11-12. Arrays

[ECE20016/ITP20003] Java Programming

Agenda

- Array Basics
- Arrays in Classes and Methods
- Sorting Arrays
- Multidimensional Arrays

Creating and Accessing Arrays

- An array is a special kind of **an object**
  - Think of as collection of variables of same type

- Syntax for declaring an array with *new* operator

  Base_Type[ ] *Array_Name* = new *Base_Type*[*Length*];

  (*Base_Type Array_Name*[ ] = new *Base_Type*[*Length*]; is also OK.)

  Ex) double [ ] temperature = new double [7];

- To access an element use
  - The name of the array + an index number enclosed in braces

    Ex) temperature[5];

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();

The Instance Variable length

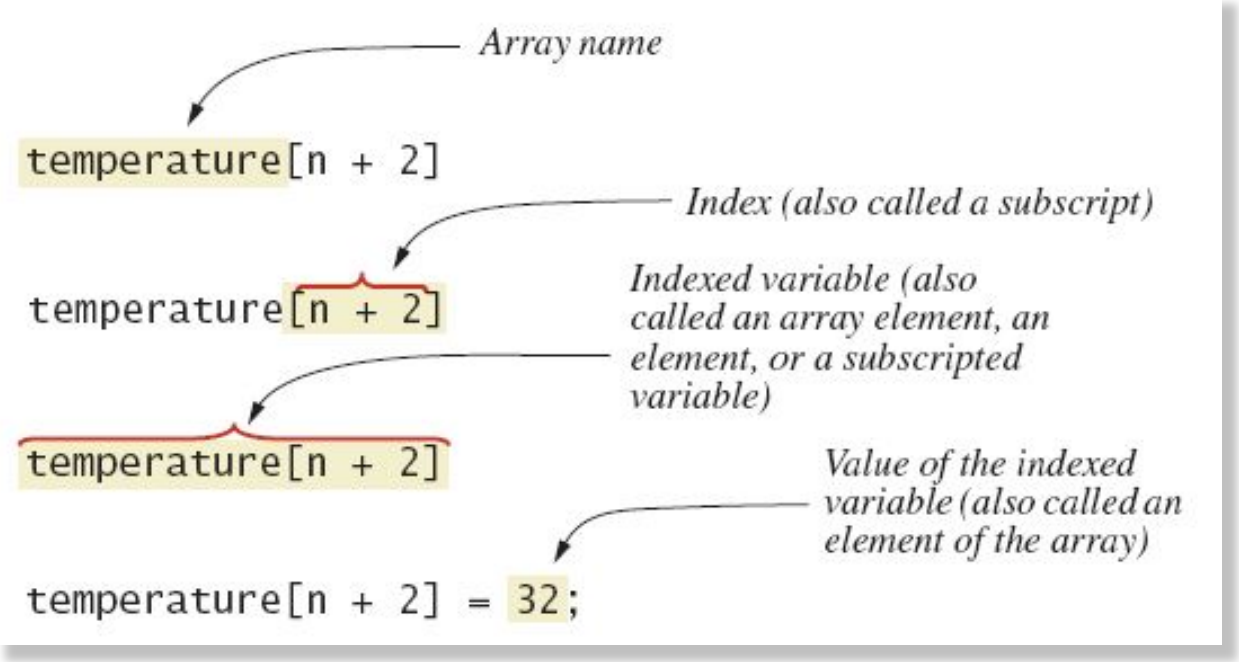
- As an object, an array has only one public instance variable
  - Variable *length*
    - Contains number of elements in the array
    - It is final, value cannot be changed

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

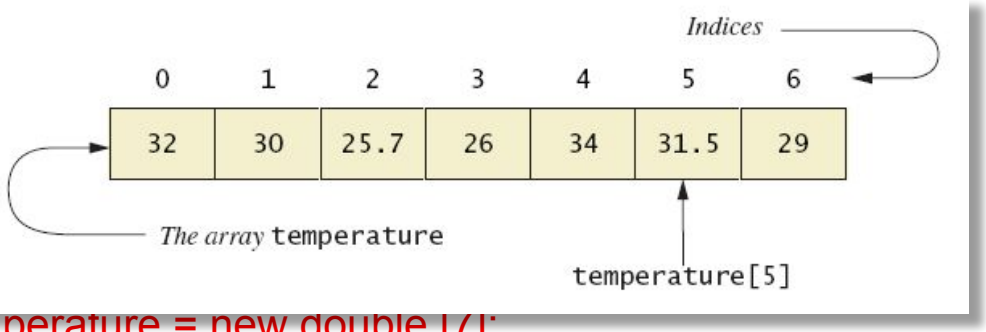
Array Details

- Array terminology



Creating and Accessing Arrays

- A common way to visualize an array




```
double [] temperature = new double [7];
// Read temperatures and compute their average:
Scanner keyboard = new Scanner (System.in);
System.out.println ("Enter 7 temperatures:");
double sum = 0;
for (int index = 0 ; index < 7 ; index++){
    temperature [index] = keyboard.nextDouble ();
    sum = sum + temperature [index];
}
double average = sum / 7;
```

