Arrays

Chapter 7

University of Zurich
Department of Informatics

s.e.a.l.
software evolution & architecture lab

Objectives

- Nature and purpose of an array
- Using arrays in Java programs
- Methods with array parameter
- Methods that return an array
- Array as an instance variable
- Use an array not filled completely

Objectives, cont.

- Order (sort) the elements of an array
- Search an array for a particular item
- Define, use multidimensional array

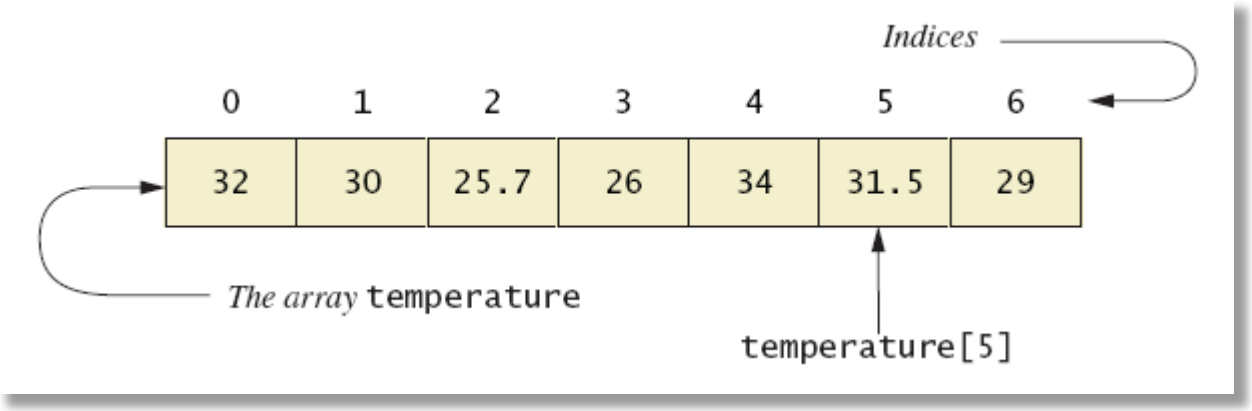
Creating and Accessing Arrays

- 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

Creating and Accessing Arrays



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

Square Brackets with Arrays

- With a data type when declaring an array

```
int [ ] pressure;
```

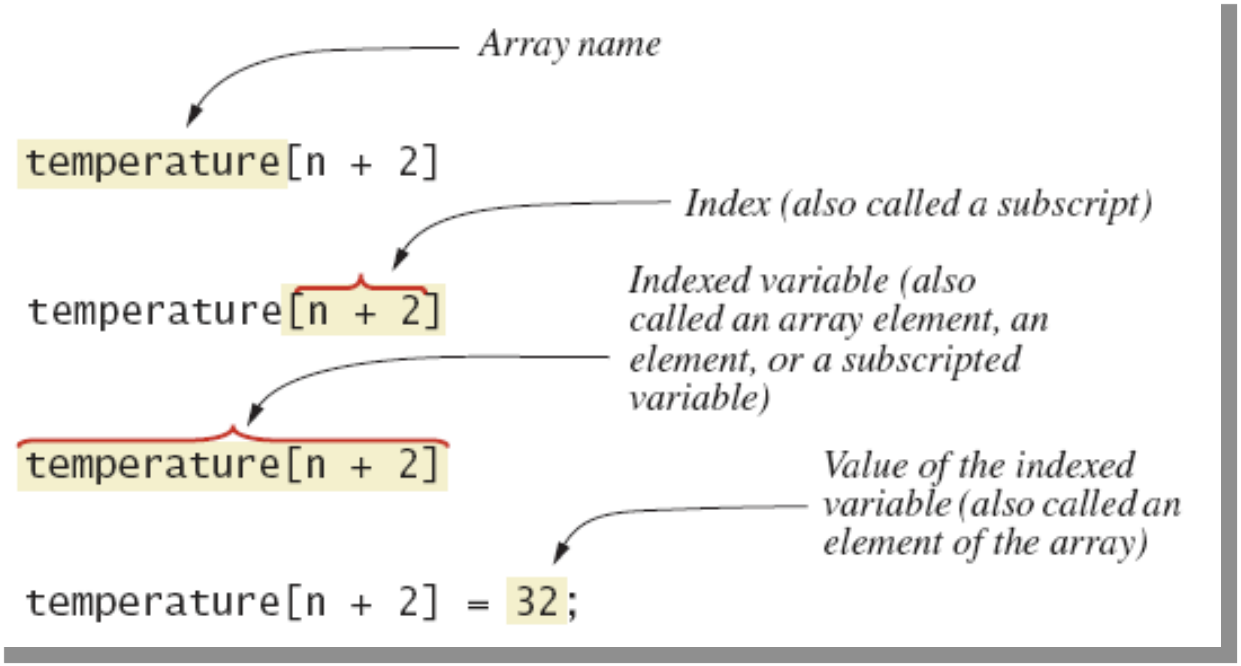
- To enclose an integer expression to declare the length of the array

```
pressure = new int [100];
```

