Chapter 9

Arrays

OBJECTIVES

After you have read and studied this chapter, you should be able to

- Manipulate a collection of data values using an array.
- Declare and use an array of primitive data types in writing a program.
- Declare and use an array of objects in writing a program.
- Describe how a two-dimensional array is implemented as an array of arrays.
- Use a MultiInputBox object from the javabook package to input an array of strings.
- Define a method that accepts an array as its parameter and a method that returns an array.
- Describe how the self-reference pointer works and use it in methods.

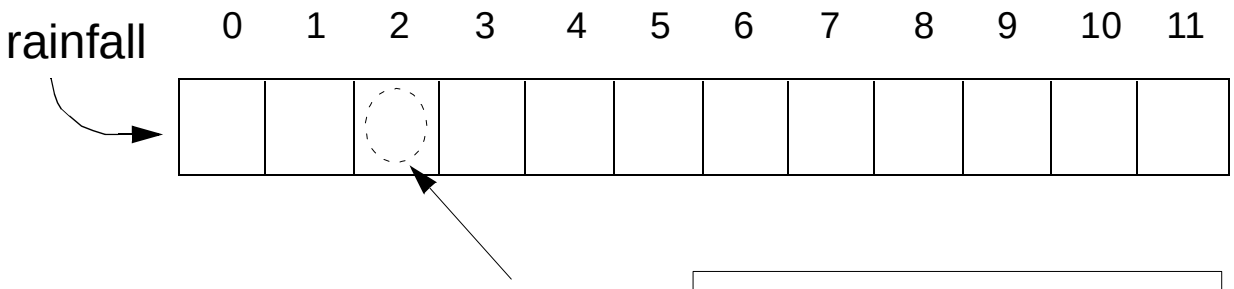
FIGURE 9.1 Monthly rainfall figures and their variation from the annual average.

Annual Average Rainfall: 15.03 mm

Month	Monthly Average	Variation from Annual Average
1	13.3	1.73
2	14.9	0.13
3	14.7	0.33
4	23.0	7.97
5	25.8	10.77
6	27.7	12.67
7	12.3	2.73
8	10.0	5.03
9	9.8	5.23
10	8.7	6.33
11	8.0	7.03
12	12.2	2.83

FIGURE 9.2 An array of 12 **float** values.

```
float[] rainfall = new float[12];
```



rainfall[2]

This is an indexed expression referring to the element at position #2, that is, the third element of the array.

FIGURE 9.3 An array of 12 **float** values after all 12 are assigned values.

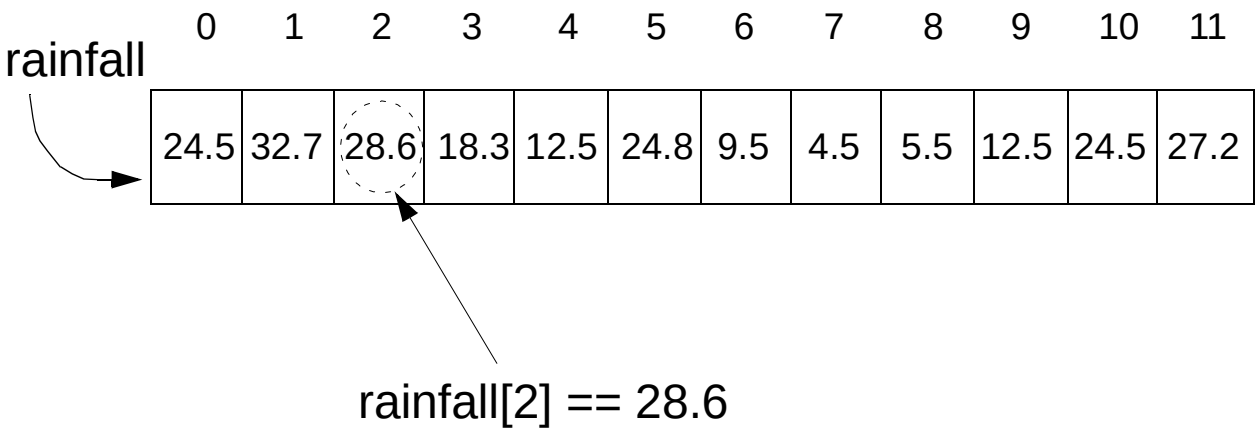


FIGURE 9.4 An array of **Person** objects after the array is created.

