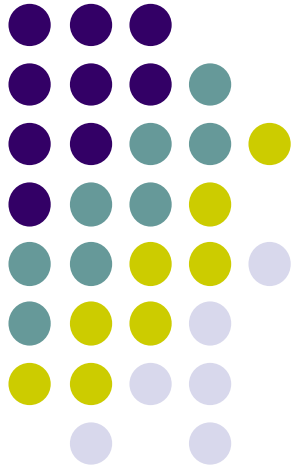
WIX1002
Fundamentals of Programming

Chapter 5

Arrays



Contents

- Introduction
- Multidimensional Arrays
- Applications
- Bubble Sort
- Linear Search
- Binary Search

Introduction

- An array is a **sequence** of values of the **same type**.
- It is a data structure that used to process **a collection of data** that is from the same type.
- An array is a group of **contiguous memory locations** that all have the **same name and the same type**. It is an ordered list of values and each value has a **numeric index** (subscript)
- An array of size N is indexed from **zero to N-1**
- In Java, an array is a **reference data type**.
- The **new** operator is used to construct the array.

Introduction

- **Array Declaration**

  typeofArray[] nameOfArray = new typeofArray[length];

  char [] line = new char[100];

  double [] score = new double[20];

- When declare an array, each element of the array receives **a default value zero for the numeric data type, false for boolean and null for references or String**.
- **Array Initialisation**
  - int[] number = {1,2,3,4,5,6};
  - String[] name = {"one", "two", "three"};

Introduction

- A particular value in an array is referenced using the array name followed by the index (subscript) in brackets.
- The first element in every array start with 0.
  - arrayName[0] refer to the first element
  - arrayName[3] refers to the 4^{th} value in the array.
- Every array in Java knows its own length.
  - **arrayName.length** - determine the length of the array
  - Java will throw an exception if an array index is **out of bounds**.
- Array elements are not limited to primitive data type but also array of object.

Introduction

- To display the element of the array
  - for(int i=0; i<arrayName.length; i++)
  - System.out.println(arrayName[i]);

- Or
  - for(int value : arrayName)
  - System.out.println(value);
  - However, this method can’t be used to modify the value of the element of the array.

Update elements of the array

```
Random r = new Random();
final int MAX=100;
int size = 20;
int[] num = new int[size];

// create an array with random number
for(int i=0; i<num.length; i++) {
    num[i] = r.nextInt(MAX+1);
}

for(int i=0; i<num.length; i++) {
    System.out.print(num[i] + " ");
}
System.out.println();
```

Exercise

- Create an int *num* array with a size of 100
  - Assign the value of 12 to the first item
  - Assign the value of 89 to the last item
- Create a char *symbol* array with the following item
  - {'$', '%', '+', '-’}
  - Change the ‘+’ item to ‘*’
  - Print all of the elements of the array using for loop.

Exercise

- Display the number of female student from a random list of 100 students.
  - Create an array with 100 students and randomly assign the students gender with male/female.
  - Declare a char gender array with ‘M’ and ‘F’.
  - Calculate how many female students in the array

Exercise

- Given the following
  - String str = "Ang,Tan,Fong,Ahmad,Ali";
  - Split the string with the following output

```
run:
Ang
Tan
Fong
Ahmad
Ali
```

Multidimensional Arrays

- We can declare arrays with more than one index in Java.
- **Multidimensional Arrays Declaration**

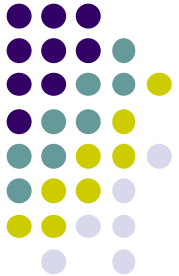
  typeofArray[] … [] nameOfArray = new
  typeofArray[length_i] … [length_n];

  double [][] scoreTable = new double[4][5];

  //4 rows 5 columns
- Multidimensional Arrays with two subscripts are often used to represent tables of values consisting of information arranged in rows and columns.
- **Multidimensional Arrays Initialisation**

  int [][]matrix = { { 1, 2}, { 4, 5} };

Multidimensional Arrays



- **Ragged Arrays**
  - Different rows of the array consists of different columns

    int [][]num = new int[3][];

    num[0] = new int[3];

    num[1] = new int[10];

    num[2] = new int[5];

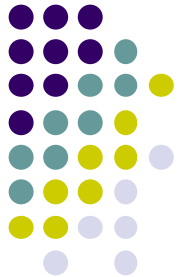
Applications

- Summing elements of an array

```
int a[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
for ( int i = 0; i < a.length; i++ )
        total += a[ i ];
```

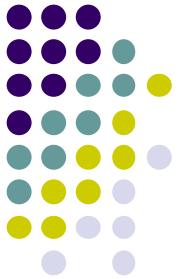
- Sorting is a technique to arrange elements in same order. Sorting data is one of the common feature that can apply on arrays. The common sorting technique is **Bubble Sort**.
- Searching is a technique to locate an item given its key. Searching locating a particular element value in the array. The common searching techniques are **Linear Search** and **Binary Search**.