- To name an indexed value of the array

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

Array Details



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
- example code

  `class ArrayOfTemperatures2`

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 error occurs
- OK to "waste" element 0
  - Program easier to manage and understand
  - Yet, 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
  - In a loop

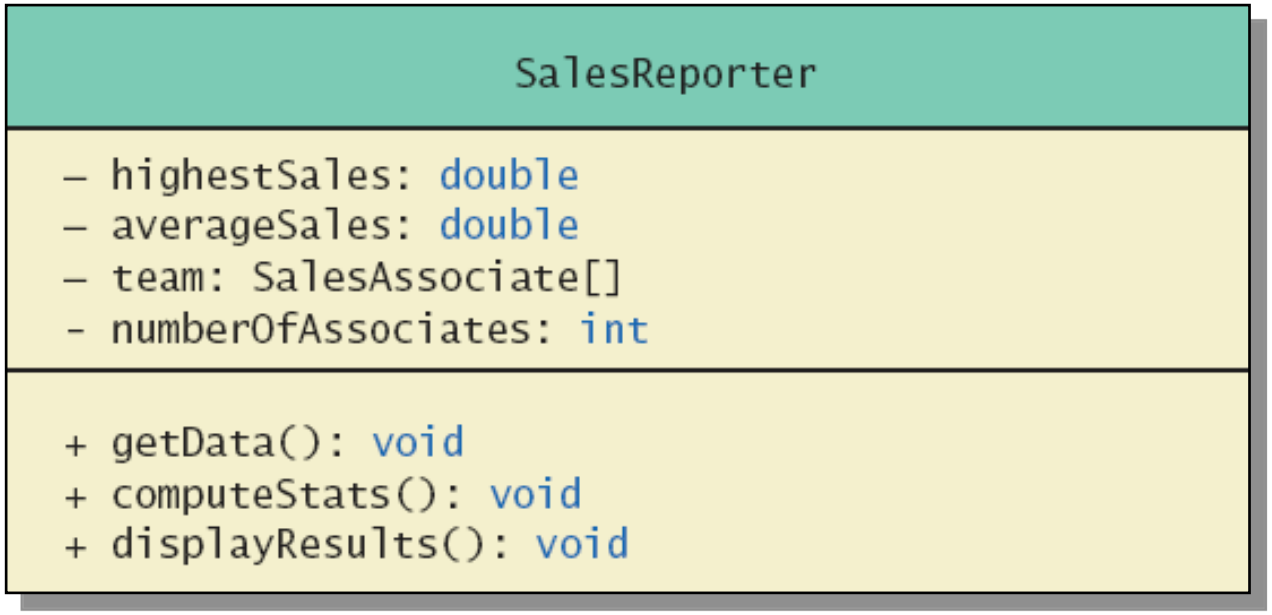
```
int[] count = new int[100];
for (int i = 0; i < 100; i++)
    count[i] = 0;
```

Case Study: Sales Report

- Program to generate a sales report
- Class will contain
  - Name
  - Sales figure
- View class declaration, listing 7.3

```
class SalesAssociate
```

Case Study: Sales Report



Case Study: Sales Report

class SalesReporter

```
Enter number of sales associates:
3
Enter data for associate number 1
Enter name of sales associate: Dusty Rhodes
Enter associate's sales: $36000
Enter data for associate number 2
Enter name of sales associate: Natalie Dressed
Enter associate's sales: $50000
Enter data for associate number 3
Enter name of sales associate: Sandy Hair
Enter associate's sales: $10000
Average sales per associate is $32000.0
```

Sample screen output

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 program using indexed variable as an argument, listing 7.5