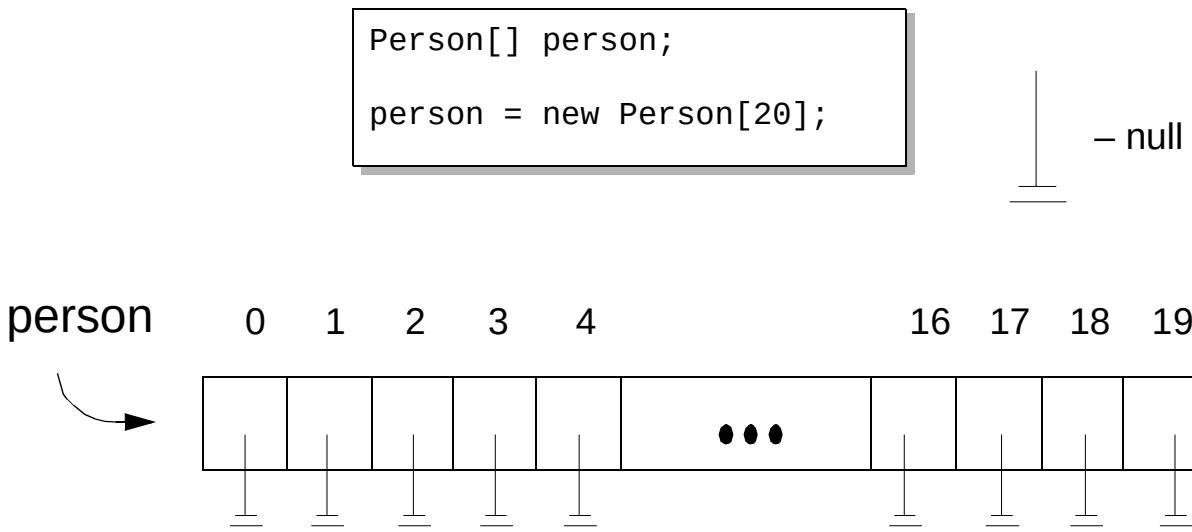


FIGURE 9.5 The person array with one **Person** object added to it.

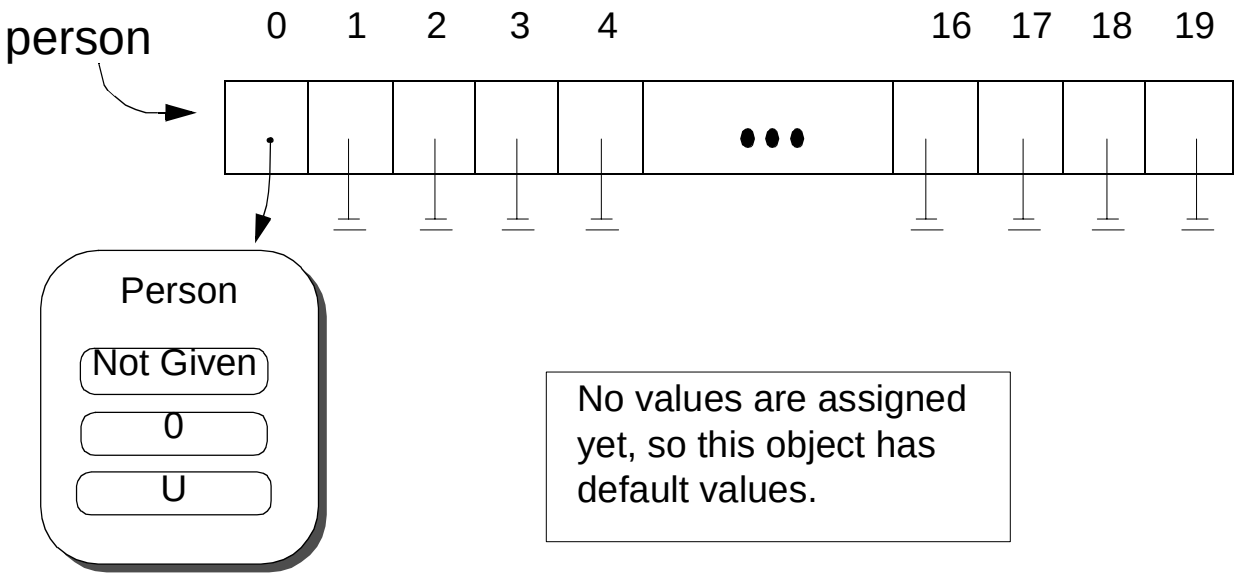


FIGURE 9.7 Approach 1 Deletion: Setting a reference to **null**. The array length is 4.

```
int delIdx = 1;
person[delIdx] = null;
```

