1495 UNIVERSITY OF ABERDEEN

JC2002 Java Programming

Lecture 6: Arrays and array lists

What are arrays?

- An array is a group of variables (called elements or components), containing values that all have the same type
- Arrays are objects (*reference types*) , whereas the elements of an array can be either *primitive types* or *reference types* (including arrays)
    - Remember: the primitive types are `boolean`, `byte`, `char`, `short`, `int`, `long`, `float` and `double`; all the other types are *reference types*
- Arrays *remain the same length* once they're created

Declaring and creating arrays

- Array objects
  - You specify the element type and the number of elements in an *array-creation expression*, which returns a reference that can be stored in an array variable
- Declaration and array-creation expression for an array of 12 `int` elements:

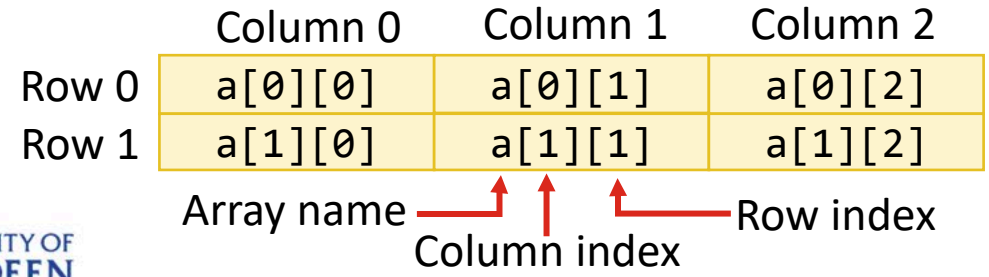
```
int[] c = new int[12];
```

- It can also be performed in two steps as follows:

```
int[] c; // declare the array variable
c = new int[12]; // create the array
```

Multidimensional arrays

- Java does *not* support multidimensional arrays directly
  - To achieve the same effect, we can specify one-dimensional arrays whose elements are also one-dimensional arrays, etc.
- Two-dimensional arrays are often used to represent tables of values with data arranged in rows and columns
  - An array with *m* rows and *n* columns is called an *m-by-n array*
- Each table element is identified with two indices
  - By convention, the first index is the row and the second is the column



Multidimensional arrays can have ***more than*** two dimensions!

Array example 1

- One-dimensional array, initialize the elements of an array to default values of zero

```
1   public class InitArray1 {
2         public static void main(String[] args) {
3               // declare variable array and initialize it with an array object
4               int[] array = new int[10]; // create the array object
5
6               System.out.printf("%s%8s%n", "Index", "Value"); // column headings
7
8               // output each array element's value
9               for (int counter = 0; counter < array.length; counter++) {
10                    System.out.printf("%5d%8d%n", counter, array[counter]);
11              }
12        }
13  }
```

Array example 2

- One-dimensional array, initialise the elements of an array with an array initialiser

```
1   public class InitArray2 {
2         public static void main(String[] args) {
3               // initializer list specifies the initial value for each element
4               int[] array = {32, 27, 64, 18, 95, 14, 90, 70, 60, 37};
5
6               System.out.printf("%s%8s\n", "Index", "Value"); // column headings
7
8               // output each array element's value
9               for (int counter = 0; counter < array.length; counter++) {
10                    System.out.printf("%5d%8d%n", counter, array[counter]);
11              }
12        }
13  }
```

Array example 3

- One-dimensional array, compute the sum of the elements of an array

```
1   public class SumArray{
2         public static void main(String[] args) {
3         int[] array = {87, 68, 94, 100, 83, 78, 85, 91, 76, 87};
4         int total = 0;
5
6         // add each element's value to total
7         for (int counter = 0; counter < array.length; counter++) {
8               total += array[counter];
9         }
10
11        System.out.printf("Total of array elements: %d%n", total);
12        }
13  }
```

Array example 4 (1)

- One-dimensional array, passing arrays and individual array elements to methods

```
1   public class PassArray {
2         // main creates array and calls modifyArray and modifyElement
3         public static void main(String[] args) {
4               int[] array = {1, 2, 3, 4, 5};
5
6               System.out.printf(
7                     "Effects of passing reference to entire array:%n" +
8                     "The values of the original array are:%n");
9
10              // output original array elements
11              for (int value : array) {
12                    System.out.printf(" %d", value);
13              }
```

Array example 4 (2)

```
14              modifyArray(array); // pass array reference
15              System.out.printf("%n%nThe values of the modified array are:%n");
16
17              // output modified array elements
18              for (int value : array) {
19                   System.out.printf(" %d", value);
20              }
21
22              System.out.printf(
23                   "%n%nEffects of passing array element value:%n" +
24                   "array[3] before modifyElement: %d%n", array[3]);
25
26              modifyElement(array[3]); // attempt to modify array[3]
27              System.out.printf(
28                   "array[3] after modifyElement: %d%n", array[3]);
29              }
```

Array example 4 (3)

```
30        // multiply each element of an array by 2
31        public static void modifyArray(int[] array2) {
32              for (int counter = 0; counter < array2.length; counter++) {
33                    array2[counter] *= 2;
34              }
35        }
36
37        // multiply argument by 2
38        public static void modifyElement(int element) {
39              element *= 2;
40              System.out.printf(
41                    "Value of element in modifyElement: %d%n", element);
42        }
43  }
```

