Java: Introduction to Arrays

Instructor: Nihshanka Debroy
(Notes borrowed from Tammy Bailey)
Arrays

- Aggregate data type: stores items/data of same type in a block of consecutive memory locations
- Individual locations in array: elements
- Size of an array: # elements it is able to store
- Analogies
  - Books in a bookshelf
  - CD rack with slots
- Simplifies naming
Analogy

- Post office mailboxes
  - Purpose is to store data of type “mail”
  - One main unit divided into fixed number of individual locations
    - mailboxes
  - Mailboxes are indexed consecutively by number
  - Contents of a mailbox are accessed by box number
Using Arrays

- Arrays have **fixed size**
- **Subscript or index** to access element (**start from 0** in Java)
  
  ```java
  A[10]=20;
  System.out.println(A[10]);
  ```

- Work well with loops
  ```java
  int i;
  int sum=0;
  for(i=1;i<=10;i++)
  {
    sum=sum+ A[i];   // A is an array
  }
  ```
An array of integers

- All elements in the array are of type integer
- The length of the array is 10
- The array is indexed from 0 to 9
An array of integers

- All elements in the array are of type integer
- The length of the array is 10
- The array is indexed from 0 to 9
- The element at index 0 is 3
An array of integers

- All elements in the array are of type integer
- The length of the array is 10
- The array is indexed from 0 to 9
- The element at index 7 is 55
Java arrays

- Arrays have names, types, and size
- Arrays must be declared and their size must be specified before you can use them in a program
- The Java statement

  ```java
  int A;
  ```

  declares a single integer variable named A
- The Java statement

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

  declares an array variable A that holds 10 integer values
- To declare an array named D to hold 20 double values:

  ```java
  double D[] = new double[20];
  ```
Creating Arrays

- **Declaration**
  ```java
double weights[];
```
- **Definition**
  ```java
weights = new double[50];
```
- **Combine**
  ```java
double weights[] = new double[50];
```

```java
int num[] = new int[6];
```

```
```

```
```
Assigning array values

- Declare an array of 5 integers
  
  ```java
  int[] A = new int[5];
  ```

- Assign values to array elements
  
  ```java
  A[0] = 4;
  ```

- Variable initializer syntax: when you know what values you want to store in the array
  
  ```java
  int[] A = {4, -6, -2, 11, 12};
  ```
Array access

- Declare an array \( A \) containing the integers 2, 4, \(-9\), 0, 2

\[
\text{int}[]\ A = \{2, 4, -9, 0, 2\};
\]

- What is the value of \( A[0] \)? (2)
- What is the value of \( A[1] \)? (4)
- What is the value of \( A[3] \)? (0)
- What is the value of \( A[5] \)? (error)
- What is the value of \( A[10] \)? (error)
Array length property

- The number of elements in an array is given by the length property.
- Length of array is determined when array is created – either explicitly specified or comes from the length of the {...} initialization list.
- The length of an array named A is given by A.length.
- For example, if we declare arrays A and B as follows:
  ```java
  int[] A = {2, 4, -9, 0, 2};
  int[] B = {-1, 7, 3, 3, 3, 6, 9};
  ```
  then A.length returns 5 and B.length returns 7.
Operations on array elements

```
int[] A = {2, 4, -9, 0, 2};
int[] B = {-1, 7, 3, 3, 3, 6, 9};
```

- All elements in arrays A and B are of type `int`
  - `int w = A[0];` \(w = 2\)
  - `int x = 5+B[2];` \(x = 8\)
  - `int y = A[1]*B[5];` \(y = 24\)
  - `int z = B[6]-A.length;` \(z = 4\)
  - `int p = A+B;` \(\text{error}\)
  - `int q = B[3]*1.5;` \(\text{error}\)
  - `int r = 2*B;` \(\text{error}\)
Manipulating array values

- Consider the following code fragment:

```java
int[] F = new int[6];
F[0] = 1;
F[1] = 1;
```

- What values will be stored in F?

Manipulating array values

- Consider the following code fragment:
  ```java
  int[] F = new int[6];
  F[0] = 1;
  F[1] = 1;
  ```

- What values will be stored in F?

  Answer: 1 1 2 3 5 8
Arrays in Loops

// creating an integer array of size 10
int[] myArray = new int[10];

// for loop
for(int i=0; i < myArray.length; i++)
{
    myArray[i] = i;
    // myArray[0]=0, myArray[1]=1, and so on.
}
Arrays and subroutines

- Subroutine `sumArray` computes the sum of the elements in an array of doubles

```java
double sumArray(double[] A)
{
    double sum = 0.0;
    for(int k = 0; k < A.length; k++)
        sum = sum + A[k];
    return sum;
}
```
Arrays with multiple dimensions may be declared and used:

```java
int[][][] A = new int[5][6];
int[][][][] B = new int[4][2][2];
```

- Number of square bracket pairs is dimension of array:
  - `A` is a two-dimensional array
  - `B` is a three-dimensional array

- Two-dimensional (2D) arrays:
  - by Java convention, in a 2D array the first index indicates the row and the second the column
  - we can visualize a 2D array as a grid or table of elements
2D arrays

- A 2D array is basically a 1D array of 1D arrays
  - these are the rows of the array
  - each row is stored in a separate block of consecutive memory locations
- If we declare array A as
  
  ```java
  int[][] A = new int[5][6];
  ```

  then
  - A[k] is a 1D array, the k\(^{th}\) row of A
  - A.length is the number of rows in A
  - A[k].length is the length of the k\(^{th}\) row of A
Example

- Let \( A \) denote a 2D array of integers with 5 rows and 6 columns, and suppose \( A \) contains the elements as shown below:

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>5</td>
<td>0</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

- What is \( A \).length?
- What is \( A[2].length \)?
- What is \( A[2][4] \)?
- What is \( A[4][0] \)?
- What is \( A[3][5] \)?
- What is \( A[5][6] \)?
Code Example
Applications

- 2D arrays are useful when data can be represented by a grid of fixed dimensions.
- Often used to represent tables, matrices, images, and game boards.
- Examples of games include checkers, chess, tic-tac-toe, crosswords, and mazes.