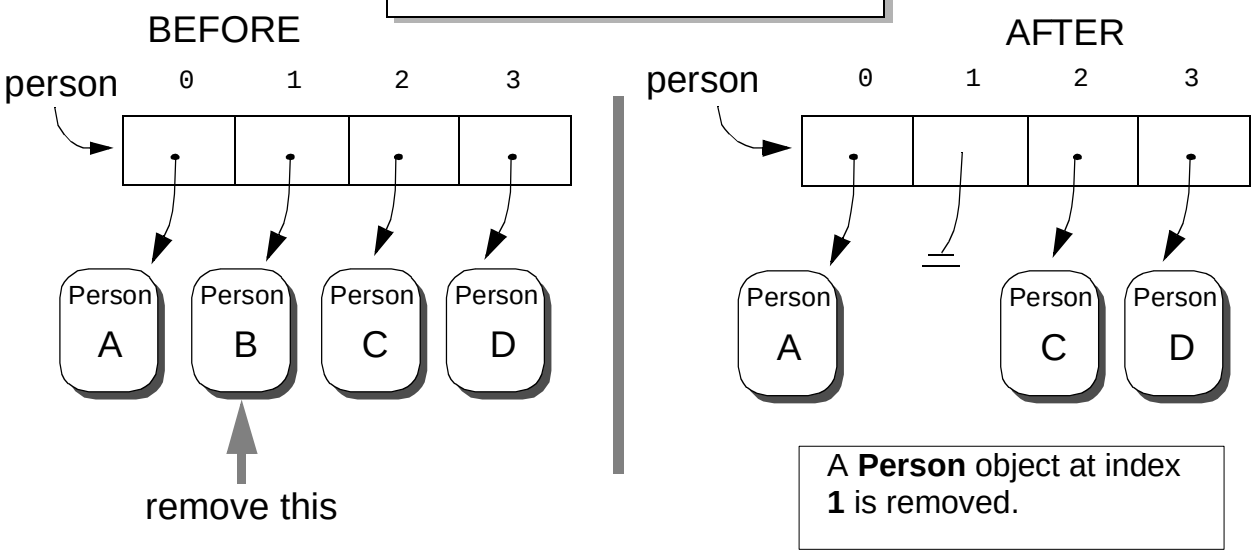


FIGURE 9.6 An array of **Person** objects with two **Person** variables.

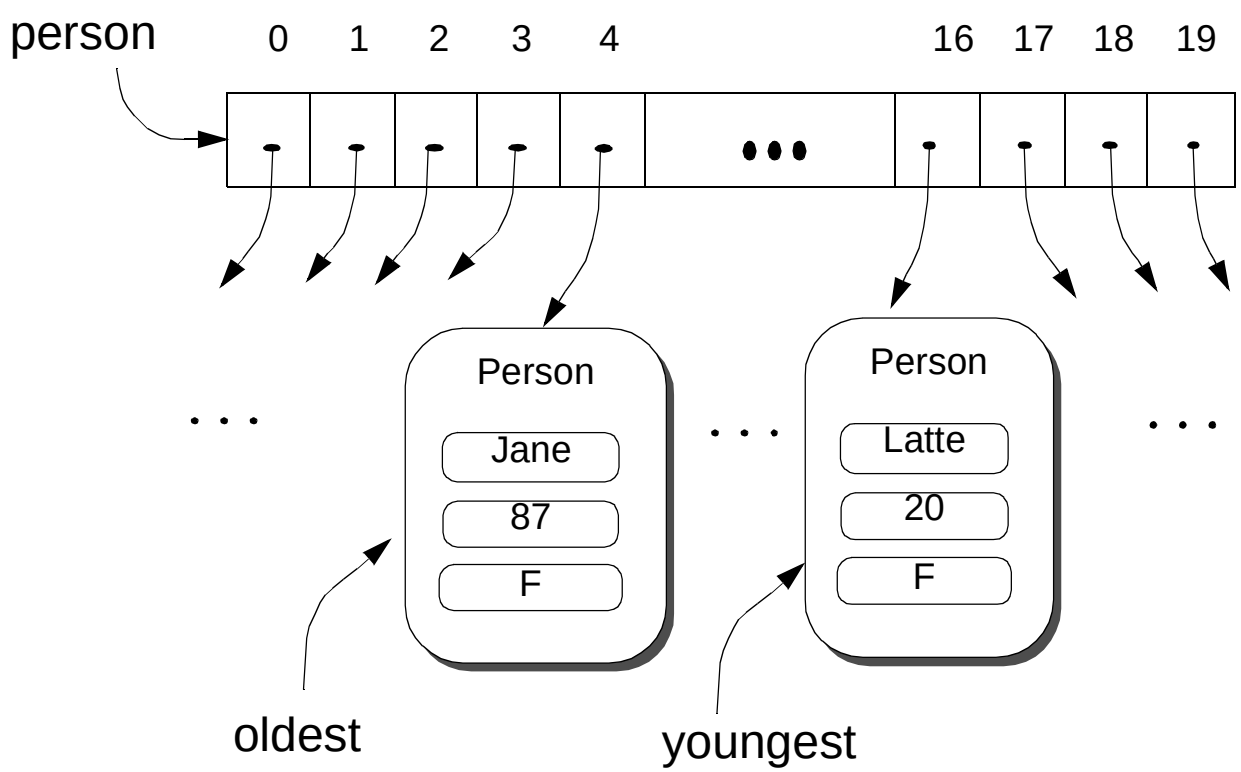


FIGURE 9.8 Approach 2 Deletion: Replace the removed element with the last element in the array. The array length is 4.

```
int delIdx = 1;
int last = 3;
person[delIdx] = person[last];
person[last] = null;
```

