



## More Arrays (Savitch, Chapter 7)

### TOPICS

- Array Basics
- Arrays in Classes and Methods
- Programming with Arrays
- Searching and Sorting Arrays
- Multi-Dimensional Arrays
- Static Variables and Constants

CS160 - Spring Semester 2014

2



## Array Basics: Outline

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

CS160 - Spring Semester 2014

2



## Creating and Accessing Arrays

- An array is a special kind of object
- Think of as collection of variables of same type
- 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 braces
- Array indices begin at zero

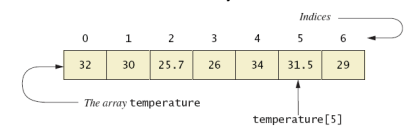
CS160 - Spring Semester 2014

3



## Creating and Accessing Arrays


- Figure 7.1 A common way to visualize an array



- Note `class ArrayOfTemperatures`

CS160 - Spring Semester 2014

4




## 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

CS160 - Spring Semester 2014 5




## 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

CS160 - Spring Semester 2014 6



## Square Brackets with Arrays


- With a data type when declaring an array:
 

```
int []pressure;
int pressure[];
```
- To enclose an integer expression to declare the length of the array:
 

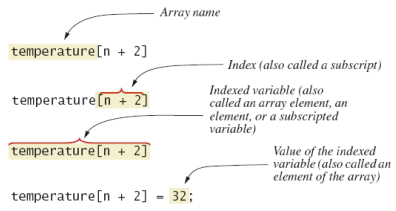
```
pressure = new int[100];
```
- To access the array with an index value:
 

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

CS160 - Spring Semester 2014 7



## Array Details

- Figure 7.2 Array terminology
 

CS160 - Spring Semester 2014 8



## The Instance Variable `length`

- As an object an array has only one public instance variable:
  - Variable `length`
  - Contains number of elements in the array
  - It is final, value cannot be changed
- Note `class ArrayOfTemperatures2`



## 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 exception occurs
- 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

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



## Arrays in Classes and Methods: Outline

- Indexed Variables as Method Arguments
- Entire Arrays as Arguments to a Method
- Arguments for the Method main
- Array Assignment and Equality
- Methods that Return Arrays

CS160 - Spring Semester 2014

13



## Indexed Variables as Method Arguments

- Indexed variable of an array
  - Example ... `a[i]`
  - Can be used anywhere variable of array base type can be used
- View `class ArgumentDemo` using indexed variable as an argument

CS160 - Spring Semester 2014

14



## Entire Arrays as Arguments

- Declaration of array parameter similar to how an array is declared
- Example:

```
public class SampleClass
{
    public static void incrementArrayBy2(double[] anArray)
    {
        for (int i = 0; i < anArray.length; i++)
            anArray[i] = anArray[i] + 2;
    }
    <The rest of the class definition goes here.>
}
```

CS160 - Spring Semester 2014

15



## Entire Arrays as Arguments

- Array parameter in a method heading does not specify the length
  - An array of any length can be passed to the method
  - Inside the method, elements of the array can be changed
- When you pass the entire array, do not use square brackets in the actual parameter

CS160 - Spring Semester 2014

16



## Arguments for Method main

- Recall heading of method **main**  
`public static void main (String[] args)`
- This declares an array
  - Formal parameter named **args**
  - Its base type is **String**
- Thus possible to pass to the run of a program multiple strings
  - These can then be used by the program

CS160 - Spring Semester 2014

17



## Array Assignment and Equality

- Arrays are objects
  - Assignment and equality operators behave (misbehave) as specified in previous chapter
- Variable for the array object contains memory address of the object
  - Assignment operator **=** copies this address
  - Equality operator **==** tests whether two arrays are stored in same place in memory

CS160 - Spring Semester 2014

18



## Array Assignment and Equality

- Two kinds of equality
- View `class TestEquals`

Sample screen output

Not equal by ==.  
Equal by the equals method.

