ARRAY BASICS

Ch7.1

Array Basics: Outline

- Creating and Accessing Arrays
- Array Details
- The Instance Variable length
- More About Array Indices
- Initializing Arrays

Creating and Accessing Arrays

- An array is a special kind of object
- Think of as collection of variables of same type that are stored in adjacent memory locations.
- Creating an array with 7 variables of type double

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

- To access an element use
  - The name of the array
  - An index number enclosed in square brackets
- Array indices begin at zero

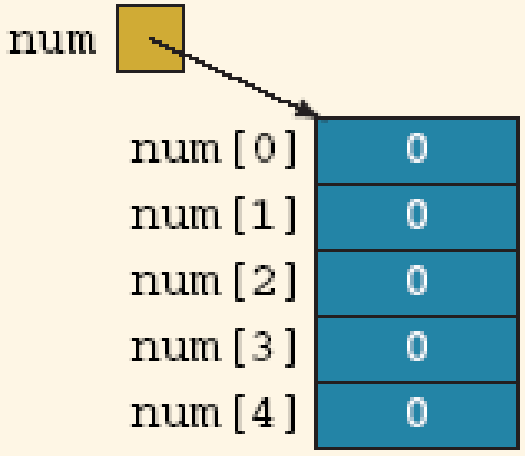
Array declaration - example

- Declaring the array `num` :

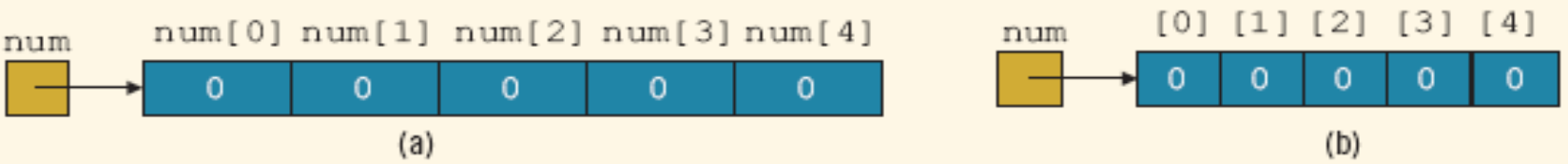
```
int[] num = new int[5];
```

When an array is instantiated, Java **automatically** initializes its elements to their default values.

numeric arrays are initialized to 0,
char arrays are initialized to the null character, which is '\u0000',
boolean arrays are initialized to false.

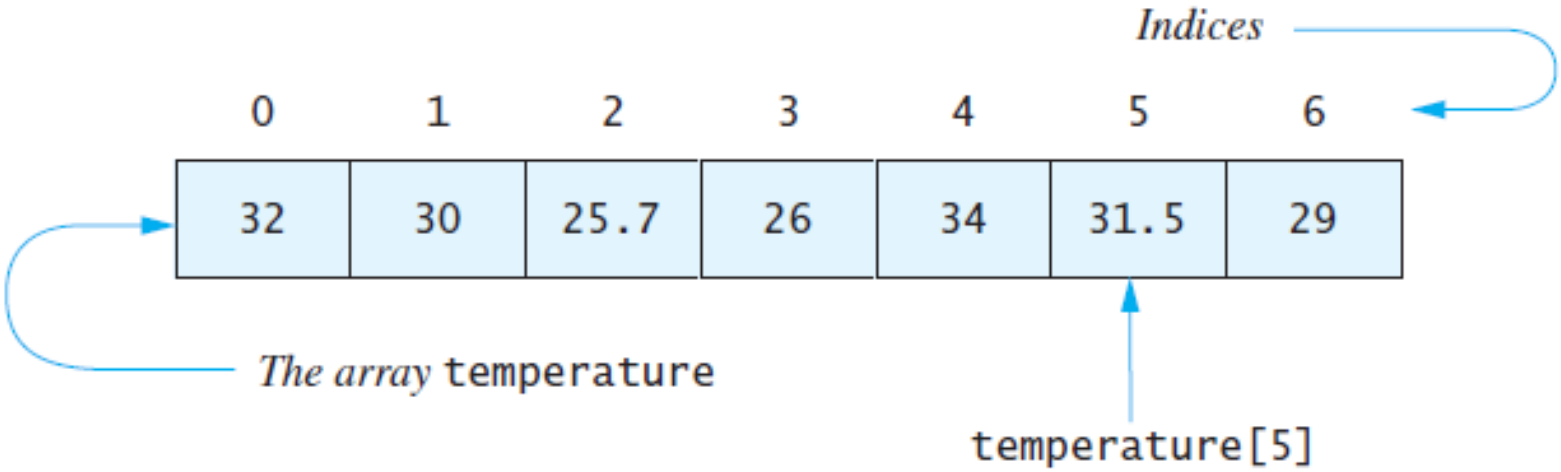


To save space, we also draw an array, as shown in Figures (a) and b).