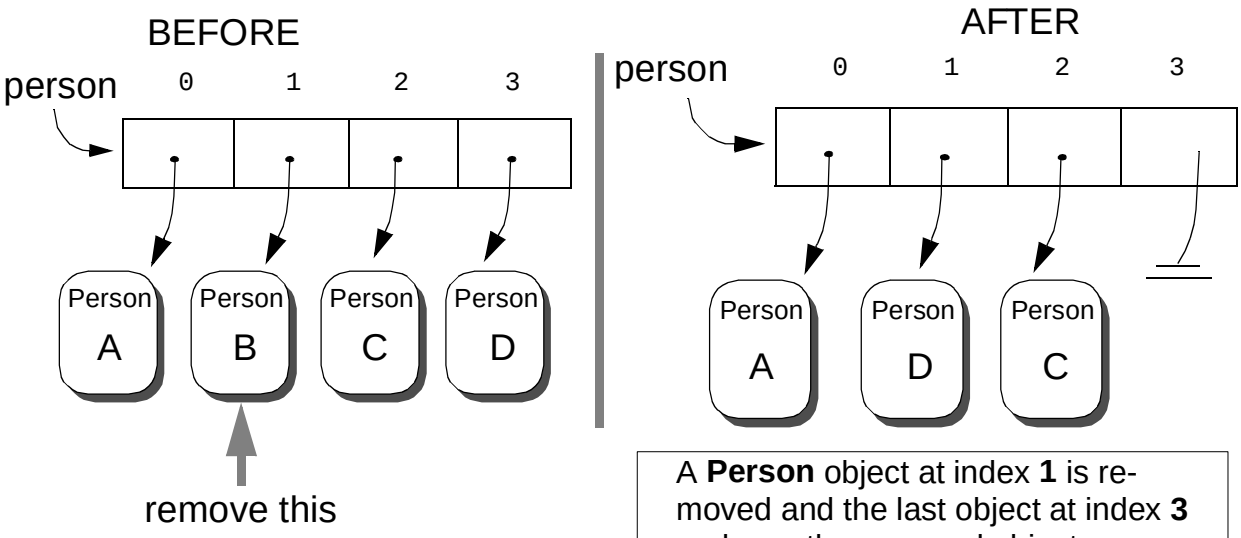


FIGURE 9.11 A **MultiInputBox** object (from the **javabook** package) to enter three values.

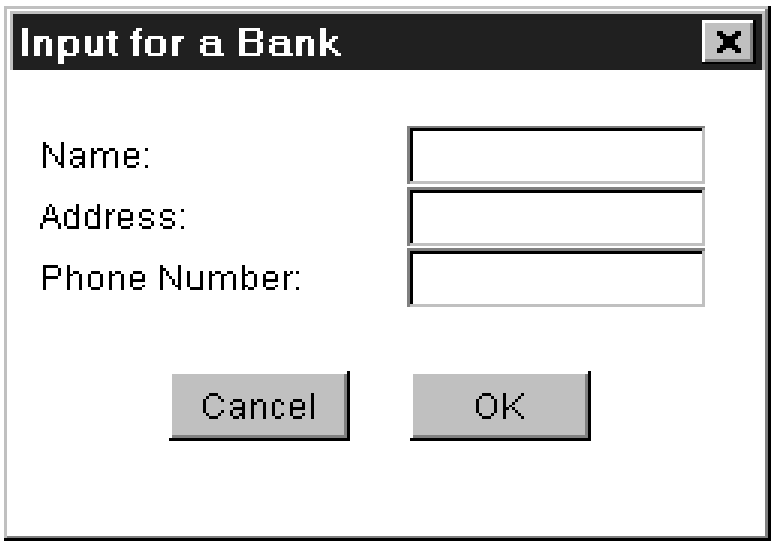


FIGURE 9.9 Passing an array to a method means we are passing a reference to an array. We are not passing the whole array.

