Array Basics

For each of the following questions, write the appropriate code in your notebook (you will need to have this exercise in your notebook to receive full credit during notebook checks). Using your array whiteboard and markers, construct a model of the object constructed by Java.

**To get full credit on this assignment, you must record your answers in your notebook *and* construct models of the arrays using the whiteboard or papers given to you. Your instructor will be checking to make sure you are able to model array construction using the modeling materials provided to you.**

Exercise 1 *(Example)*

Construct an array that stores 8 elements of type int. Don’t forget to show how Java auto-initializes your elements!

In your notebook: With your modeling supplies:

int[] array = new int[8];

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Exercise 2: Constructing & Traversing Arrays

*Problem 2a.*

Using indexes, assign values to each element of your array.

*Problem 2b.*

Write code that will initialize the variables to non-default values in your model array. Trade this code with another group, and check to see if they construct the array you wanted it. See if they construct the array the way you wanted it.

*Problem 2c.*

Generate code that constructs an array with 5 elements, and fills the array with 5 consecutive integers.

*Problem 2d.*

Construct a loop that will grab input from the user and traverse the array to fill it in.

*Problem 2e.*

Using only one statement, declare an array of six integer values: -4, 19, 3, 20, 72, 58.

Exercise 3. Accessing Arrays

For each of the following problems, you should write the code in your notebooks, then work as a group to trace the flow of control and model where and how Java accesses the array.

*Problem 3a.*

Using array.length, write code that will output the last value of the array you constructed in Problem 2d. *Hint: Don’t forget to use 0-based indexing!*

*Problem 3b.*

Generate code to output the middle value of the array you wrote in Problem 2d.

*Problem 3c.*

Using the array you wrote in Problem 2d, write code that decrements each element by 1.

*Problem 3d.*

Write code that creates an array named *data*, with the following contents:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 31.1 | 0.4 | -.33 | 28.5 | 5.7 |
| 0 | 1 | 2 | 3 | 4 |

data 🡪

*Hint: If you’re having trouble constructing your structure diagram or pseudocode, try using your model to figure out which steps you need Java to execute.*

*Problem 3e.*

Construct an array called *even* that stores the first 499 even integers starting at 0 (you don’t have to model the complete array!)

*Problem 3f.*

Write code that will output the value at index 100 in array *even*.

*Problem 3g.*

Write code that accesses array *even* and outputs the value at indexes 278 and 456.

*The methods in 3f and 3g illustrate “random access,” a very fast way to access your data! The methods you write access data values that are very far apart in your array, without making Java traverse the whole array sequentially.*

*Problem 3h.*

Write code that stores all odd numbers between -4 and 5 into an array using a loop. Make the array’s size exactly large enough to store the numbers (including -4 and 5).

*Problem 3i.*

Try generalizing your code from Problem 3h so that it will work for any minimum and maximum values, not just -4 and 5. To test your generalized code, use the modeling supplies to test out your code using different values.