Introduction to Arrays

Chris Kiekintveld
CS 2401 (Fall 2010)
Elementary Data Structures and Algorithms
Arrays

- Arguably the most fundamental data structure
  - Other data structures built using arrays
  - Computer memory is like a giant array

- Convenient way to process large amounts of related data
Example: print three integers in reverse order (without array)

```java
public static void main(String[] args) {
    int num1, num2, num3;
    System.out.println("Enter three integers: ");
    num1 = console.nextInt();
    num2 = console.nextInt();
    num3 = console.nextInt();
    System.out.println(num3);
    System.out.println(num2);
    System.out.println(num1);
}
```
Example: print three integers in reverse order (without array)

```java
public static void main(String[] args) {
    int[] num = new int[3];
    System.out.println("Enter three integers:");

    for (int i = 0; i < 3; i++) {
        num[i] = console.nextInt();
    }

    for (int i = 2; i >= 0; i--)
        System.out.println(num[i]);
}
```
Array Definition

- A structured data type with a **fixed** number of components
- Every component is of the same type
- Components are accessed using their relative positions in the array
- In Java, arrays are objects
Example Array

```java
int[] num = new int[5];
```

Figure 9-1  Array num
Example 2

Table:

<table>
<thead>
<tr>
<th>list</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>list[0]</td>
<td>0</td>
</tr>
<tr>
<td>list[1]</td>
<td>0</td>
</tr>
<tr>
<td>list[2]</td>
<td>0</td>
</tr>
<tr>
<td>list[3]</td>
<td>10</td>
</tr>
<tr>
<td>list[4]</td>
<td>0</td>
</tr>
<tr>
<td>list[5]</td>
<td>45</td>
</tr>
<tr>
<td>list[6]</td>
<td>35</td>
</tr>
<tr>
<td>list[7]</td>
<td>0</td>
</tr>
<tr>
<td>list[8]</td>
<td>0</td>
</tr>
<tr>
<td>list[9]</td>
<td>0</td>
</tr>
</tbody>
</table>

**Figure 9-4** Array `list` after the statements `list[3] = 10;`, `list[6] = 35;`, and `list[5] = list[3] + list[6];` execute
Array Syntax

♦ Syntax to declare an array:
  ♦ dataType[] arrayName;
  ♦ arrayName = new dataType[N]
  ♦ dataType[] arrayName = new dataType[N]
  ♦ dataType[] arrayName1, arrayName2;

♦ Syntax to access an array component:
  ♦ arrayName[index]
  ♦ 0 <= index < length of array
Array Initialization During Declaration

double[] sales = {12.25, 32.50, 16.90, 23};

♦ Array size is determined by the number of initial values within the braces  
♦ If an array is declared and initialized simultaneously, do not use the operator new to instantiate the array object
Specifying Array Size During Program Execution

```java
int arraySize;
System.out.print("Enter the size of "+ "the array: ");
arraySize = console.nextInt();
System.out.println();
int[] list = new int[arraySize];
```

Must wait until you know the size to initialize
Array Default Values

What does the following code snippet print?

```java
int[] numList = new int[10];
System.out.println(numList[6]);
```

Arrays are initialized to the default value for the type

- `int`: 0
- `boolean`: false
- `String`: null
Array Initialization

What does the following code snippet print?

```java
int[] numList = new int[10];
Arrays.fill(numList, 15);
System.out.println(numList[6]);
```
A public instance variable length is associated with each array that has been instantiated.

length contains the size of the array.

```java
int[] numList = new int[10];
```

The value of numList.length is 10.
Loops used to step through elements in array and perform operations

```java
int[] list = new int[100];

for (int i = 0; i < list.length; i++)
    //process list[i], the (i + 1)th
    //element of list

for (int i = 0; i < list.length; i++)
    list[i] = console.nextInt();

for (int i = 0; i < list.length; i++)
    System.out.print(list[i] + " ");
```
Determining Largest Element in Array

```java
int[] sales = {5, 12, 14, 11, 19};
maxIndex = 0;

for (int i=1; i<sales.length; i++) {
    if (sales[maxIndex] < sales[i]) {
        maxIndex = i;
    }
}
largestSale = sales[maxIndex];
```
Extermination Exercise

```java
int[] numList = {1, 2.8, 4, 6.7};
double ave = 0;

for (int i=1; i<=numList.length(); i++) {
    ave += numList[i]
}
ave = ave / numList.length;
System.println("Average: " + ave);
```
Extermination Exercise

double[] numList = {1,2.8,4,6.7};
double ave = 0;

for (int i=0; i<numList.length; i++) {
    ave += numList[i];
}
ave = ave / numList.length;
System.out.println("Average: " + ave);
Array Index Out of Bounds