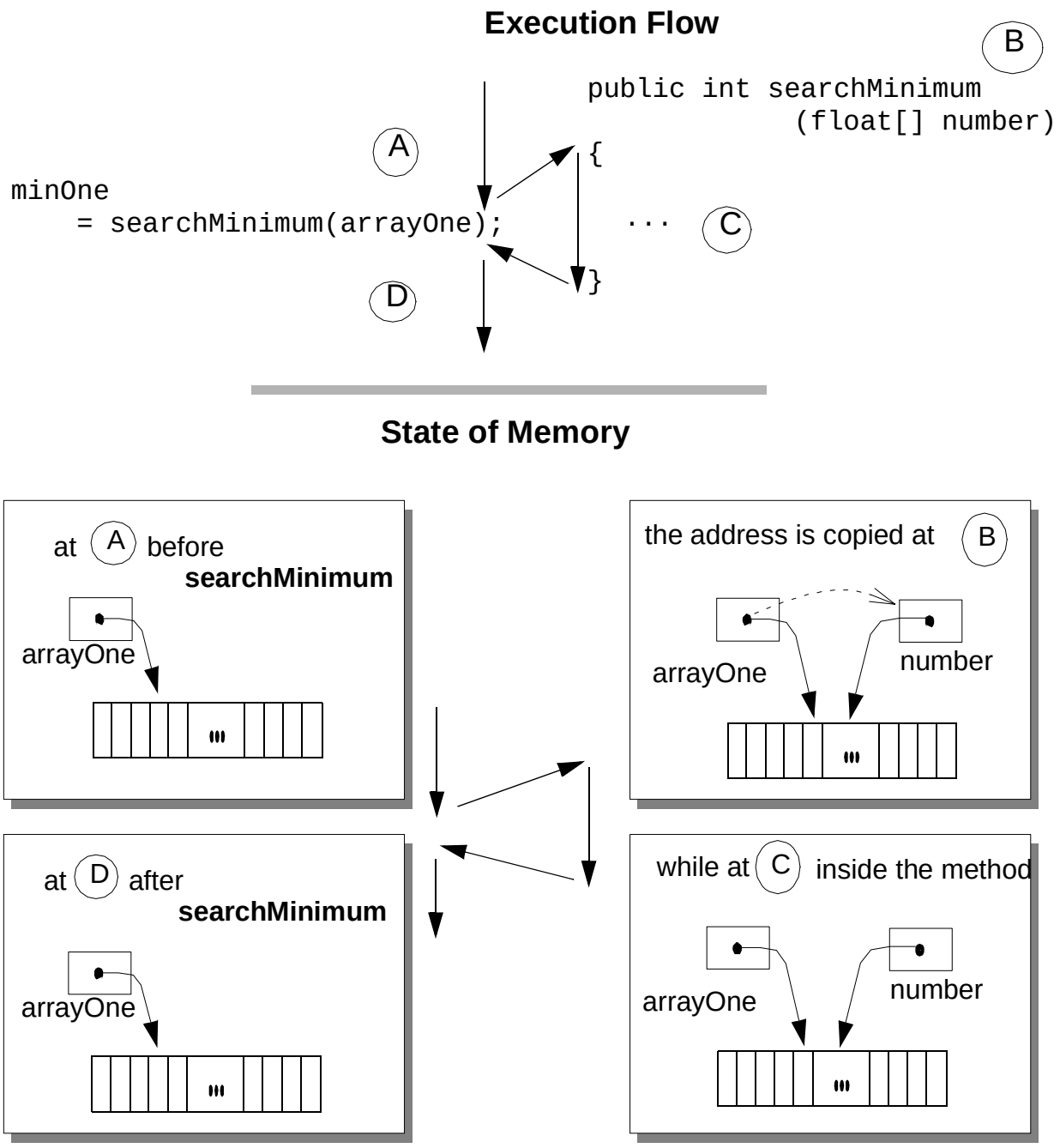


FIGURE 9.10 Effect of creating a local array and not returning it.

