

Arrays

©The McGraw-Hill Companies, Inc. Permission required for reproduction or display.



Array Basics

- An **array** is a collection of data values.
- If your program needs to deal with 100 integers, 500 Account objects, 365 real numbers, etc., you will use an array.
- In Java, an array is an indexed collection of data values of the same type.

©The McGraw-Hill Companies, Inc. Permission required for reproduction or display.



Arrays of Primitive Data Types

- Array Declaration

```
<data type> [ ] <variable>           //variation 1  
<data type> <variable>[ ]           //variation 2
```

- Array Creation

```
<variable> = new <data type> [ <size> ]
```

- Example

Variation 1

```
double[ ] rainfall;  
rainfall  
    = new double[12];
```

Variation 2

```
double rainfall [ ];  
rainfall  
    = new double[12];
```

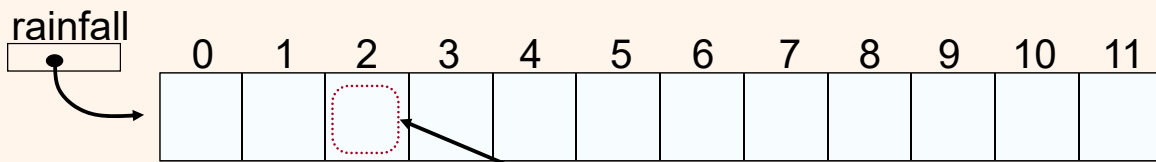
An array is like an object!



Accessing Individual Elements

- Individual elements in an array accessed with the indexed expression.

```
double[ ] rainfall = new double[12];
```



The index of the first position in an array is **0**.

rainfall[2]

This indexed expression refers to the element at position #2



Array Processing – Sample 1

```
Scanner scanner = new Scanner(System.in);
double[] rainfall = new double[12];

double    annualAverage,
          sum = 0.0;

for (int i = 0; i < rainfall.length; i++) {

    System.out.print("Rainfall for month " + (i+1));
    rainfall[i] = scanner.nextDouble( );
    sum += rainfall[i];
}

annualAverage = sum / rainfall.length;
```

The public constant length returns the capacity of an array.



Array Processing – Sample 2

```
Scanner scanner = new Scanner(System.in);
double[] rainfall = new double[12];
String[] monthName = new String[12];
monthName[0] = "January";
monthName[1] = "February";
...
double    annualAverage, sum = 0.0;

for (int i = 0; i < rainfall.length; i++) {
    System.out.print("Rainfall for " + monthName[i] + ": ");
    rainfall[i] = scanner.nextDouble();
    sum += rainfall[i];
}

annualAverage = sum / rainfall.length;
```

The same pattern for the remaining ten months.

The actual month name instead of a number.



Array Processing – Sample 3

- Compute the average rainfall for each quarter.

```
//assume rainfall is declared and initialized properly

double[] quarterAverage = new double[4];

for (int i = 0; i < 4; i++) {
    sum = 0;
    for (int j = 0; j < 3; j++) {
        sum += rainfall[3*i + j];    //compute the sum of
    }                               //one quarter
    quarterAverage[i] = sum / 3.0;  //Quarter (i+1) average
}
```



Array Initialization

- Like other data types, it is possible to declare and initialize an array at the same time.

```
int[] number = { 2, 4, 6, 8 };

double[] samplingData = { 2.443, 8.99, 12.3, 45.009, 18.2,
                          9.00, 3.123, 22.084, 18.08 };

String[] monthName = { "January", "February", "March",
                       "April", "May", "June", "July",
                       "August", "September", "October",
                       "November", "December" };
```

number.length → **4**
samplingData.length → **9**
monthName.length → **12**



Variable-size Declaration

- In Java, we are not limited to fixed-size array declaration.
- The following code prompts the user for the size of an array and declares an array of designated size:

```
Scanner scanner = new Scanner(System.in);  
int size;  
int[] number;  
  
System.out.print("Size of an array:");  
size= scanner.nextInt( );  
  
number = new int[size];
```



The For-Each Loop

- This new for loop is available from Java 5.0
- The for-each loop simplifies the processing of elements in a collection
- Here we show examples of processing elements in an array

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

standard for loop

```
int sum = 0;  
  
for (int value : number) {  
    sum = sum + value;  
}
```

for-each loop



For-Each: Key Points to Remember

- A for-each loop supports read access only. The elements cannot be changed.
- A single for-each loop allows access to a single array only, i.e., you cannot access multiple arrays with a single for-each loop.
- A for-each loop iterates over every element of a collection from the first to the last element. You cannot skip elements or iterate backward.