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.
```

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;
```

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.
```

Agenda

- Array Basics
- **<u>Arrays in Classes and Methods</u>**
- Sorting Arrays
- Multidimensional Arrays

Case Study: Sales Report

- Program to generate a sales report
  - Associates (salesmen) that have the highest sales
  - How the sales of each associate compare to the average
- Class will contain
  - Name
  - Sales figure (amount of sales)

Case Study: Sales Report

- SalesReporter Class
  - Variables
    - private double highestSales;
    - private double averageSales;
    - private SalesAssociate[] team;

    - private int numberOfAssociates; *//Same as team.length*

Case Study: Sales Report

- Main subtasks for our program
  1. Get ready
  2. Obtain the data
  3. Compute some statistics (update instance variables)
     - *averageSales*, *highestSales*
  4. Display the results

Case Study: Sales Report

- Class diagram for class SalesReporter

```
SalesReporter
- highestSales: double
- averageSales: double
- team: SalesAssociate[]
- numberOfAssociates: int

+ getData(): void
+ computeStats(): void
+ displayResults(): void
```

Case Study: Sales Report

- Class diagram for class SalesAssociate

SalesAssociate
- name: String - sales: double
+ setName(String name): void + setSales(double sales): void + getName(): String + getSales(): double

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
```

Case Study: Sales Report

- class SalesReporter (2)

```
Average sales per associate is $32000.0
The highest sales figure is $50000.0

The following had the highest sales:
Name: Natalie Dressed
Sales: $50000.0
$18000.0 above the average.

The rest performed as follows:
Name: Dusty Rhodes
Sales: $36000.0
$4000.0 above the average.

Name: Sandy Hair
Sales: $10000.0
$22000.0 below the average.
```

Indexed Variables as Method Arguments

- Indexed variable of an array, such as a[i], can be used anywhere variable of array base type can be used.

```
public class ArgumentDemo {
    public static void main (String [] args)
    {
        Scanner keyboard = new Scanner (System.in);
        System.out.println ("Enter your score on exam 1:");
        int firstScore = keyboard.nextInt ();
        int [] nextScore = new int [3];
        for (int i = 0 ; i < nextScore.length ; i++)
            nextScore [i] = firstScore + 5 * i;
        for (int i = 0 ; i < nextScore.length ; i++) {
            double possibleAverage = getAverage (firstScore, nextScore [i]);
            System.out.println ("If your score on exam 2 is " + nextScore [i]);
            System.out.println ("your average will be " + possibleAverage);
        }
    }

    public static double getAverage (int n1, int n2)
    {
        return (n1 + n2) / 2.0;
    }
}
```

Entire Arrays as Arguments

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

  Ex)

```
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

- 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 (=) vs. equality (==) operators.
- 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 (address) in memory
- Remember array types are reference types

Methods that Return Arrays

- A Java method may return an array
- To return the array value
  - Declare a local array
  - Use that identifier in the return statement

```
public String[] getStudentNames() {

        String[] studentNames = new String[2];

        studentNames[0] = "JC Nam";
        studentNames[0] = "JH Park";