Creating and Accessing Arrays

FIGURE 7.1 A Common Way to Visualize an Array



- Note sample program, listing 7.1
  `class ArrayOfTemperatures`

LISTING 7.1 An Array of Temperatures *(part 1 of 2)*

```
/**
Reads 7 temperatures from the user and shows which are above
and which are below the average of the 7 temperatures.
*/
import java.util.Scanner;

public class ArrayOfTemperatures
{
    public static void
    {
        double[] temper

        // Read tempera
        Scanner keyboar
        System.out.prin
        double sum = 0;
        for (int index
        {
            temperature
            sum = sum +
        }
        double average = sum / 7;
        System.out.println("The average temperature is " +
                           average);

        // Display each temperature and its relation to the average:
        System.out.println("The temperatures are");
        for (int index = 0; index < 7; index++)
        {
            if (temperature[index] < average)
                System.out.println(temperature[index] +
                                   " below average");
            else if (temperature[index] > average)
                System.out.println(temperature[index] +
                                   " above average");
            else //temperature[index] == average
                System.out.println(temperature[index] +
                                   " the average");
        }

        System.out.println("Have a nice week.");
    }
}
```

Creating and Accessing Arrays

```
Enter 7 temperatures:
32
30
25.7
26
34
31.5
29
The average temperature is 29.7428
The temperatures are
32.0 above average
30.0 above average
25.7 below average
26.0 below average
34.0 above average
31.5 above average
29.0 below average
Have a nice week.
```

Sample screen output

Array Details

- Syntax for declaring an array with **`new`**

```
Base_Type[] Array_Name = new Base_Type[Length];
```

- The number of elements in an array is its length
- The type of the array elements is the array's base type

Square Brackets [ ] with Arrays

- They are used as follows:
  - With a data type when declaring an array

```
int [ ] pressure;
```

  - To enclose an integer expression to declare the length of the array

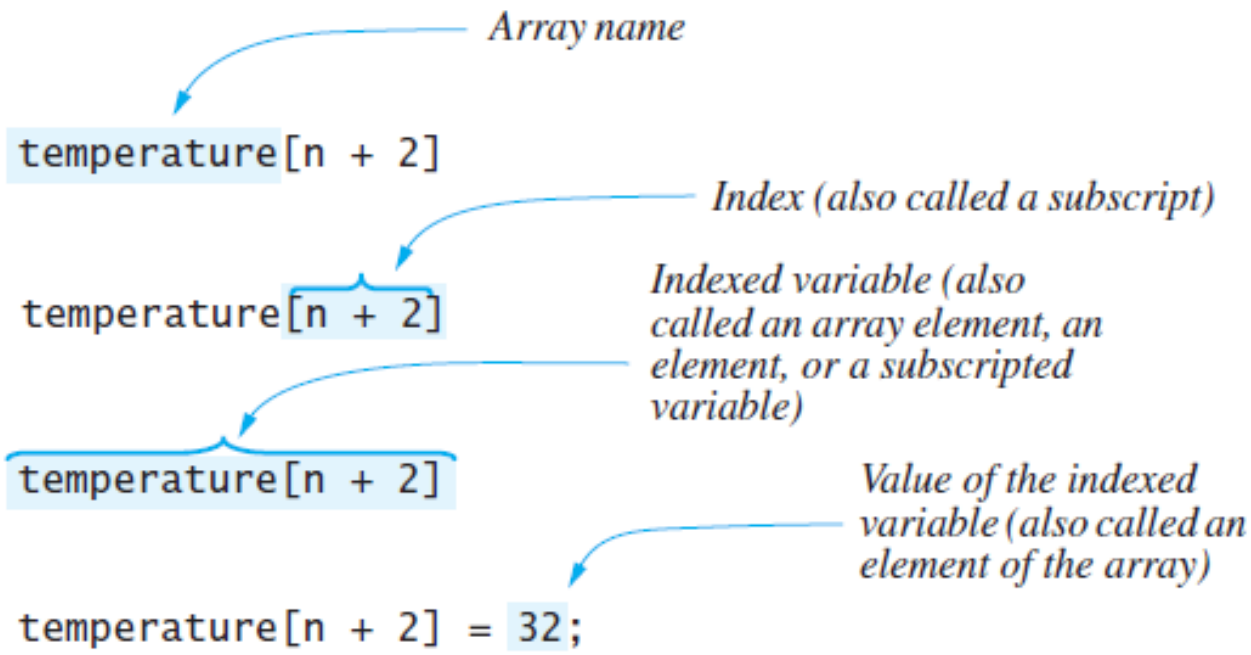
```
pressure = new int [100];
```

  - To name an indexed value of the array

```
pressure[3] = keyboard.nextInt();
```