CS160 - Spring Semester 2014

19



## Array Assignment and Equality

- Note results of **==**
- Note definition and use of method **equals**
  - Receives two array parameters
  - Checks length and each individual pair of array elements
- Remember array types are reference types

CS160 - Spring Semester 2014

20



## Methods that Return Arrays

- A Java method may return an array
- View **class ReturnArrayDemo**
- Note definition of return type as an array
- To return the array value
  - Declare a local array
  - Use that identifier in the **return** statement



## Programming with Arrays and Classes: Outline

- Programming Example: A Specialized List Class
- Partially Filled Arrays




## Programming Example

- A specialized List class
  - Objects can be used for keeping lists of items
- Methods include
  - Capability to add items to the list
  - Also delete entire list, start with blank list
  - But no method to modify or delete list item
- Maximum number of items can be specified



## Programming Example

- View **class ListDemo**
- Note declaration of the list object
- Note method calls




## Programming Example

```

Enter items for the list, when prompted.
Enter an item:
Buy milk
More items for the list? yes
Enter an item:
Walk dog
More items for the list? yes
Enter an item:
Buy milk
More items for the list? yes
Enter an item:
Write program
The list is now full.
The list contains:
Buy milk
Walk dog
Write program
    
```

Sample screen output


CS160 - Spring Semester 2014 25



## Programming Example

- Now view [array wrapped in a class](#) to represent a list, **class OneWayNoRepeatsList**
- Notable code elements
  - Declaration of private array
  - Method to find  $n^{\text{th}}$  list item
  - Method to check if item is on the list or not


CS160 - Spring Semester 2014 26



## Partially Filled Arrays

- Array size specified at definition
- Not all elements of the array might receive values
  - This is termed a *partially filled array*
- Programmer must keep track of how much of array is used

CS160 - Spring Semester 2014 27



## Partially Filled Arrays

- Figure 7.4 A partially filled array

	entry	
entry[0]	Buy milk.	
entry[1]	Call home.	
entry[2]	Go to beach.	← The item at index countOfEntries - 1
entry[3]	?	← Unused elements
entry[4]	?	← Unused elements

entry.length has a value of 5.  
 countOfEntries has a value of 3.

CS160 - Spring Semester 2014 28



## Multidimensional-Array Basics

- Consider Figure 7.6, a table of values

Year	5.00%	5.50%	6.00%	6.50%	7.00%	7.50%
1	\$1050	\$1055	\$1060	\$1065	\$1070	\$1075
2	\$1103	\$1113	\$1124	\$1134	\$1145	\$1156
3	\$1158	\$1174	\$1191	\$1208	\$1225	\$1242
4	\$1216	\$1239	\$1262	\$1286	\$1311	\$1335
5	\$1276	\$1307	\$1338	\$1370	\$1403	\$1436
6	\$1340	\$1379	\$1419	\$1459	\$1501	\$1543
7	\$1407	\$1455	\$1504	\$1554	\$1606	\$1659
8	\$1477	\$1535	\$1594	\$1655	\$1718	\$1783
9	\$1551	\$1619	\$1689	\$1763	\$1838	\$1917
10	\$1629	\$1708	\$1791	\$1877	\$1967	\$2061

CS160 - Spring Semester 2014

29



## Multidimensional-Array Basics

- Figure 7.7 Row and column indices for an array named `table`

Indices	0	1	2	3	4	5
0	\$1050	\$1055	\$1060	\$1065	\$1070	\$1075
1	\$1103	\$1113	\$1124	\$1134	\$1145	\$1156
2	\$1158	\$1174	\$1191	\$1208	\$1225	\$1242
3	\$1216	\$1239	\$1262	\$1286	\$1311	\$1335
4	\$1276	\$1307	\$1338	\$1370	\$1403	\$1436
5	\$1340	\$1379	\$1419	\$1459	\$1501	\$1543
6	\$1407	\$1455	\$1504	\$1554	\$1606	\$1659
7	\$1477	\$1535	\$1594	\$1655	\$1718	\$1783
8	\$1551	\$1619	\$1689	\$1763	\$1838	\$1917
9	\$1629	\$1708	\$1791	\$1877	\$1967	\$2061

