Lecture - 07

Declarations: Arrays

Arpit Rana

10th Feb 2022

Arrays

An array is a ***linear*** data structure -

- that defines an ***indexed*** collection of
- a ***fixed*** number of
- ***homogeneous*** data elements

Arrays

In Java, arrays are ***objects*** which can be of

- Primitive data types (all elements are of a specific primitive type), or
- Reference type (all elements are references of a specific reference type)
    - Each reference can denote object of this reference type or its subtypes.

Arrays

Each array object has a **`public final`** field called **`length`**

- **`length`** specifies the number of elements that an array can accommodate.
- The first element is always at index ***0*** and the last element at index ***n - 1***, where ***n*** is the value of the **`length`** field.

Arrays

Array Dimensions

- SImple arrays are *one-dimensional arrays*—that is, a simple list of values.
- Since arrays can store reference values, the objects can also be array objects, i.e. *multidimensional arrays are array of arrays*.

Arrays

Declaring Array Variables

- *One-dimensional array* declaration

element_type [ ] *array_name* ;

or

element_type array_name [ ] ;

- Note that array size is not specified,
  - it means that the variable *array_name* can store the reference of an array of *element_type* of any size.

Arrays

Declaring Array Variables

- Note that array declaration simply declares a reference that can refer to an array object, it does not create an array.

```
int anIntArray[], oneInteger;
Pizza[] mediumPizzas, largePizzas;
```

- `anIntArray` can refer to the `int` array.
- `oneInteger` is simply an `int` variable.
- `mediumPizzas` and `largePizzas` can denote an array of `Pizza` objects.

Arrays

Constructing an Array

- An array can be constructed for a fixed number of elements of a specific type, using the **`new`** operator.
- The reference value of the resulting array can be assigned to an array variable of the corresponding type.

array_name = `new` *element_type*`[`*array_size*`];`

- The minimum value of `array_size` is `0`; in other words zero-length arrays can be constructed in Java.
- If the array size is negative, a `NegativeArraySizeException`is thrown at runtime.

Arrays

Constructing an Array

- The array declaration and construction can be combined.

$$element\_type_1\,[\,]\ array\_name = \texttt{new}\ element\_type_2\,[array\_size]\,;$$

- For example,

```
int[] anIntArray = new int[10];                    // Default element value: 0
Pizza[] mediumPizzas = new Pizza[5];               // Default element value:
null
```

Arrays

Initializing an Array

- Java provides the means of declaring, constructing, and explicitly initializing an array in one declaration statement:

$$element\_type[\,]\ array\_name = \{\ array\_initialize\_list\ \};$$

- For example,

```
int[] anIntArray = {13, 49, 267, 15, 215};

Pizza[] pizzaOrder = { new Pizza(), new Pizza(), null };

// Array with 4 String objects:
String[] pets = {"crocodiles", "elephants", "crocophants", "elediles"}; //
(1)

// Array of 3 characters:
char[] charArray = {'a', 'h', 'a'};    // (2) Not the same as "aha"
```

Arrays

Using an Array

- The array object is referenced by the array name, but individual array elements are accessed by specifying an index with the [ ] operator.

array_name [*index_expression*]

- *`index_expression`* value should be promotable to an `int` value; otherwise, a compile-time error is flagged.
- The upper bound of an array index is 1 less than the array size—that is, `array_name.length-1`.
- If the index value is less than 0, or greater than or equal to `array_name.length`, an `ArrayIndexOutOfBoundsException` is thrown.

Arrays

```
public class Trials {
  public static void main(String[] args) {
    // Declare and construct the local arrays:
    double[] storeMinimum = new double[5];                  // (1)
    double[] trialArray = new double[15];                   // (2)
    for (int i = 0; i < storeMinimum.length; ++i) {         // (3)
      // Initialize the array.
      randomize(trialArray);

      // Find and store the minimum value.
      storeMinimum[i] = findMinimum(trialArray);
    }

    // Print the minimum values:                               (4)
    for (double minValue : storeMinimum)
      System.out.printf("%.4f%n", minValue);
  }

  public static void randomize(double[] valArray) {         // (5)
    for (int i = 0; i < valArray.length; ++i)
      valArray[i] = Math.random() * 100.0;
  }

  public static double findMinimum(double[] valArray) {  // (6)
    // Assume the array has at least one element.
    double minValue = valArray[0];
    for (int i = 1; i < valArray.length; ++i)
      minValue = Math.min(minValue, valArray[i]);
    return minValue;
  }
}
```

Arrays

Anonymous Arrays

- Neither the name of the arrays nor the size is specified while creating anonymous arrays.

```
new element_type[]  { array_initialize_list }

new int[] {3, 5, 2, 8, 6}

int[] daysInMonth;
daysInMonth = {31, 28, 31, 30, 31, 30,
               31, 31, 30, 31, 30, 31};                    // Compile-time
error
daysInMonth = new int[] {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}; //
OK
```

Arrays

Anonymous Arrays

- Neither the name of the arrays nor the size is specified while creating anonymous arrays.

```
public class AnonArray {
  public static void main(String[] args) {
    System.out.println("Minimum value: " +
        findMinimum(new int[] {3, 5, 2, 8, 6}));                    // (1)
  }

  public static int findMinimum(int[] dataSeq) {                    // (2)
    // Assume the array has at least one element.
    int min = dataSeq[0];
    for (int index = 1; index < dataSeq.length; ++index)
      if (dataSeq[index] < min)
        min = dataSeq[index];
    return min;
  }
}
```

Arrays

Multidimensional Arrays

- In Java, array of arrays can be defined as follows.

element_type array_name `[] [] . . . [];`

or

element_type `[] [] . . . []` *array_name;*

- For example,

```
Pizza[][] pizzaGalore = {
  { new Pizza(), null, new Pizza() },     // 1. row is an array of 3 elements.
  { null, new Pizza()},                   // 2. row is an array of 2 elements.
  new Pizza[1],                           // 3. row is an array of 1 element.
  {},                                     // 4. row is an array of 0 elements.
  null                                    // 5. row is not constructed.
};
```

Arrays

Multidimensional Arrays

- For example,

```
double[][] matrix = new double[3][];      // Number of rows.

for (int i = 0; i < matrix.length; ++i)
  matrix[i] = new double[i + 1];          // Construct a row.
```

Arrays

Multidimensional Arrays

```
public class MultiArrays {

  public static void main(String[] args) {
    // Declare and construct the M X N matrix.
    int[][] mXnArray = {                                          // (1)
        {16,  7, 12}, // 1. row
        { 9, 20, 18}, // 2. row
        {14, 11,  5}, // 3. row
        { 8,  5, 10}  // 4. row
    }; // 4 x 3 int matrix

    // Find the minimum value in a M X N matrix:
    int min = mXnArray[0][0];
    for (int i = 0; i < mXnArray.length; ++i)                     // (2)
      // Find min in mXnArray[i], in the row given by index i:
      for (int j = 0; j < mXnArray[i].length; ++j)                // (3)
        min = Math.min(min, mXnArray[i][j]);

    System.out.println("Minimum value: " + min);
  }
}
```

Next lecture -
Access Control