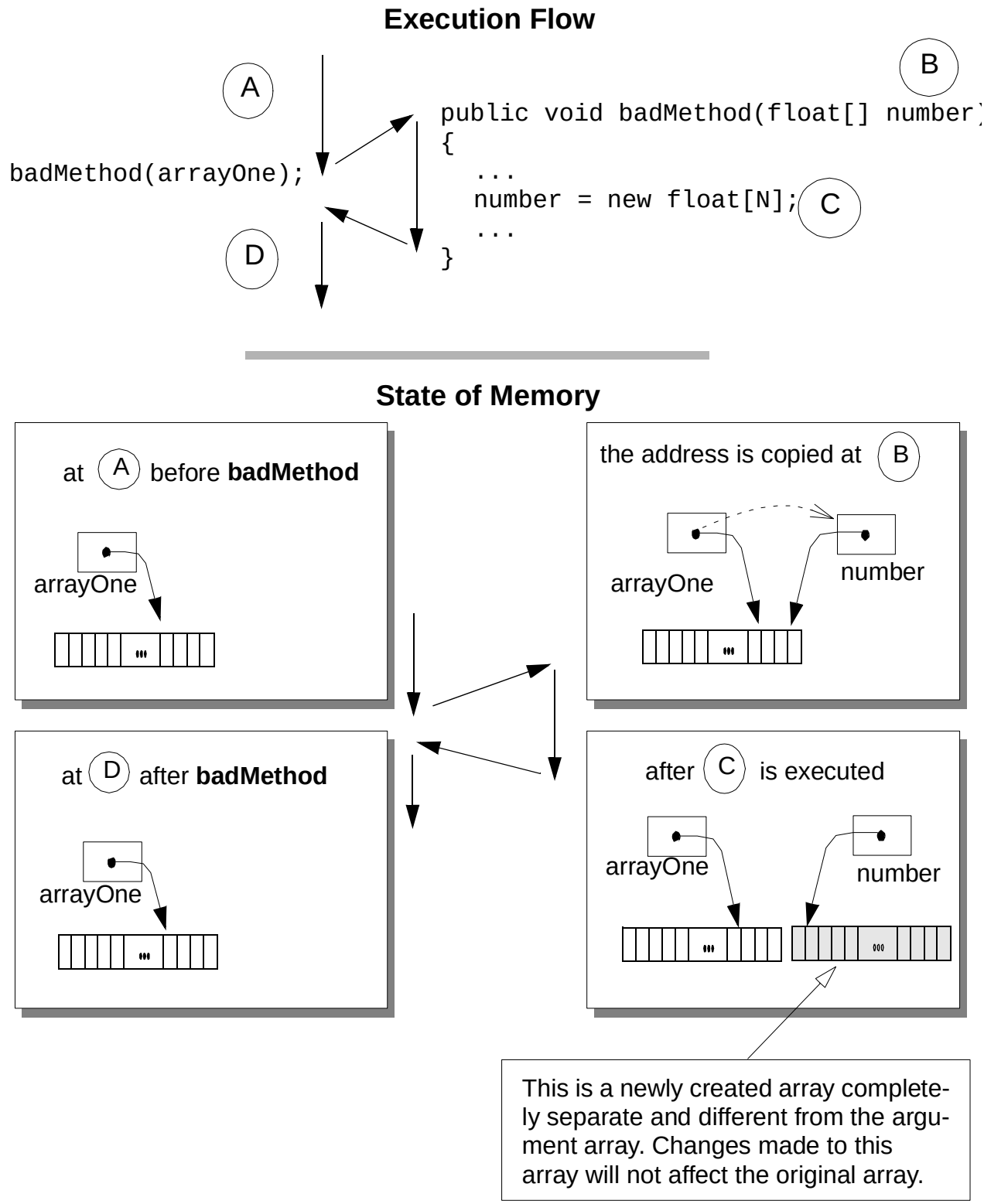


FIGURE 9.12 The answer array returned from **multiBox**.

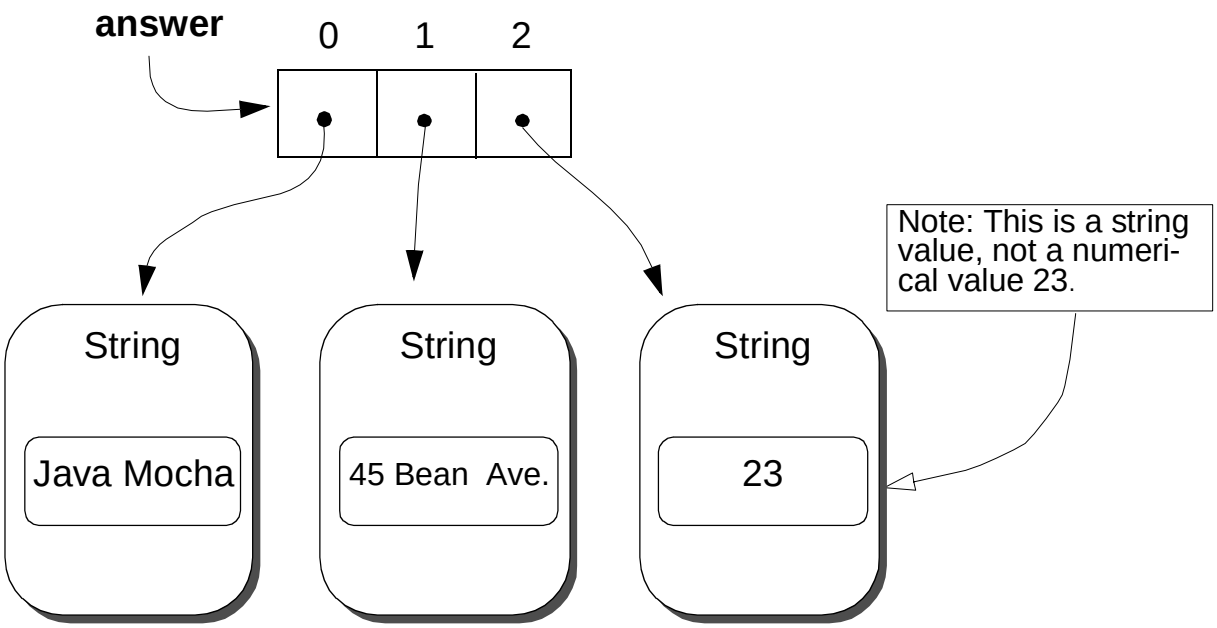


TABLE 9.1 A list of **MultiInputBox** methods.

CLASS:	**MultiInputBox**	
Method	**Argument**	**Description**
`<constructor>`	`MainWindow, int`	Creates a MultiInputBox object. The second argument specifies the number of labels.
`<constructor>`	`MainWindow, array of String`	Creates a MultiInputBox object. The second argument is an array of String for labels.
`setLabels`	`array of String`	Sets the labels of a MultiInputBox object to the passed array of String.
`getInputs`	`<none>`	Returns an array of String entered by the user.

FIGURE 9.13 How identifiers used in a method are associated to a local variable, parameter, or instance/class variable.