`table[3][2]` has a value of 1262

CS160 - Spring Semester 2014

30



## Multidimensional-Array Basics

- We can access elements of the table with a nested for loop

Example:

```
for (int row = 0; row < 10; row++)
  for (int column = 0; column < 6; column++)
    table[row][column] =
      balance(1000.00, row + 1, (5 + 0.5 * column));
```

- View `class InterestTable`

CS160 - Spring Semester 2014

31



## Multidimensional-Array Basics

Balances for Various Interest Rates Compounded Annually (Rounded to Whole Dollar Amounts)

Years	5.00%	5.50%	6.00%	6.50%	7.00%	7.50%
1	\$1050	\$1055	\$1060	\$1065	\$1070	\$1075
2	\$1103	\$1113	\$1124	\$1134	\$1145	\$1156
3	\$1158	\$1174	\$1191	\$1208	\$1225	\$1242
4	\$1216	\$1239	\$1262	\$1286	\$1311	\$1335
5	\$1276	\$1307	\$1338	\$1370	\$1403	\$1436
6	\$1340	\$1379	\$1419	\$1459	\$1501	\$1543
7	\$1407	\$1455	\$1504	\$1554	\$1606	\$1659
8	\$1477	\$1535	\$1594	\$1655	\$1718	\$1783
9	\$1551	\$1619	\$1689	\$1763	\$1838	\$1917
10	\$1629	\$1708	\$1791	\$1877	\$1967	\$2061

Sample screen output

CS160 - Spring Semester 2014

32





## Java's Representation of Multidimensional Arrays

- Multidimensional array represented as several one-dimensional arrays
- Given  

```
int [][] table = new int [10][6];
```
- Array table is actually 1 dimensional of type `int[]`
  - It is an array of arrays
- Important when sequencing through multidimensional array



## Summary

- An array is a collection of variables all of the same type
- Arrays are objects, created with operator `new`
- Elements numbered starting with 0, ending with 1 less than length
- Indexed variable can be used as a parameter – treated like variable of base type



## Summary

- Entire array can be passed as parameter to a method
- Method return value can be an array
- Partially filled array usually stores values in initial segment, use an `int` to track how many are used



## Summary

- Multidimensional arrays are implemented as an array of arrays
- Treat two-dimensional array as a table with rows and columns



## public vs. private

- Variables defined as public are visible (and changeable) outside the class
  - Sometimes called global variables
- Methods defined as public are visible (and callable) outside the class
- Private variables are visible/defined only within their scope

CS160 - Spring Semester 2014

37



## Scope

- Class instance variables
  - Can be accessed anywhere in the class
- Method instance variables
  - Can be accessed anywhere in the method
  - Method parameters are local to the method
  - Method parameters are initialized to the value of the passed argument
- Loop instance variables
  - Can be accessed anywhere in the loop

CS160 - Spring Semester 2014

38



## Static Variables

- Static variables are shared by all objects of a class
  - Variables declared **static final** are considered constants – value cannot be changed
- Variables declared **static** (without **final**) can be changed
  - Only one instance of the variable exists
  - It can be accessed by all instances of the class

CS160 - Spring Semester 2014

39



## Static Variables

- Static variables also called *class variables*
  - Contrast with *instance variables*
- Do not confuse class variables with variables of a class type
- Both static variables and instance variables are sometimes called *fields* or *data members*

CS160 - Spring Semester 2014

40



## Using Named Constants

- To avoid confusion, always name constants (and variables).  
`area = PI * radius * radius;`  
is clearer than  
`area = 3.14159 * radius * radius;`
- Place constants near the beginning of the program.



## Named Constants

- Once the value of a constant is set (or changed by an editor), it can be used (or reflected) throughout the program.  
`public static final double  
INTEREST_RATE = 6.65;`
- If a literal (such as 6.65) is used instead, every occurrence must be changed, with the risk that another literal with the same value might be changed unintentionally.



## Declaring Constants

- Syntax  
`public static final  
Variable_Type = Constant;`
- Examples  
`public static final double  
PI = 3.14159;`  
`public static final String MOTTO =  
"The customer is always right.";`
- By convention, uppercase letters are used for constants.