Array example 4 (4)

- Program output

```
Effects of passing reference to entire array:
The values of the original array are:
 1 2 3 4 5

The values of the modified array are:
 2 4 6 8 10

Effects of passing array element value:
array[3] before modifyElement: 8
Value of element in modifyElement: 16
array[3] after modifyElement: 8
```

Array example 5 (1)

- Initialising two-dimensional arrays

```
1   public class InitArray {
2        // create and output two-dimensional arrays
3        public static void main(String[] args) {
4             int[][] array1 = {{1, 2, 3}, {4, 5, 6}};
5             int[][] array2 = {{1, 2}, {3}, {4, 5, 6}};
6
7             System.out.println("Values in array1 by row are");
8             outputArray(array1); // displays array1 by row
9
10            System.out.printf("%nValues in array2 by row are%n");
11            outputArray(array2); // displays array2 by row
```

Array example 5 (2)

```
12
13        // output rows and columns of a two-dimensional array
14        public static void outputArray(int[][] array) {
15             // loop through array's rows
16             for (int row = 0; row < array.length; row++) {
17                  // loop through columns of current row
18                  for (int column = 0; column < array[row].length;
19                  column++) {
20                       System.out.printf("%d ", array[row][column]);
21                  }
22
23                  System.out.println();
24             }
25        }
26   }
```

Array example 5 (3)

- Program output

```
Values in array1 by row are
1 2 3
4 5 6

Values in array2 by row are
1 2
3
4 5 6
```

Class ArrayList

- Arrays do not change their size at execution time to accommodate additional elements
  - If you need to increase array size during execution, you need to declare and initialize it again
- For arrays with dynamic size, you can use class `ArrayList<T>` in package `java.util`
  - Note that T is a placeholder for the type of element stored in the `ArrayList` object. Classes with this kind of placeholder that can be used with any type are called *generic classes*; we will discuss more about generic classes later in this course

ArrayList methods

Method	Description
`add`	Adds an element to the end or at the specific index of the `ArrayList`.
`clear`	Removes all the elements from the `ArrayList`.
`contains`	Returns `true` if the `ArrayList` contains the specified element, otherwise returns `false`.
`get`	Returns the element at the specified index.
`indexOf`	Returns the index of the first occurrence of the specified element in the `ArrayList`.
`remove`	Removes the first occurrence of the specified value or the element at the specified index.
`size`	Returns the number of elements stored in the `ArrayList`.
`trimToSize`	Trims the capacity of the `ArrayList` to its current size.

ArrayList example (1)

- Generic `ArrayList<E>` example

```
1   import java.util.ArrayList;
2
3   public class ArrayListCollection {
4        public static void main(String[] args) {
5             // create a new ArrayList of Strings with an initial capacity of 10
6             ArrayList<String> items = new ArrayList<String>();
7
8             items.add("red"); // append an item to the list
9             items.add(0, "yellow"); // insert "yellow" at index 0
10
11       // header
12       System.out.print(
13            "Display list contents with counter-controlled loop:");
```

ArrayList example (2)

```
14        // display the colors in the list
15        for (int i = 0; i < items.size(); i++) {
16              System.out.printf(" %s", items.get(i));
17        }
18
19        // display colors using enhanced for in the display method
20        display(items,
21              "\nDisplay list contents with enhanced for statement:");
22
23        items.add("green"); // add "green" to the end of the list
24        items.add("yellow"); // add "yellow" to the end of the list
25        display(items, "List with two new elements:");
26
27        items.remove("yellow"); // remove the first "yellow"
28        display(items, "Remove first instance of yellow:");
29
30        items.remove(1); // remove item at index 1
31        display(items, "Remove second list element (green):");
```

ArrayList example (3)

```
32          // check if a value is in the List
33          System.out.printf("\"red\" is %sin the list\n",
34          items.contains("red") ? "": "not ");
35
36          // display number of elements in the List
37          System.out.printf("Size: %s%n", items.size());
38          }
39
40          // display the ArrayList's elements on the console
41          public static void display(ArrayList<String> items, String header) {
42          System.out.printf(header); // display header
43
44          // display each element in items
45          for (String item : items) {
46          System.out.printf(" %s", item);
47          }
48          System.out.println();
49          }
50  }
```

ArrayList example (4)

- Program output

```
Display list contents with counter-controlled loop: yellow red
Display list contents with enhanced for statement: yellow red
List with two new elements: yellow red green yellow
Remove first instance of yellow: red green yellow
Remove second list element (green): red yeallow
"red" is in the list
Size: 2
```

Summary

- Version control is essential for program development in collaboration with others
  - Allows merging changes by different persons as well as rolling back to earlier versions of the software
- Basic syntax and structure of Java programs introduced
  - Writing simple console programs displaying data and taking input from users using the keyboard
  - Using basic arithmetic operations
  - Implementing conditional statements and loops
  - Defining and using arrays and `ArrayLists`

Questions, comments?