature[6] = 29;
```

2

```
double[] temperature = {6.9, 30, 25.7, 26, 34, 31.5, 29};
```


Array size

To find the size of the array, we can use the **length** attribute.

The length of the array is a **final** attribute, it can never be changed.

```
double[] temperature = new double[7];  
int len = temperature.length;  
System.out.println(len); //prints 7
```

The last element of an array will always be **length - 1**.

```
temperature[temperature.length - 1] = 29;
```

Array size

What would be the size of the following arrays?

```
double[] array1 = new double[10];  
double[] array2 = new double[0];  
double[] array3;
```

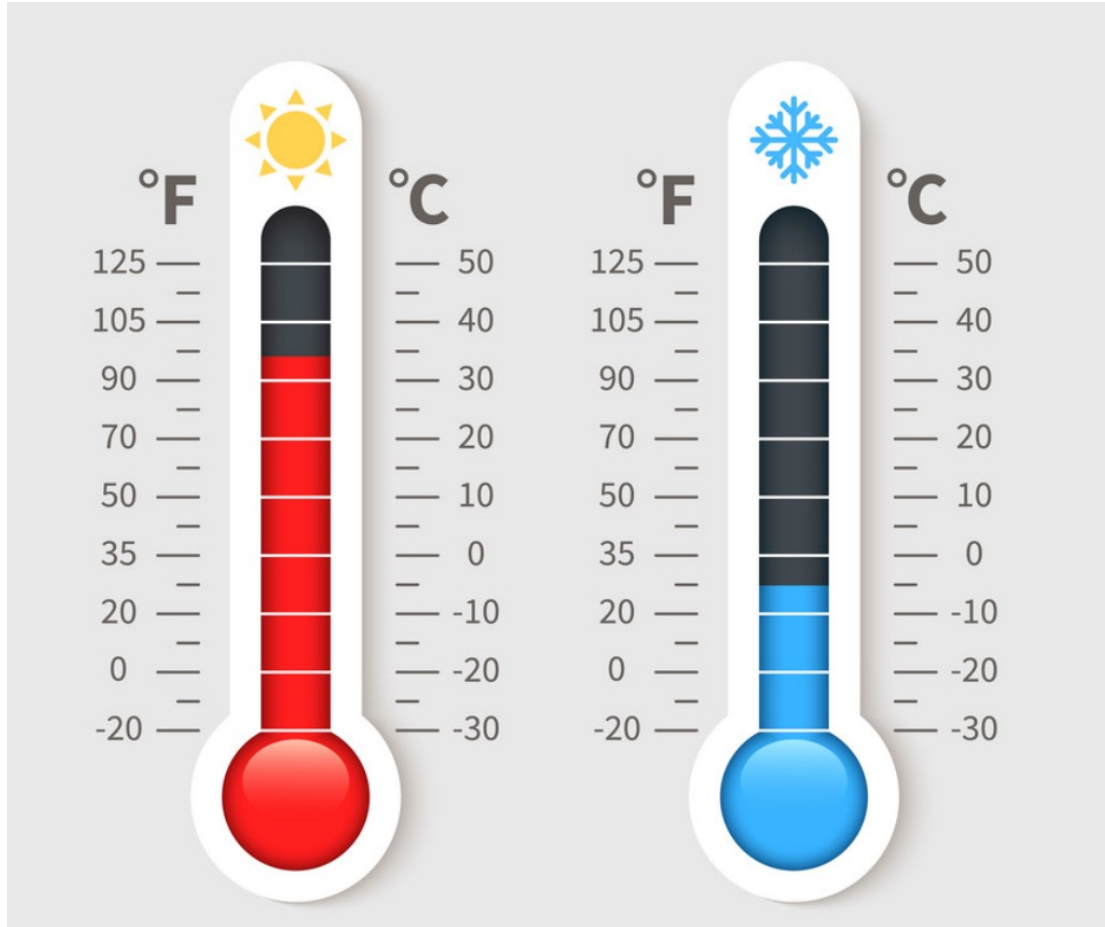
Answer:

array1 has a size of 10
array2 has a size of 0
array3 is null, it has no size



0 vs NULL

Exercise!



Build a program that can read 7 temperatures from the keyboard. Then it should calculate the average temperature and show which temperatures are above and below the average.