- An array is in bounds if:
  
  \[ 0 \leq \text{index} \leq \text{arraySize} - 1 \]

- If \( \text{index} < 0 \) or \( \text{index} > \text{arraySize} \):
  
  \text{ArrayIndexOutOfBoundsException} \text{ exception is thrown}
Declaring Arrays as Formal Parameters to Methods

- General syntax to declare an array as a formal parameter: `dataType[] arrayName`

```java
public static void arraysAsFormalParameter(int[] listA,
                                           double[] listB, int num)
{
    //...
}

int[] intList = new int[10];
double[] doubleNumList = new double[15];
int number;

arraysAsFormalParameter(intList, doubleNumList, number);
```
int[] listA = {5, 10, 15, 20, 25, 30, 35};
int[] listB = {0, 0, 0, 0, 0, 0, 0, 0};

listB = listA;
System.out.println("Test1: " + listB[3]);

listB[2] = -1;
System.out.println("Test2: " + listA[2]);
The Assignment Operators and Arrays

Figure 9-6 Arrays listA and listB
The Assignment Operators and Arrays (continued)

Figure 9-7  Arrays after the statement `listB = listA;` executes
How do you copy an array?

- Use a for loop
- Use Arrays.copyOf()
  ```java
  int[] copy = Arrays.copyOf(original, original.length);
  ```
- Use System.arraycopy();
  ```java
  int[] copy = new int[original.length];
  System.arraycopy(original, 0, copy, 0, original.length);
  ```
Relational Operators Arrays

```java
if (listA == listB)
...```

- The expression `listA == listB` determines if the values of `listA` and `listB` are the same (refer to the same array)
- To determine whether `listA` and `listB` contain the same elements, compare them component by component
- You can write a method that returns `true` if two `int` arrays contain the same elements
Testing Array Equality

boolean isEqualArrays(int[] firstArray, int[] secondArray)
{
    if (firstArray.length != secondArray.length)
        return false;
    for (int index = 0; index < firstArray.length; index++)
        if (firstArray[index] != secondArray[index])
            return false;
    return true;
}

Check out Arrays.equals()
Arrays of Objects

- Can use arrays to manipulate objects
- Example: Create an array named array1 with N objects of type T:
  
  ```java
  T[] array1 = new T[N]
  ```
- Can instantiate array1 as follows:
  
  ```java
  for(int j=0; j <array1.length; j++)
      array1[j] = new T();
  ```
Array of String Objects

```java
String[] nameList = new String[5];

nameList[0] = "Amanda Green";
nameList[1] = "Vijay Arora";
nameList[2] = "Sheila Mann";
nameList[3] = "Rohit Sharma";
nameList[4] = "Mandy Johnson";
```
Array of String Objects (continued)

Figure 9-9  Array nameList
Arrays of Objects

Clock[] arrivalTimeEmp = new Clock[100];

Figure 9-10 Array arrivalTimeEmp
Instantiating Array Objects

```java
for (int j = 0; j < arrivalTimeEmp.length; j++)
    arrivalTimeEmp[j] = new Clock();
```

*Figure 9-11  Array arrivalTime after instantiating the objects for each component*
Instantiating Array Objects

arrivalTimeEmp[49].setTime(8, 5, 10);

Figure 9-12  Array arrivalTimeEmp after setting the time of employee 49
Arrays and Variable Length Parameter List

- The syntax to declare a variable length formal parameter (list) is:
  
  ```
  dataType ... identifier
  ```
Arrays and Variable Length Parameter List (continued)