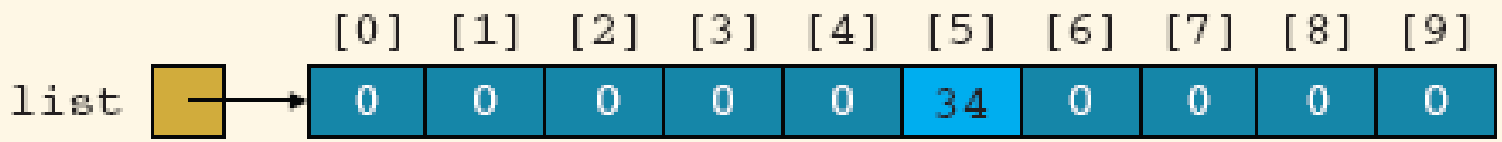
Array Details

FIGURE 7.2 Array Terminology

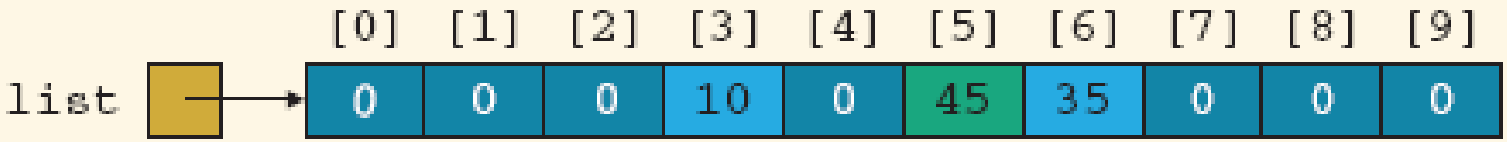


ACCESSING ARRAY ELEMENTS

```
int[] list = new int[10];
list[5] = 34;
```




```
list[3] = 10;
list[6] = 35;
list[5] = list[3] + list[6];
```



ACCESSING ARRAY ELEMENTS - Example

- Consider the following declarations and their reflection in the memory:

1
```
double[]  list = new double[10];
```

	list[0]	list[1]	list[2]	list[3]	list[4]	list[5]	list[6]	list[7]	list[8]	list[9]
list→	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

2
3
```
int  i = 2;
list[2*i-1] = 46.0;          // list[??] = 46.0
```

	list[0]	list[1]	list[2]	list[3]	list[4]	list[5]	list[6]	list[7]	list[8]	list[9]
list→	0.0	0.0	0.0	**46.0**	0.0	0.0	0.0	0.0	0.0	0.0

4
```
list[2*i+1] = 20.0;          // list[??] = 20.0
```

	list[0]	list[1]	list[2]	list[3]	list[4]	list[5]	list[6]	list[7]	list[8]	list[9]
list→	0.0	0.0	0.0	**46.0**	0.0	**20.0**	0.0	0.0	0.0	0.0

5
```
list[7] = list[3] + list[5];       // list[7] = 46.0 + 20.0 = 66.0
```

	list[0]	list[1]	list[2]	list[3]	list[4]	list[5]	list[6]	list[7]	list[8]	list[9]
list→	0.0	0.0	0.0	**46.0**	0.0	**20.0**	0.0	**66.0**	0.0	0.0

ACCESSING ARRAY ELEMENTS - Example

5
```
list[7] = list[3] + list[5];        //list[7] = 46.0 + 20.0 = 66.0
```

	list[0]	list[1]	list[2]	list[3]	list[4]	list[5]	list[6]	list[7]	list[8]	list[9]
list→	0.0	0.0	0.0	**46.0**	0.0	**20.0**	0.0	**66.0**	0.0	0.0

- Note the difference to the following statements:

6
```
list[7] = list[3 + 5];              // list[7] = ??
```

	list[0]	list[1]	list[2]	list[3]	list[4]	list[5]	list[6]	list[7]	list[8]	list[9]
list→	0.0	0.0	0.0	**46.0**	0.0	**20.0**	0.0	**0.0**	0.0	0.0

7
```
list[3+5] = list[3] + list[5];      // list[??] = ?? + ??
```

	list[0]	list[1]	list[2]	list[3]	list[4]	list[5]	list[6]	list[7]	list[8]	list[9]
list→	0.0	0.0	0.0	**46.0**	0.0	**20.0**	0.0	**0.0**	**66.0**	0.0