```
import java.util.*;

public class ArrayOfTemperatures {
    public static void main(String[] args) throws Exception {
        double[] temperatures = new double[7];
        Scanner keyboard = new Scanner(System.in);
        System.out.println("Enter 7 temperatures");

        double sum = 0;
        for(int i=0; i<temperatures.length; i++){
            temperatures[i] = keyboard.nextDouble();
            sum = sum + temperatures[i];
        }

        double average = sum / 7;
        System.out.println("The average is: " + average);
        System.out.println("The temperatures are:");
        for(int i=0; i<temperatures.length; i++){
            if (temperatures[i] > average){
                System.out.println(temperatures[i] + " is above average.");
            }
            if (temperatures[i] < average){
                System.out.println(temperatures[i] + " is below average.");
            }
            if (temperatures[i] == average){
                System.out.println(temperatures[i] + " is the average.");
            }
        }
        keyboard.close();
    }
}
```

Arrays are objects

Arrays are **objects**, which means that when we reference it we are actually trying to read the memory location where the object is stored.

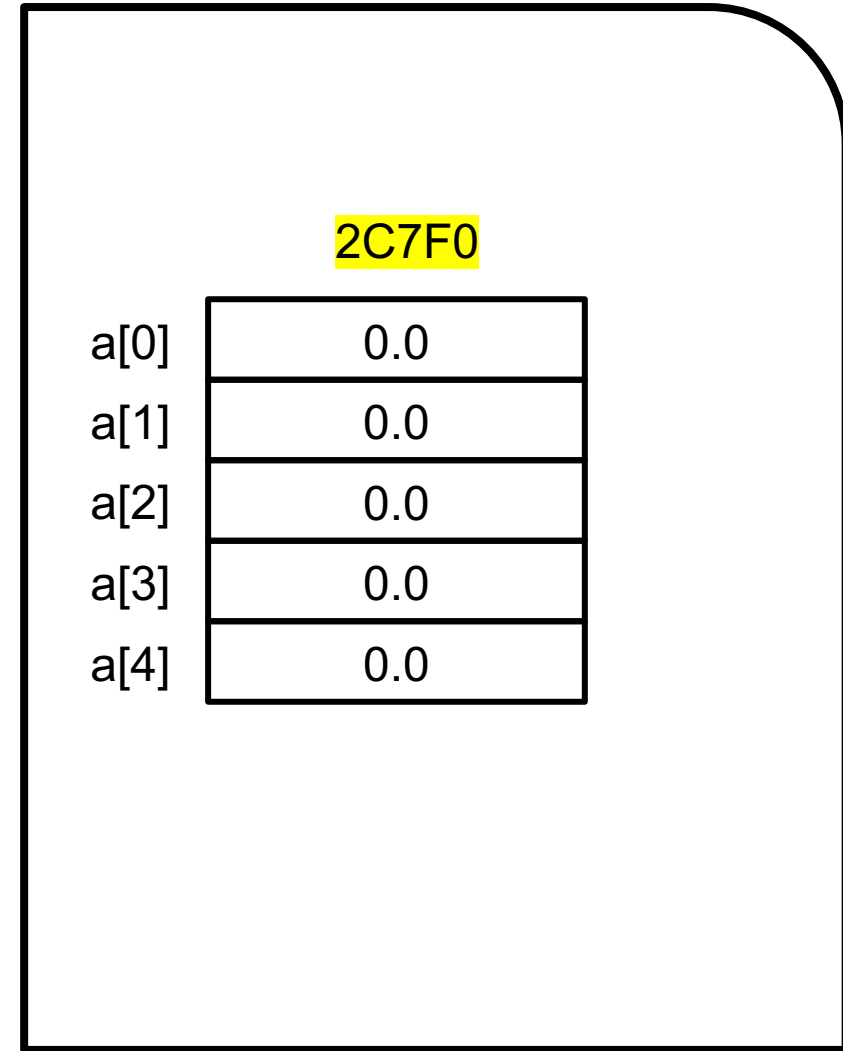
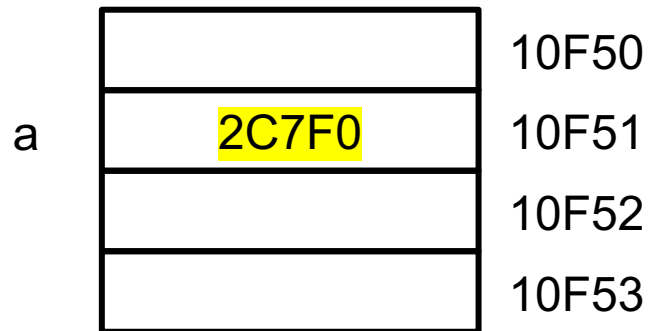
Let's look at an example:

```
double[] a;
```



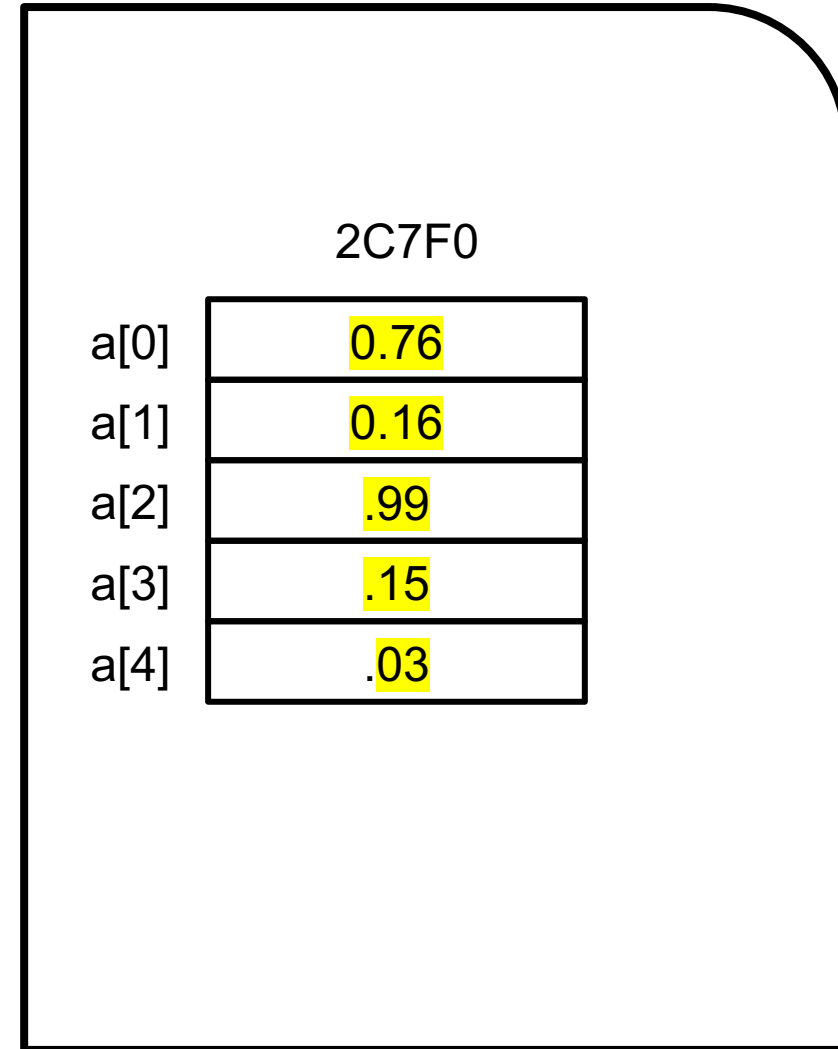
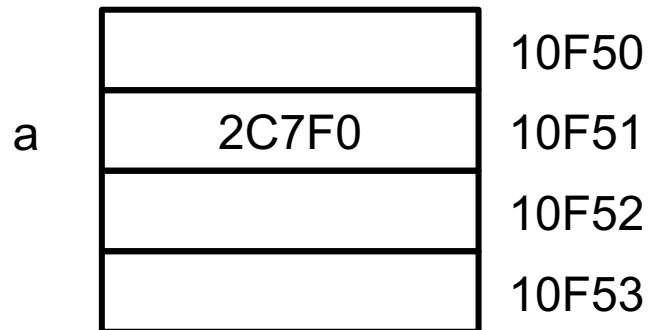
When declaring variable `a`, and it is stored in `10F51`

```
double[] a;  
a = new double[5];
```



We create a new array of size 5.
The array starts in memory position
2C7F0.

```
double[] a;  
a = new double[5];  
a[0] = Math.random();  
a[1] = Math.random();  
a[2] = Math.random();  
a[3] = Math.random();  
a[4] = Math.random();
```



Fill array with random values

MORSE CODE

International Morse Code

1. The length of a dot is one unit.
2. A dash is three units.
3. The space between parts of the same letter is one unit.
4. The space between letters is three units.
5. The space between words is seven units.

A ● —
B — ● ● ●
C — ● — ●
D — ● ●
E ●
F ● ● — ●
G — — ●
H ● ● ● ●
I ● ●
J ● — — —
K — ● —
L ● — ● ●
M — —
N — ●
O — — —
P ● — — ●
Q — — ● —
R ● — ●
S ● ● ●
T —

U ● ● —
V ● ● ● —
W ● — —
X — ● ● —
Y — ● — —
Z — — ● ●

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

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

Diseña un programa que lea un número entero positivo del teclado, y lo convierta a clave morse.

Cada dígito distinto deberá estar separado por un espacio.

Por ejemplo:

El número 13, resultaría en:

.---- ...--