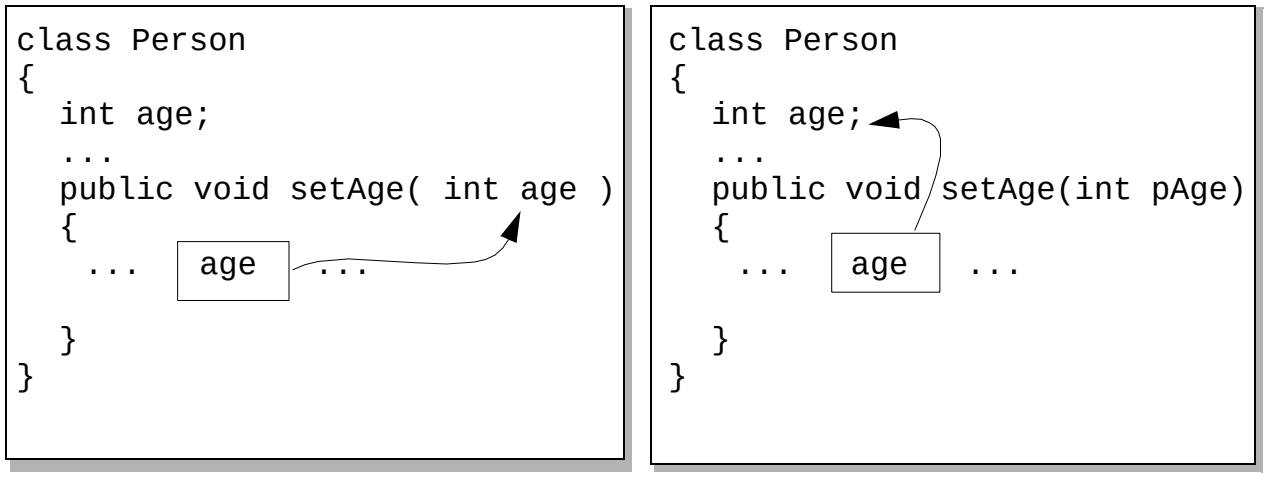


FIGURE 9.14 Using the self-referencing pointer to avoid the naming conflict.

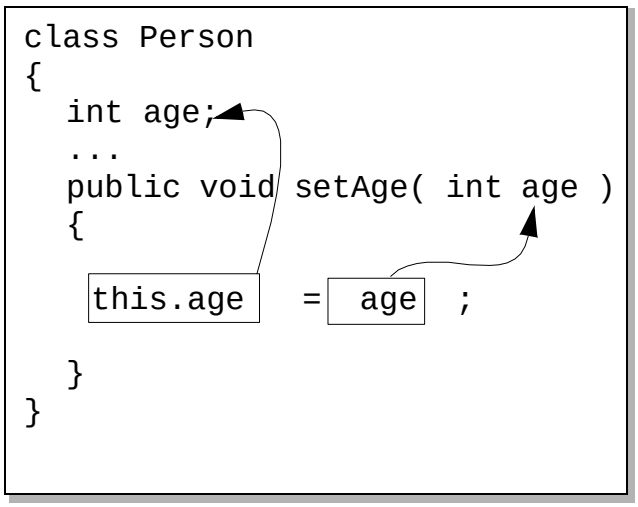
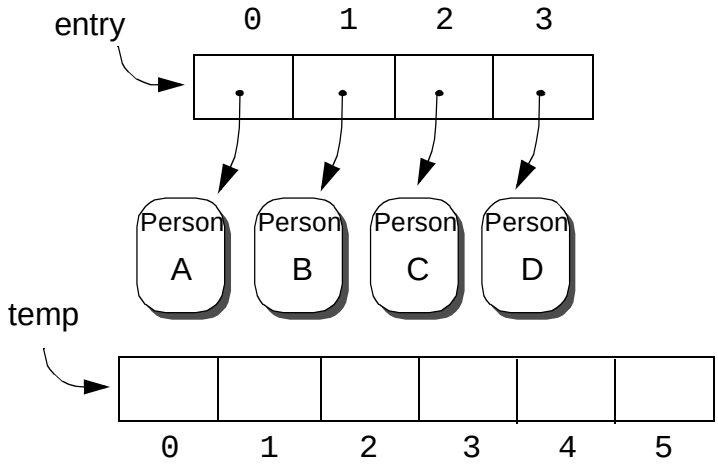