        return studentNames;
}
```

Agenda

- Array Basics
- Arrays in Classes and Methods
- **Sorting Arrays**
- Multidimensional Arrays

Selection Sort

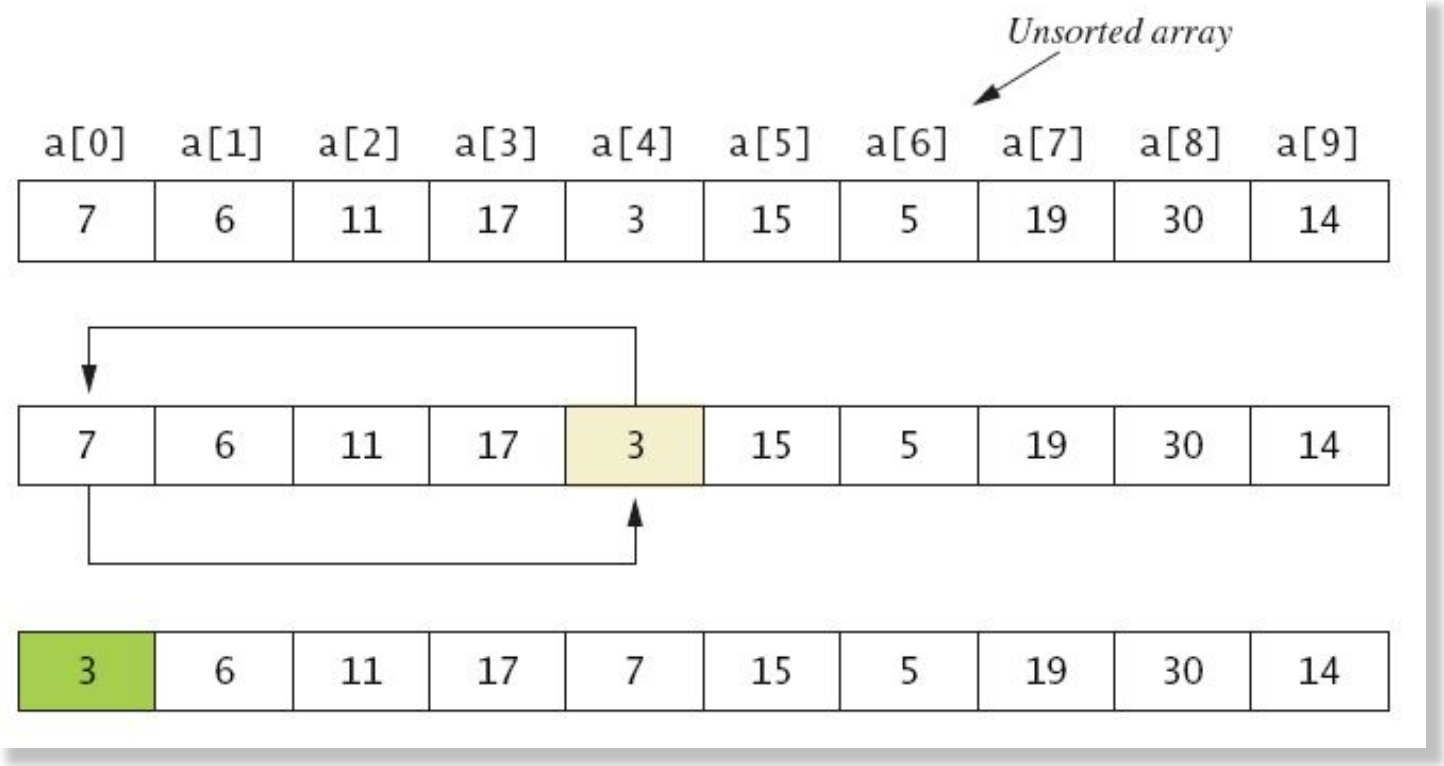
- Sorting: arranging all elements of an array so they are ascending (or descending) order