```java
public static double largest(double ... numList)
{
    double max;
    int index;
    if (numList.length != 0)
    {
        max = list[0];
        for (index = 1; index < numList.length; index++)
        {
            if (max < numList[index])
                max = numList[index];
        }
    }
    return max;
}
return 0.0;
}```
foreach loop

♦ The syntax to use for loop to process the elements of an array is:

   for (dataType identifier : arrayName) 
      statements

♦ identifier is a variable, and the data type of identifier is the same as the data type of the array components
foreach loop

```java
sum = 0;
for (double num : list)
    sum = sum + num;
```

♦ The `for` statement is read for each `num` in `list`  
♦ The identifier `num` is initialized to `list[0]`  
♦ In the next iteration, the value of `num` is `list[1]`, and so on
Multi-Dimensional Arrays
Multi-Dimensional Arrays

We can have arrays of objects

Arrays are objects…

Can we have an array of arrays?
Why?

♦ Great for storing and manipulating “matrix” data

♦ Examples
  ♦ Board Games
  ♦ Excel Spreadsheets
  ♦ Others?
Two-Dimensional Arrays

- To declare/instantiate a two-dimensional array:

  ```java
dataType[][] name = new dataType[4][3];
```

- To initialize in the declaration:

  ```java
dataType[][] name = {{1,2,3},{4,5,6},
  {7,8,9},{10,11,12}};
```
Accessing 2d Arrays

- To access a component of a two-dimensional array:
  ```java
  arrayName[index1][index2];
  ```
  - `index1` = row position
  - `index2` = column position
Two-Dimensional Arrays

double[][][] sales = new double[10][5];

![Two-dimensional array sales](image)

*Figure 9-14  Two-dimensional array sales*
Accessing Two-Dimensional Array Components

Figure 9-15  sales[5][3]
Example: Seating Chart

- Suppose I want to write a program to store the names of everyone in this course, organized by where they are currently sitting
- Write a piece of code to declare a data structure to store this information
- Initialize the correct element with your name
  - Front row is “0”
  - Left column is “0”
Size of 2d Arrays

dataType[][] name = new dataType[4][3];

♦ How do we get the number of rows in a 2d array?

    numRows = name.length; // 4

♦ How do we get the number of columns?

    numCols = name[0].length; // 3
Processing 2d Arrays

- Three ways to process two-dimensional arrays:
  - Entire array
  - Particular row of array (row processing)
  - Particular column of array (column processing)
- Processing algorithms is similar to processing algorithms of one-dimensional arrays
- Use two for loops
Processing 2d Arrays

Initialization

```java
for (row = 0; row < matrix.length; row++)
    for (col = 0; col < matrix[row].length; col++)
        matrix[row][col] = 10;
```

Print

```java
for (row = 0; row < matrix.length; row++) {
    for (col = 0; col < matrix[row].length; col++) {
        System.out.printf("%7d", matrix[row][col]);
    }
    System.out.println();
}
```
Processing 2d Arrays

Exercise: write code to output the sum of each row of a 2d matrix

Sum by Row

```java
for (row = 0; row < matrix.length; row++) {
    sum = 0;
    for (col = 0; col < matrix[row].length; col++) {
        sum = sum + matrix[row][col];
    }
    System.out.println("Sum of row " + (row + 1) + " = "+ sum);
}
```
Processing 2d Arrays

Exercise: write code to output the sum of each column of a 2d matrix

Sum by Column

```java
for (col = 0; col < matrix[0].length; col++) {
    sum = 0;
    for (row = 0; row < matrix.length; row++) {
        sum = sum + matrix[row][col];
    }
    System.out.println("Sum of column \" + (col + 1) + \" = \" + sum);
}
```
Processing 2d Arrays

Exercise: write code to output the maximum element of each row in a 2d matrix

```java
for (row = 0; row < matrix.length; row++) {
    largest = matrix[row][0];
    for (col = 1; col < matrix[row].length; col++) {
        if (largest < matrix[row][col]) {
            largest = matrix[row][col];
        }
    }
    System.out.println("The largest element of row "+ (row + 1) + "+ largest);}
```
Multi-Dimensional Arrays (continued)
Exercise: Reverse an Array

Write code to reverse the elements of an array

double[] a = {9.3, 1.1, 7.8, 8.9, 3.0};
double[] b = new double[a.length];

for (int i = 0; i < a.length; i++) {
    b[a.length-i-1] = a[i];
}

Reverse an Array
(without declaring a new array)

double[] a = {9.3, 1.1, 7.8, 8.9, 3.0};

int N = a.length;
for (int i = 0; i < N/2; i++) {
    double temp = a[N-i-1];
    a[N-i-1] = a[i];
    a[i] = temp;
}
“Ragged” Arrays

Does every row in a 2d array need to be the same size?

