Activity 8: Arrays

Programs often need to store multiple values of the same type, such as a list of 1000 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.

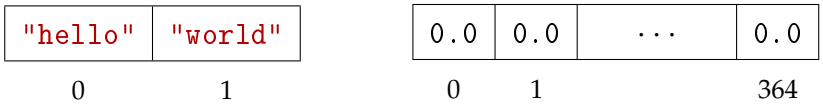
Model 1 Array Syntax

Each value in the array is known as an *element*. The programmer must specify the *length* of the array (the number of array elements).

```
String[] wordArray = {"hello", "world"};
System.out.println(wordArray[0]);            // outputs hello
System.out.println(wordArray.length);        // outputs 2

double[] numberArray = new double[365];
System.out.println(numberArray[0]);          // outputs 0.0
System.out.println(numberArray.length);      // outputs 365
```

Once the array is created, its length cannot be changed. Array elements are accessed by *index* number, starting at zero:



Questions (15 min) Start time: _____

1. Examine the results of the above code.

   a) What is the index for the element `"world"`?

   b) What is the length of the `wordArray`?

   c) What is the length of the `numberArray`?

   d) How would you access the last element of `numberArray`?

2. Now examine the syntax of the code.

   a) What are three ways that square brackets [] are used?

   b) In contrast, how are curly braces used for an array?

Array variables can be initialized without the `new` keyword:

```
int[] sizes = {3, 5, 7, 2, 1};
String[] names = {"James", "Madison", "University"};
```

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

```
sizes = new int[] {3, 5, 7, 2, 1};
names = new String[] {"James", "Madison", "University"};
```

3. Write *expressions* that create the following `new` arrays. (Do not declare variables.)

a)

0	14	1024	127	3	5521

b)

3.23	1.52	4.23	32.5	2.45	5.23	3.33

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

a)

0	14	1024	127	3	5521

b)

3.23	1.52	4.23	32.5	2.45	5.23	3.33

5. 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]`

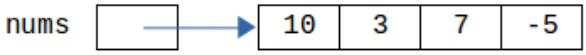
c) `c[1].charAt(a[0])`

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

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;
```



Questions (15 min) Start time: _____

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

   a)
   ```
   int[] sizes = new int[5];
   sizes[2] = 7;
   ```

   b)
   ```
   char[] codes = new char[3];
   codes[2] = 'X';
   ```

   c)
   ```
   double[] costs = new double[4];
   costs[0] = 0.99;
   ```

7. What is the *default* value for uninitialized array elements? (Hint: You should have no empty boxes in your memory diagram.)

8. Like strings, arrays are reference types. What is the *value* of an array variable?

9. Draw a memory diagram of the following array. (Hint: You should have four arrows.)

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

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. The standard form of iteration is as follows:

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

For example:

```
// 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];
}
```

Questions (15 min) Start time: _____

10. What is the value of `array` and `accumulator` after the following iteration? 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];
    }
}
```

11. 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; returns null if x.length
 *          != y.length or if either of the arrays is null
 */
public static double[] pairwiseMax(double[] x, double[] y) {









}
```

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

```
/**
 * Computes the final letter grade for a student. The labs are
 * worth 40% and the exams are worth 60%. All scores range from
 * 0 to 100, inclusive. Letter grades are assigned on the scale
 * A=90-100, B=80-89, C=70-79, D=60-69, F=0-59.
 *
 * @param labs the student's lab scores
 * @param exams the student's exam scores
 * @return letter grade 'A', 'B', 'C', 'D', or 'F'
 */
public static char finalGrade(int[] labs, int[] exams) {









}
```