  `class ArgumentDemo`

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.>
}
```

Entire Arrays as Arguments

- Note – 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

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

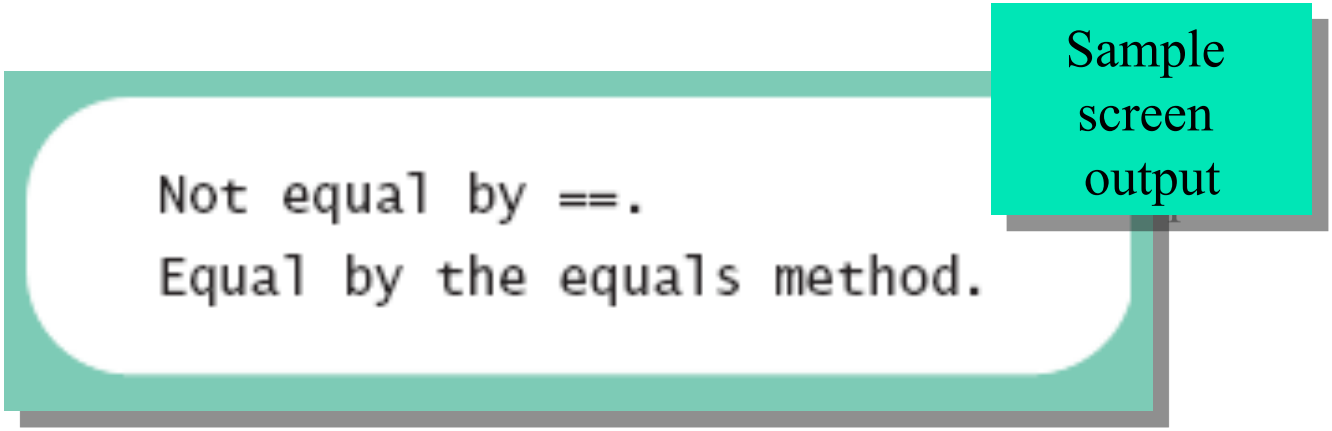
Array Assignment and Equality

- 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

Array Assignment and Equality

- Two kinds of equality
- View example program, listing 7.6

  `class TestEquals`



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

Methods that Return Arrays

- A Java method may return an array
- View example program, listing 7.7
  `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 demo program, listing 7.8