No!

Figure 9-16  Two-dimensional array board
Initializing Ragged Arrays

dataType[][] name = {{1,2},
                  {3,4,5},
                  {6,7,8,9}};

dataType[][] name = new datatype[3][];

name[0] = new dataType[2];
name[1] = new dataType[3];
name[2] = new dataType[4];
Processing Ragged Arrays

- Ragged arrays can be useful, but have a big pitfall
- It is very easy to get `ArrayIndexOutOfBoundsException`

```java
for (row = 0; row < matrix.length; row++) {
    for (col = 0; col < matrix[0].length; col++) {
        System.out.printf("%7d", matrix[row][col]);
    }
    System.out.println();
}
```
Exercise

♦ Write a method to determine if an array is ragged.

```java
boolean isRagged(int[][][] X) {
    for (int i = 1; i < X.length; i++) {
        if (X[i].length != X[0].length) {
            return true;
        }
    }
    return false;
}
```
Multidimensional Arrays

- Can define three-dimensional arrays or n-dimensional arrays (n can be any number)
- Syntax to declare and instantiate array:

  ```java
dataType[][][]...[] arrayName = new
dataType[intExp1][intExp2]...[intExpn];
```

- Syntax to access component:

  ```java
arrayName[indexExp1][indexExp2]...[indexExpn]
```

  - `intExp1, intExp2, ..., intExpn = positive integers`
  - `indexExp1, indexExp2, ..., indexExpn = non-negative integers`
Loops to Process Multidimensional Arrays

double[][][] carDealers = new double[10][5][7];

for (i = 0; i < 10; i++)
    for (j = 0; j < 5; j++)
        for (k = 0; k < 7; k++)
            carDealers[i][j][k] = 10.00;
Arrays in Memory

♦ What does a multi-dimensional array look like in computer memory?
♦ How is the space allocated?

```java
dataType[][] name = {{1,2,3},
                   {4,5,6},
                   {7,8,9},
                   {10,11,12}};
```
Exercise

- Can we create a class that implements a 2d array using a 1d array?
2d Array Using 2d

```java
public class Array2d {
    Object[] array;
    int nRows;
    int nCols;

    public Array2d(int nRows, int nCols) {
        this.nRows = nRows;
        this.nCols = nCols;
        array = new Object[nRows * nCols];
    }
}
```
private computeIndex(int row, int col) {
    return (nCols * row) + col;
}

public void set(int row, int col, Object value) {
    array[computeIndex(row, col)] = value;
}

public Object get(int row, int col) {
    return array[computeIndex(row, col)];
}
Arrays and Memory

What is the problem with the following code?

```java
int N = 1000;
int[] a = new int[N*N*N*N];
```
Exercise

* Write a program that counts the number of times each digit between 0 and 9 occurs in a 2d array.

```java
int[] counts = new int[10];
countOcc(myArray, counts);

public void countOcc(int[][] array, int[] counts) {
}
```
int[] counts = new int[10];
countOcc(myArray, counts);

public void countOcc(int[][] a,
                     int[] counts) {
    for (int i = 0; i < a.length; i++) {
        for (int j = 0; j < a[i].length; j++) {
            if (a[i][j] < 0) continue;
            if (a[i][j] > 9) continue;
            counts[a[i][j]]++;
        }
    }
}
Exercise: Sudoku
Exercise: Sudoku
Group Exercise

- Write a program to determine if a 2d input matrix is a valid Sudoku solution
- Hint: write a method similar to the previous example, and use this method repeatedly
- Solution posted on website