```
Array values before sorting:
7 5 11 2 16 4 18 14 12 30
Array values after sorting:
2 4 5 7 11 12 14 16 18 30
```

- Selection sort
  1. Algorithm is to step through the array
  2. Place smallest element in index 0
  3. Swap elements as needed to accomplish this

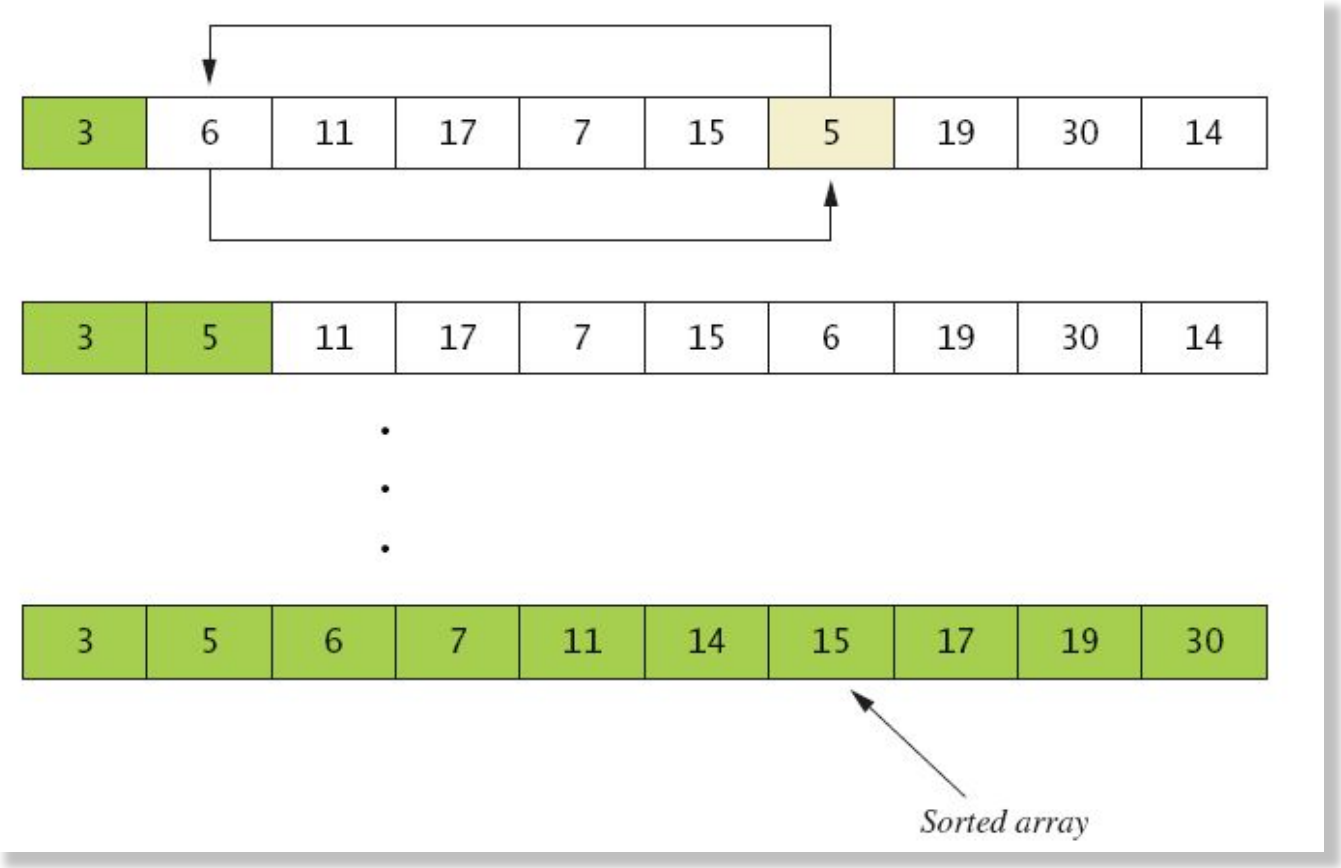
Selection Sort

- Figure 7.5a



Selection Sort

- Figure 7.5b



Selection Sort

- Algorithm for selection sort of an array

```
for (index = 0; index < a.length - 1; index++)
{// Place the correct value in a[index]:
    indexOfNextSmallest = the index of the smallest value among
                          a[index], a[index+1], ..., a[a.length - 1]
    Interchange the values of a[index] and a[indexOfNextSmallest].
    // Assertion: a[0] <= a[1] <=...<= a[index] and these
    // are the smallest of the original array elements.
    // The remaining positions contain the rest of the
    // original array elements.
}
```

- See JC's implementation
  - https://github.com/lifove/SelectionSorting/blob/master/src/main/java/edu/handong/csee/java/array/sort/ArraySorter.java

Other Sorting Algorithms

- Selection sort is simplest
  - But it is very inefficient for large arrays

- Java Class Library provides for efficient sorting
  - Has a class called *Arrays*
  - Class has multiple versions of a sort method (static methods)

    Ex) Arrays.sort(int[] a), Arrays.sort(double[] a), …
  - See http://java.oracle.com or

    http://docs.oracle.com/javase/9/docs/api/java/util/Arrays.html

Agenda

- Array Basics
- Arrays in Classes and Methods
- Sorting Arrays
- **Multidimensional Arrays**

Multidimensional-Array Basics

- A table of values

  Ex) int [ ][ ] table = new int [10][6];

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

- Row and column indices for an array named table

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

Row index 3

Column index 2

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

  Ex)

```
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));
```

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

Multidimensional-Array Parameters and Returned Values

- Methods can have
  - Parameters that are multidimensional-arrays
  - Return values that are multidimensional-arrays

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

Ragged Arrays

- Not necessary for all rows to be of the same length

  Ex)

```
int[][] b;
b = new int[3][];
b[0] = new int[5]; //First row,  5 elements
b[1] = new int[7]; //Second row, 7 elements
b[2] = new int[4]; //Third row,  4 elements
```

Programming Example

- Employee Time Records
  - Two-dimensional array stores hours worked
    - For each employee
    - For each of 5 days of work week
  - Array is private instance variable of class
- See the code from JC's GitHub
  - https://github.com/lifove/EmployeeTimeRecord/blob/master/src/main/java/edu/handong/csee/java/lecture/multidimensionalarray/TimeRecoder.java?ts=4

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	

Programming Example

- Arrays for the class TimeBook