```
class ListDemo
```

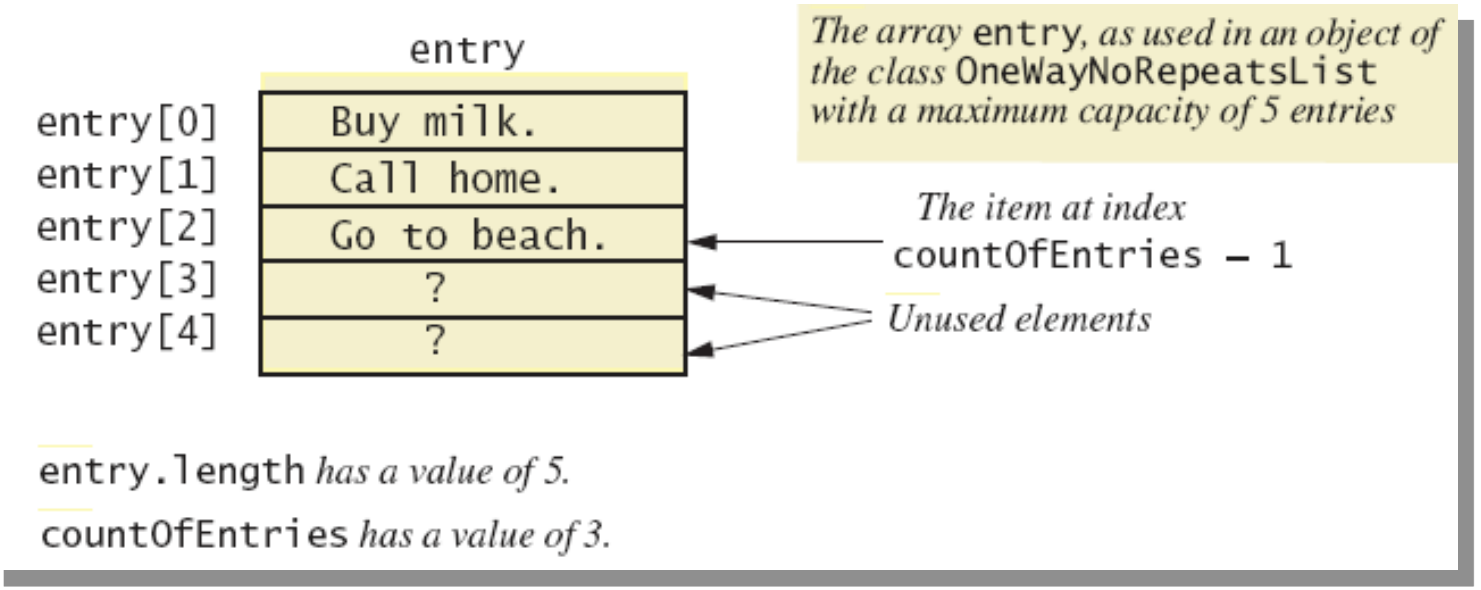
- Note declaration of the list object
- Note method calls

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

Partially Filled Arrays

- Figure 7.4 A partially filled array



Sorting, Searching Arrays: Outline

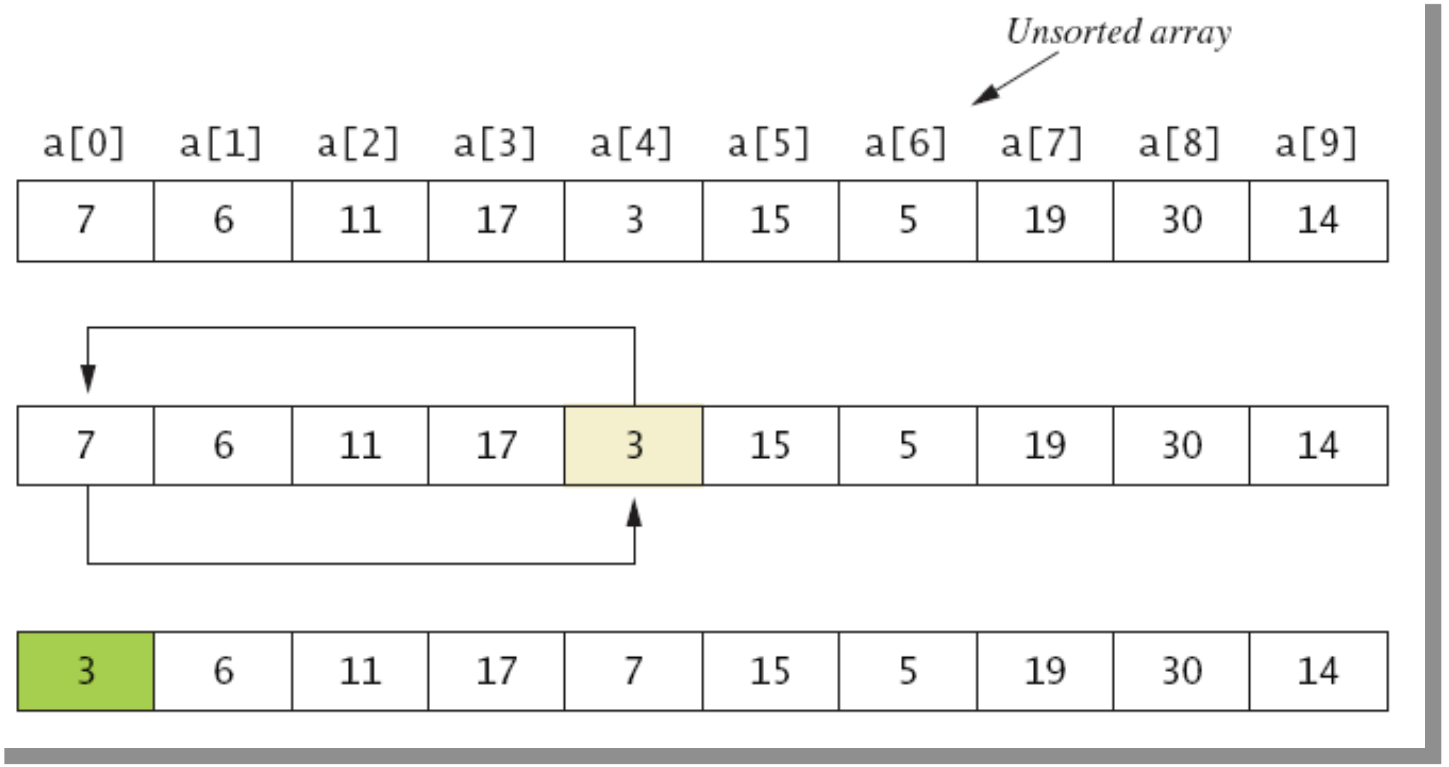
- Selection Sort
- Other Sorting Algorithms
- Searching an Array

Selection Sort

- Consider arranging all elements of an array so they are ascending order
- Algorithm is to step through the array
  - Place smallest element in index 0
  - Swap elements as needed to accomplish this
- Called an interchange sorting algorithm

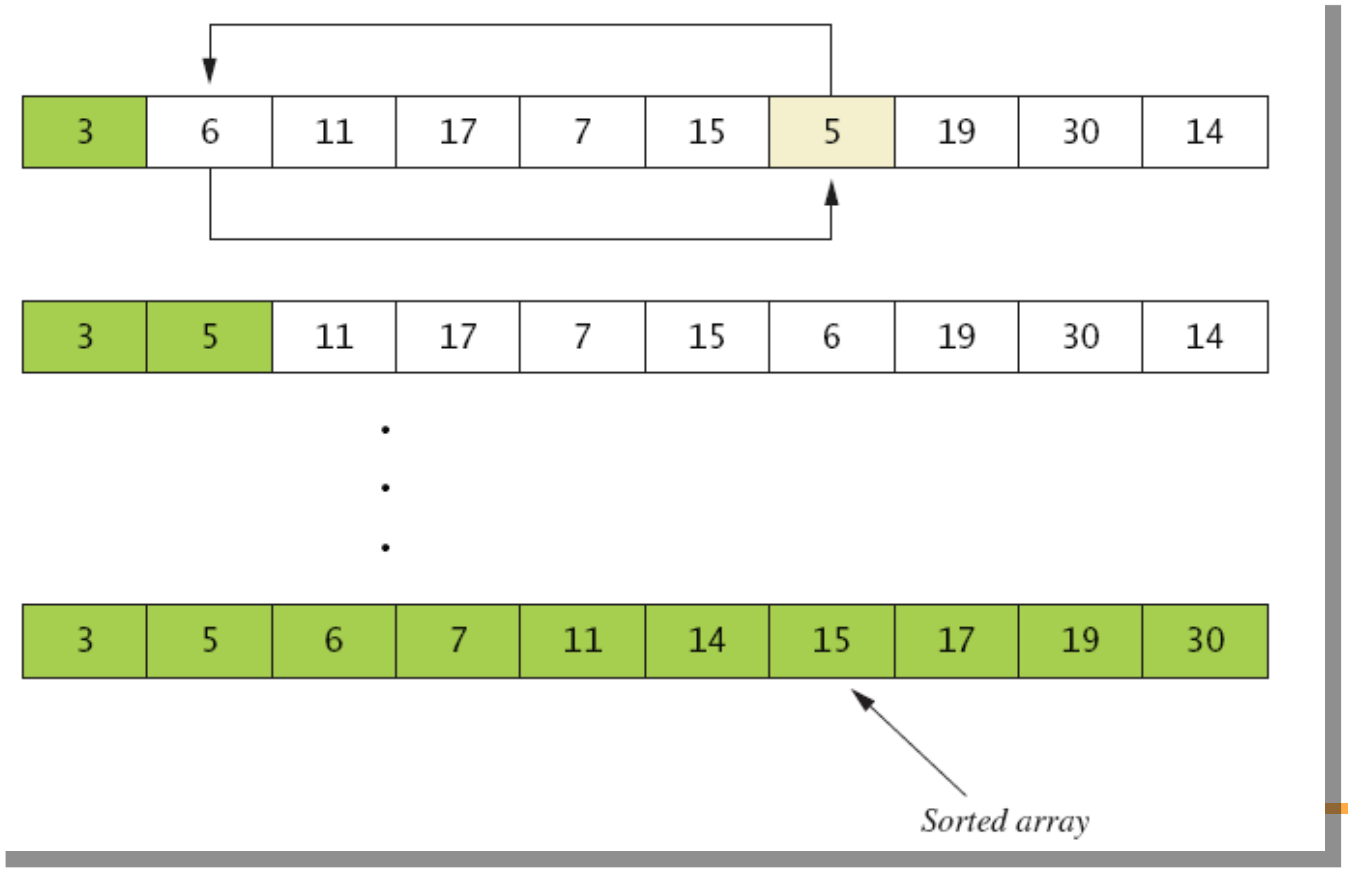
Selection Sort

- Figure 7.5a



Selection Sort

- Figure 7.5b



Other Sorting Algorithms

- Selection sort is simplest
  - But it is very inefficient