Question

consider this array:

int list[ ] = new int[10];

	list[0]	list[1]	list[2]	list[3]	list[4]	list[5]	list[6]	list[7]	list[8]	list[9]
list→	5	7	3	0	9	10	2	3	6	1

list[list[2]] = 77;

list [1] = list[7] * list[7];

list [0] = list[list[8] -1];

The Instance Variable `length`

- As an object an array has only one public instance variable
  - Variable **`length`**
  - Contains the size of the array: the max number of elements the array can hold
  - It is final, value cannot be changed
- Note revised code, listing 7.2
  `class ArrayOfTemperatures2`

LISTING 7.2 An Array of Temperatures—Revised *(part 1 of 2)*

```
/**
Reads temperatures from the user and shows which are above
and which are below the average of all the temperatures.
*/
import java.util.Scanner;

public class ArrayOfTemperatures2
{
    public static void main(String[] args)
    {
        Scanner keyboard = new Scanner(System.in);
        System.out.println("How many temper
        int size = keyboard.nextInt( );
        double[] temperature = new double[s

        // Read temperatures and compute th
        System.out.println("Enter " + tempe
                           " temperatures:"
        double sum = 0;
        for (int index = 0; index < tempera
        {
            temperature[index] = keyboard.n
            sum = sum + temperature[index];
        }
        double average = sum / temperature.
        System.out.println("The average tem
                           average);
        // Display each temperature and its relation to the
        // average:
        System.out.println("The temperatures are");
        for (int index = 0; index < temperature.length; index++)
        {
            if (temperature[index] < average)
                System.out.println(temperature[index] +
                                   " below average");
            else if (temperature[index] > average)
                System.out.println(temperature[index] +
                                   " above average");
            else //temperature[index] == average
                System.out.println(temperature[index] +
                                   " the average");
        }

        System.out.println("Have a nice week.");
    }
```

The Instance Variable `length`

```
How many temperatures do you have?
3
Enter 3 temperatures:
32
26.5
27
The average temperature is 28.5
The temperatures are
32.0 above average
26.5 below average
27.0 below average
Have a nice week.
```

Sample screen output

More About Array Indices

- Index of first array element is 0
- Last valid Index is `arrayName.length - 1`
- Array indices must be within bounds to be valid
  - When program tries to access outside bounds, run time error occurs
- OK to "waste" element 0
  - Program easier to manage and understand
  - Yet, get used to using index 0

Initializing Arrays

- Possible to initialize at declaration time

```
double[] reading = {3.3, 15.8, 9.7};
```

- Also may use normal assignment statements
  - One at a time
  - In a loop

```
int[] count = new int[100];
for (int i = 0; i < 100; i++)
    count[i] = 0;
```

Initializing Arrays

- It is a common practice for a program to keep track of the number of filled elements in an array
- Simply store it in a variable, say **numOfElements**
- Then **numOfElements** will tell you the actual number of elements in the array.
- For example:

```
int numOfElements = 0;
numList[0] = 5;
numList[1] = 10;
numList[2] = 15;
numList[3] = 20;
numOfElements = 4;
System.out.print(n
```

How about this instead?

```
int numOfElements = 0;
numList[numOfElements++] = 5;
numList[numOfElements++] = 10;
numList[numOfElements++] = 15;
numList[numOfElements++] = 20;
System.out.print(numOfElements);
```

Recap: Specifying array size

- Array size can be specified using a constant

```
final int ARRAY_SIZE = 10;
int[] list = new int[ARRAY_SIZE];
```

- Array size can be specified during program execution time

```
int arraySize;

System.out.print("Enter the size of the array: ");
arraySize = console.nextInt();
System.out.println();

int[] list = new int[arraySize];
```

Recap: Specifying array size

- Array size can be specified during array declaration with initialization:

```
double[] sales = {12.25, 32.50, 16.90, 23, 45.68};
```

- The **initializer list** contains values, called **initial values**, that are placed between braces and separated by commas
- The size of the array will be equal to the number of values.
- In this example: `sales.length` will be equal to 5
- `sales` would be initialized as follows:
  `sales[0]= 12.25, sales[1]= 32.50, sales[2]= 16.90, sales[3]= 23.00,` and `sales[4]= 45.68`

- **Note:** If an array is declared and initialized simultaneously, we don't use the operator **`new`** to instantiate the array object

Note on Array Declaration

- What is the difference between the following declarations?

```
int      alpha[], beta;

int[ ]   gamma,   delta;
```

Remember

- The size of the array specified during instantiation cannot be changed.
- Sometimes, the programmer does not know how many elements will be needed.
- Therefore, it is common to specify a large size and use only the elements which were actually filled by the user.
- Assume that numOfElements is a variable that represents the number of actually filled elements within the array (numOfElements <= length)
- The value of numOfElements is used as a counter: it is initialized to zero, and then incremented with each new element filled in the array.