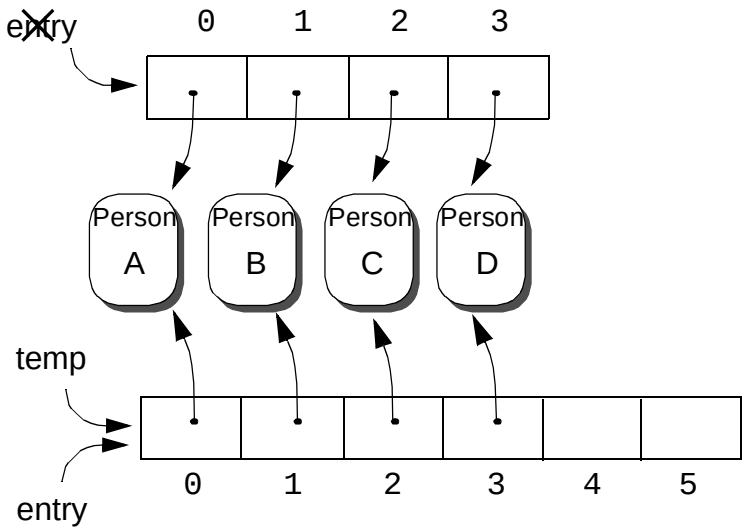
FIGURE 9.15 How a new array that is 150 percent larger than the original array is created. The size of the original array is 4.

```
Person[] temp;
int newLength = (int)(1.5f * entry.length);

temp = new Person[newLength];
```




```
for (int i = 0; i < entry.length; i++)
{
  temp[i] = entry[i];
}
entry = temp;
```



NOTE: The old array will eventually get returned to the system via garbage collection.

FIGURE 9.16 Adding a new **Person** object to the next available location. The array length is 4.

```
entry[count] = newPerson;

count++;
```

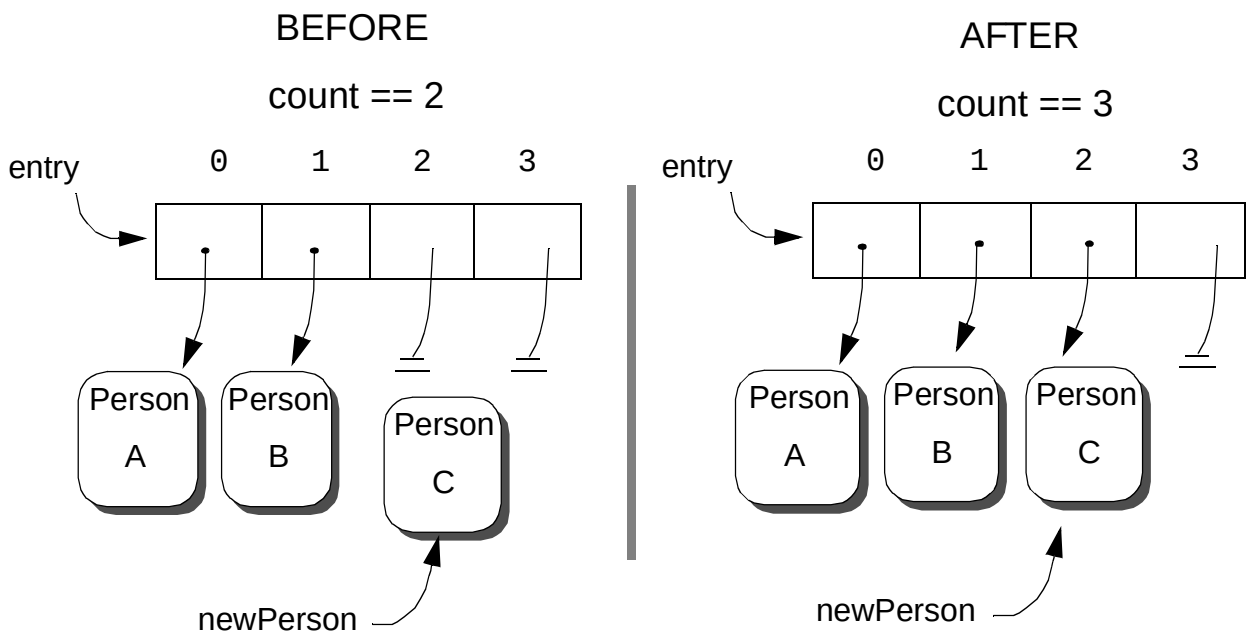


FIGURE 9.17 Examples of information represented as tables.

Distance Table (in miles)

	Los Angeles	San Francisco	San Jose	San Diego	Monterey
Los Angeles	—	600	500	150	450
San Francisco	600	—	100	750	150
San Jose	500	100	—	650	50
San Diego	150	750	650	—	600
Monterey	450	150	50	600	—

Multiplication Table

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	1	2	3	4	5	6	7	8	9
2	2	4	6	8	10	12	14	16	18
3	3	6	9	12	15	18	21	24	27
4	4	8	12	16	20	24	28	32	36
5	5	10	15	20	25	30	35	40	45
6	6	12	18	24	30	36	42	48	54
7	7	14	21	28	35	42	49	56	63
8	8	16	24	32	40	48	56	64	72
9	9	18	27	36	45	54	63	72	81

Tuition Table

	Day Students	Boarding Students
Grades 1 – 6	$ 6,000.00	$ 18,000.00
Grades 7 – 8	$ 9,000.00	$ 21,000.00
Grades 9 – 12	$ 12,500.00	$ 24,500.00

FIGURE 9.18 Accessing an element of a two-dimensional array.