- Java Class Library provides for efficient sorting
  - Has a class called Arrays
  - Class has multiple versions of a sort method

Searching an Array

- Method used in **`OneWayNoRepeatsList`** is sequential search
  - Looks in order from first to last
  - Good for unsorted arrays
- Search ends when
  - Item is found … or …
  - End of list is reached
- If list is sorted, use more efficient searches

Multidimensional Arrays: Outline

- Multidimensional-Array Basics
- Multidimensional-Array Parameters and Returned Values
- Java's Representation of Multidimensional
- Ragged Arrays
- Programming Example: Employee Time Records

Multidimensional-Array Basics

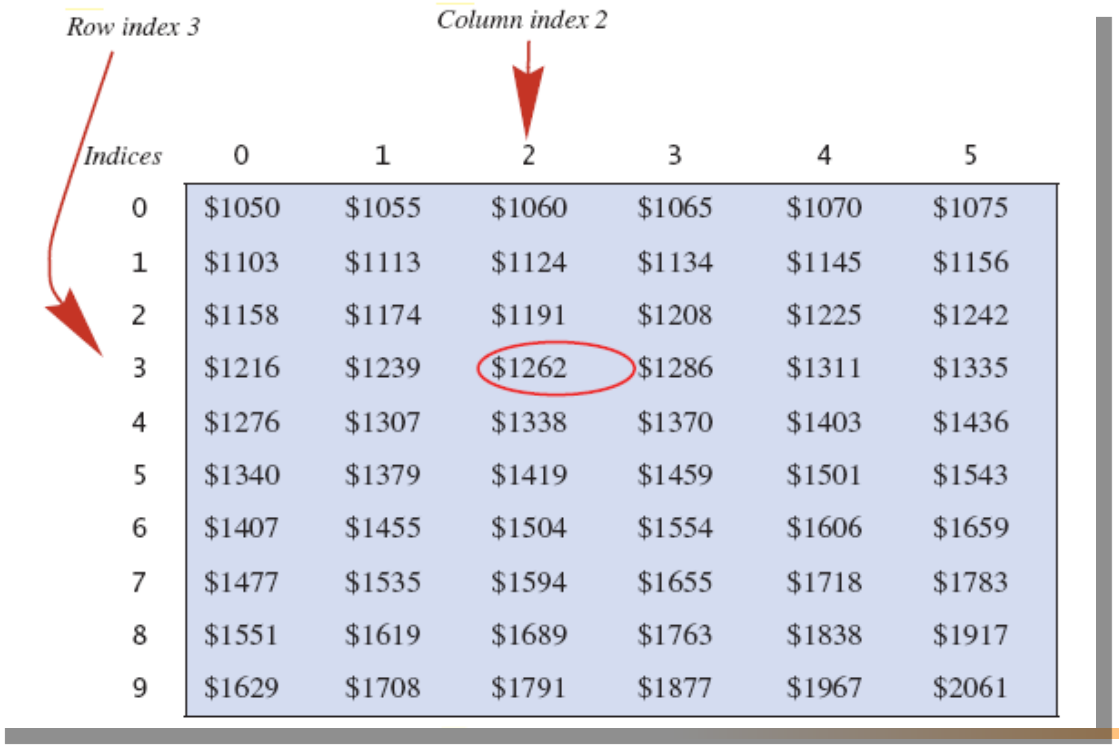
- Consider Figure 7.6, a table of values

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

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

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

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 sample program, listing 7.12

`class InterestTable`

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

Multidimensional-Array Parameters and Returned Values

- Methods can have
  - Parameters that are multidimensional-arrays
  - Return values that are multidimensional-arrays
- View sample code, listing 7.13

```
class InterestTable2
```

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

Programming Example

```
Employee   1    2    3    Totals
Monday     8    0    9    17
Tuesday    8    0    9    17
Wednesday  8    8    8    24
Thursday   8    8    4    20
Friday     8    8    8    24
Total  =   40   24   38
```

Sample screen output

Programming Example

- Figure 7.8 Arrays for the class **`TimeBook`**