Activity 7: Arrays

Programs often need to store multiple values of the same type, such as a list of phone numbers, or the names of your top 20 favorite songs. Rather than create a separate variable for each one, we can store them together using an array.

## Content Learning Objectives

*After completing this activity, students should be able to:*

* Declare and initialize array variables of primitive types.
* Draw a memory diagram of an array of reference types.
* Write a for loop that iterates the contents of an array.

## Process Skill Goals

*During the activity, students should make progress toward:*

* Developing algorithms that loop through arrays to compute a result. (Problem Solving)

.

# Model 1 Array Syntax

An *array* allows you to declare a collection of related variables (of the same type) at once. Each value in an array is known as an *element*. The programmer must specify the *length* of the array (the number of array elements). Once the array is created, its length cannot be changed.

char[] letterArray = {'H', 'i'};

System.out.println(letterArray[0]); // outputs H

System.out.println(letterArray.length); // outputs 2

double[] numberArray = new double[365];

System.out.println(numberArray[0]); // outputs 0.0

System.out.println(numberArray.length); // outputs 365

Array elements are accessed by *index* number, starting at zero:

|  |  |  |  |
| --- | --- | --- | --- |
| 0.0 | 0.0 | · · · | 0.0 |

|  |  |  |
| --- | --- | --- |
| 'H' | 'i' |  |
| 0 | 1 | 0 | 1 | 364 |

1. Examine the results of the above code.
	1. What is the index for the element 'i'?
	2. What is the length of letterArray?
	3. What is the length of numberArray?
	4. How would you access the last element of numberArray?
2. Now examine the syntax of the code.
	1. When square brackets [] are used?
	2. In contrast, how are curly braces {} used for an array?
3. What are the type and value for each of the four *expressions* below?

int[] a = {3, 6, 15, 22, 100, 0};

double[] b = {3.5, 4.5, 2.0, 2.0, 2.0};

String[] c = {"alpha", "beta", "gamma"};

a) a[3] + a[2]

b) b[2] - b[0] + a[4]

c) c[1].charAt(a[0])

d) a[4] \* b[1] <= a[5] \* a[0]

-------------------------------------------------------------------------------

Array variables can be initialized without the new keyword:

int[] picks = {3, 5, 7, 2, 1};

String[] names = {"Grace", "Alan", "Tim"};

However, if the variable is already declared, new is required:

picks = new int[] {3, 5, 7, 2, 1};

names = new string[] {"Grace", "Alan", "Tim"};

-------------------------------------------------------------------------------

1. Write *expressions* that create the following new arrays.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 14 | 1024 | 127 | 3 | 5521 |

a)

b)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3.23 | 1.52 | 4.23 | 32.5 | 2.45 | 5.23 | 3.33 |

1. Write *statements* that both declare and initialize variables for these new arrays.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 14 | 1024 | 127 | 3 | 5521 |

a)

b)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3.23 | 1.52 | 4.23 | 32.5 | 2.45 | 5.23 | 3.33 |

# Model 2 Array Diagrams

Array elements are stored together in one contiguous block of memory. To show arrays in memory diagrams, we simply draw adjacent boxes.

int[] nums = {10, 3, 7, -5}; double[] stats = new double[3];

 

## Questions

1. Draw a memory diagram for the following array declarations.

a) int[] sizes = new int[5];

 sizes[2] = 7;

b) double[] costs = new double[4];

 costs[0] = 0.99;

c) String[] names = new String[3];

 names[1] = "Anita";

1. What is the *default* value for uninitialized array elements? (Hint: You should have no empty boxes in your memory diagrams above.)
2. Like strings, arrays are reference types. What is the *value* of an array variable?

#

# Model 3 Arrays and Loops

The real power of arrays is the ability to process them using loops, i.e., performing the same task for multiple elements.

for (int i = 0; i < array.length; i++) {

// ... process array[i] ...

}

Here are two specific examples:

// set all of the elements of x to -1.0

double[] x = new double[100];

for (int i = 0; i < x.length; i++) {

x[i] = -1.0;

 }

 // sum the elements of scores

 int sum = 0;

 for (int i = 0; i < scores.length; i++) {

sum += scores[i];

}

1. What is the value of array and accumulator at the end of the following code? Trace the loop by hand in the space below.

int[] array = {5, 26, 13, 12, 37, 15, 16, 4, 1, 3};

int accumulator = 0;

for (int i = 0; i < array.length; i++) {

if (array[i] % 2 == 1 && i + 1 < array.length) {

array[i] \*= -1;

accumulator += array[i+1];

}

}

1. Implement the following method that creates and returns a new array.

/\*\*

\* Return a new array containing the pairwise maximum value of

\* the arguments. For each subscript i, the return value at [i]

\* will be the larger of x[i] and y[i].

\*

\* @param x an array of double values

\* @param y an array of double values

\* @return pairwise max of x and y

\*/

public static double[] pairwiseMax(double[] x, double[] y) {

}

1. Implement the following method that reads through two integer arrays.

/\*\*

\* Computes the final average grade for a student. The labs are

\* worth 40% and the exams are worth 60%. All scores range from

\* 0 to 100, inclusive.

\*

\* @param labs the student's lab scores

\* @param exams the student's exam scores

\* @return weighted average of all scores

\*/

public static double finalGrade(int[] labs, int[] exams) {

}