CS5000: Foundations of Programming

Mingon Kang, PhD
Computer Science, Kennesaw State University
Arrays

- A data structure for a collection of data that is all of the same data type.
- The data type can be either primitive data types or classes.
- Multiple values, but one variable name
Creating Arrays

- Creating one-dimensional arrays

```
DataType[] array_name;
DataType[] array_name = new DataType[LENGTH];
```

E.g.
```
double[] score;
Score = new double[5];
or
double[] score = new double[5];
```
Creating Arrays

- Arrays in Java are of fixed size that is specified when they are declared.
- Can specify the size of the array using a variable.
  - Only non-negative integers

Scanner sc = new Scanner(System.in);
int nSize = sc.nextInt();
double[] score = new double[nSize];
Creating Arrays

Example?

```java
String[] strNames = new String[100];
Student[] myStudents = new Student[30];
Vehicle[] myCars = new Vehicle[100];
```
Arrays are automatically initialized to null

Can be initialized when declared

- The array size is automatically set to the number of the given values.

```java
int[] nIDs = {500100, 500101, 500102};
```
Accessing Arrays

- Indices of the element in an array
  - The indexed variables are numbered starting with 0

```java
char[] alphabets = new char[26];
```

<table>
<thead>
<tr>
<th>65</th>
<th>66</th>
<th>67</th>
<th>68</th>
<th>69</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘A’</td>
<td>‘B’</td>
<td>‘C’</td>
<td>‘D’</td>
<td>‘E’</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

← Values stored in the array memory
← Values we assign
← Indices of the array
Accessing Arrays

- Access an element value of the array

```java
char[] alphabets = new char[26];
// Assignment
alphabets[0] = 'A';
alphabets[1] = 'B';
// Access
System.out.println (alphabets[0]); // 'A'
```
Accessing Arrays

- For loop is ideally suited for initializing or accessing the elements of arrays.

```java
// Initialization
for (int i = 0; i < 26; i++)
    alphabets[i] = (char)('A' + i);

// Access
for (int i = 0; i < 26; i++)
    System.out.println(alphabets[i]);
```
The ‘length’ Instance variable

- When an array is created, one instance variable named ‘length’ is automatically set equal to its size.

```java
int[] IDs = {500100, 500101, 500102};
IDs.length // has a value of 3.
```
An array of characters is not an object of the class `String`

```
char[] a = {'A', 'B', 'C'};
String s = a; // wrong

// convert char[] to a string
// see constructors of the String class.
String s = new String(a);
```
Arrays and References

- A variable of an array type holds a reference.
  - Arrays are objects
  - A variable of an array type holds the address of where the array objects is stored in memory
Arrays as Parameters

- Arrays can be used as parameters to methods
- Array parameters behave like objects of a class
  - A method can change the values stored in the indexed variables of the array.
  - A method with an array parameter have to specify the data type of the array
Arrays as Parameters

```java
public class Sample{
    public static void example(double[] a){
        a[0] = 1;
    }

    public static void main(String[] args){
        double[] a = new double[5];
        Sample.example(a);
        System.out.println(a[0]);
    }
}
```
Array assignment

- Assignment operator =

```java
double[] a = new double[5];
double[] b = new double[5];
a = b;  // what is it doing?
```

- Copy the memory address
Copy each element one by one using for loop

```java
for (i = 0; (i < a.length) && (i < b.length); i++)
    b[i] = a[i];
```
Array comparison

- Equality operator `==`

```java
double[] a = new double[5];
double[] b = new double[5];
a == b; // what is it doing?
```

- Compare whether they are stored in the same location in the memory
Main method’s arguments

- The main method has a parameters for an array of String

```java
public class Hello {
    
    public static void main(String[] args) {
        System.out.println(args[0] + " " + args[1] + args[2]);
    }
}
```
Main method’s arguments

- If we run the program by `java Hello Mingon Kang`
  - Set
    - `args[0]` to “Hello”
    - `args[1]` to “Mingon”
    - `args.length` to 3
There are a number of collection data in Java
  - Set, List, Queue, Map, and so on..
  - Normally, cannot be accessed by index

Enhanced for loop ➔ for each loop
“for each” Loop

- **Syntax**

  ```
  for (ArrayBaseType VariableName : ArrayName)
      statement
  ```

  - VariableName must be declared within the for each loop, not before
“for each” Loop

Examples

// using for loop
for (int i = 0; i < a.length; i++)
    a[i] = 0;

// using for each loop
for (double e : a)
    e = 0;
Multidimensional Arrays

- Two-dimensional arrays
- Three-dimensional arrays
- Even higher-dimensional arrays
  - Simply use as many square brackets as there are indices
Practical usages of 2D arrays

- Often data come naturally in the form of a table such as spreadsheet.

**Grades**

<table>
<thead>
<tr>
<th>Student ID</th>
<th>HW1</th>
<th>HW2</th>
<th>EX1</th>
<th>EX2</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>10001001</td>
<td>100</td>
<td>80</td>
<td>100</td>
<td>87</td>
<td>90</td>
</tr>
<tr>
<td>10001002</td>
<td>60</td>
<td>90</td>
<td>80</td>
<td>78</td>
<td>96</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

**Scientific Data**

<table>
<thead>
<tr>
<th>Individual</th>
<th>DNA1</th>
<th>DNA2</th>
<th>DNA3</th>
<th>DNA4</th>
<th>....</th>
</tr>
</thead>
<tbody>
<tr>
<td>PN1</td>
<td>A</td>
<td>C</td>
<td>A</td>
<td>G</td>
<td>A</td>
</tr>
<tr>
<td>PN2</td>
<td>A</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>G</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Practical usages of 3D, 4D arrays

- Modeling 3D objects (X-Y-Z)
  - 3D graphic tool, 3D game

- Movie file
  - X-Y axis for a scene and Z axis for time

- 3D Movie File
  - X-Y-Z axis for a scene and W axis for time
Multidimensional Arrays

- Syntax

Example)

double[][] grades = new double[100][10];
int[][] images = new int[100][200];
Person[][] p = new Person[10][100];

double[][][] grades = new double[10][20][30];
A two-dimensional array is an array of arrays

```
char[][] a = new char[5][12];
```

- The variable ‘a’ contains a reference to a one-dimensional array of size 5 with a base type of char[].
- ‘a[i]’ contains a reference to a one-dimensional array of size 12.
A two-dimensional array is an array of arrays.

```java
char[][] a = new char[5][12];
```

Code that fills the array is not shown.

Blank entries contain the space (blank) character.
The instance variable ‘length’ in multidimensional arrays

```java
char[][] a = new char[5][12];
a.length // 5
a[0].length // 12
```
Ragged Arrays

- Each row in a two-dimensional array need not have the same number of elements → Ragged Arrays

```java
double[][] a = new double[5][];
a[0] = new double[5];
a[1] = new double[10];
a[2] = new double[4];
```