Two-Dimensional Arrays

- Two-dimensional arrays are useful in representing tabular information.

Distance Table (in miles)					
	Los Angeles	San Francisco	San Jose	San Diego	Monterey
Los Angeles	—	600	500	150	450
San Francisco	600	—	100	750	150
San Jose	500	100	—	650	50
San Diego	150	750	650	—	600
Monterey	450	150	50	600	—

Multiplication Table									
	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9
2	2	4	6	8	10	12	14	16	18
3	3	6	9	12	15	18	21	24	27
4	4	8	12	16	20	24	28	32	36
5	5	10	15	20	25	30	35	40	45
6	6	12	18	24	30	36	42	48	54
7	7	14	21	28	35	42	49	56	63
8	8	16	24	32	40	48	56	64	72
9	9	18	27	36	45	54	63	72	81

Tuition Table		
	Day Students	Boarding Students
Grades 1 – 6	\$ 6,000.00	\$ 18,000.00
Grades 7 – 8	\$ 9,000.00	\$ 21,000.00
Grades 9 – 12	\$ 12,500.00	\$ 24,500.00



Declaring and Creating a 2-D Array

Declaration

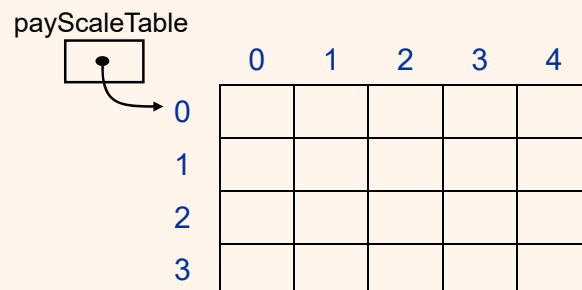
```
<data type> [][] <variable>    //variation 1  
<data type> <variable>[][]    //variation 2
```

Creation

```
<variable> = new <data type> [ <size1> ][ <size2> ]
```

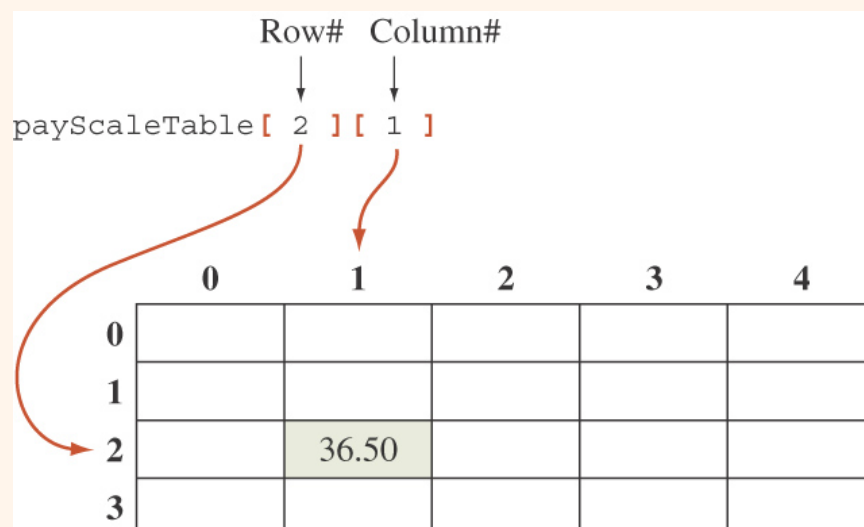
Example

```
double[][] payScaleTable;  
payScaleTable  
    = new double[4][5];
```



Accessing an Element

- An element in a two-dimensional array is accessed by its row and column index.





Sample 2-D Array Processing

- Find the average of each row.

```
double[ ] average = { 0.0, 0.0, 0.0, 0.0 };  
  
for (int i = 0; i < payScaleTable.length; i++) {  
    for (int j = 0; j < payScaleTable[i].length; j++) {  
        average[i] += payScaleTable[i][j];  
    }  
    average[i] = average[i] / payScaleTable[i].length;  
}
```



Java Implementation of 2-D Arrays

- The sample array creation

```
payScaleTable = new double[4][5];
```

is really a shorthand for

```
payScaleTable = new double [4][ ];  
  
payScaleTable[0] = new double [5];  
payScaleTable[1] = new double [5];  
payScaleTable[2] = new double [5];  
payScaleTable[3] = new double [5];
```