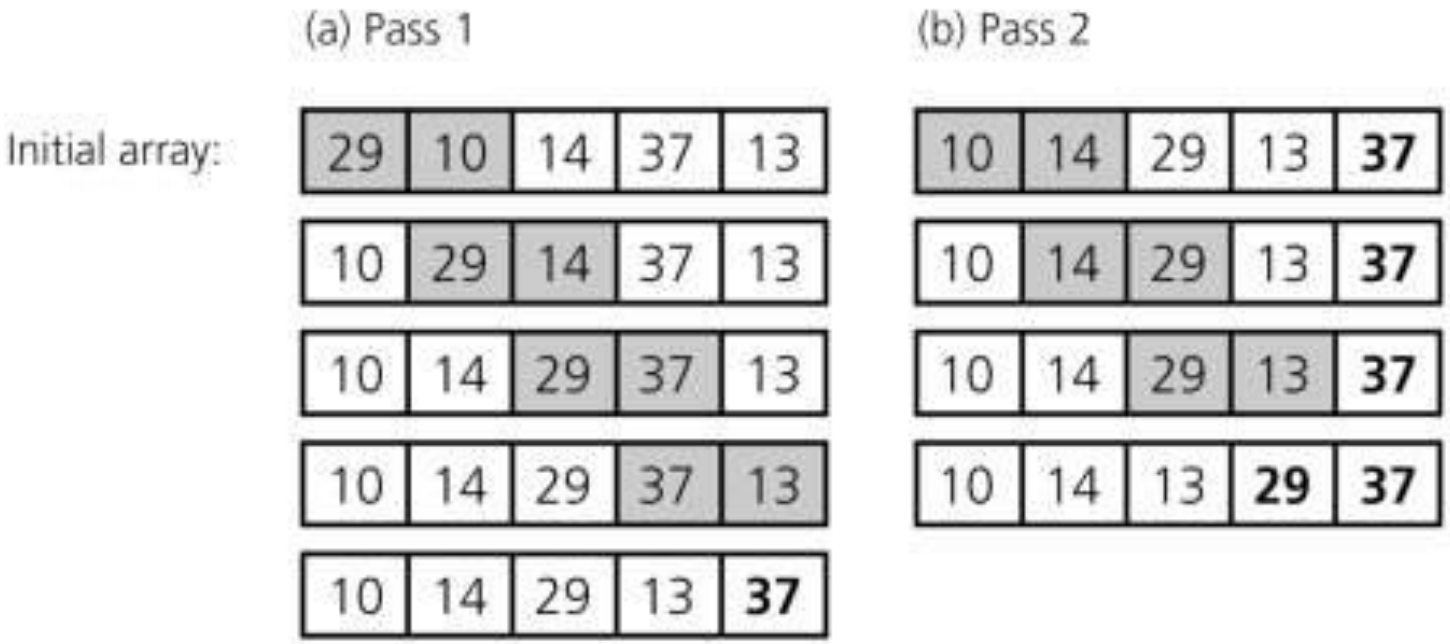
BubbleSort



- The bubble sort uses **nested loops** to make several passes through the array. Each pass compares successive pairs of elements.
- If a pair is in increasing order, the bubble sort leaves the value as they are.
- If a pair is in decreasing order, the bubble sort swaps their values in the array.
- It requires several pass over the data.

BubbleSort




(a) Pass 1
(b) Pass 2
Initial array:
29 10 14 37 13
10 29 14 37 13
10 14 29 37 13
10 14 29 37 13
10 14 29 13 37
10 14 29 13 37
10 14 29 13 37
10 14 13 29 37
10 14 13 29 37

BubbleSort

```
// control number of passes
for ( int pass = 1; pass < b.length; pass++ )
        // control number of comparison
        for ( int i = 0; i < b.length - 1; i++ )
                if ( b[ i ] > b[ i + 1 ] )  {
                        int hold = b[i];
                        b[i] = b[i+1];
                        b[i+1] = hold;
                }
```

Exercise

- Create an array with a size of 20 and assign random number from 0 – 99 in the array
- Sort the elements in the array using bubble sort method

```
run:
24 25 82 61 40 17 100 26 21 47 41 67 63 75 74 38 25 43 89 42
Bubble Sort
17 21 24 25 25 26 38 40 41 42 43 47 61 63 67 74 75 82 89 100
```

Linear Search

- This method works well for **small arrays or for unsorted arrays**.
- The search technique search the array from the **first position to the last position** in a linear progression.
- It compares each element of the array with a search key.

```
for ( int cnt = 0; cnt < arrayName.length; cnt++ )
    if ( arrayName[ cnt ] == searchKey )
            return cnt;
return -1; // key not found
```

Exercise

- Create an array with a size of 20 and assign random number from 0 – 99 in the array
- Prompt the user to ‘enter number to search’
- Use linear search method to find the position

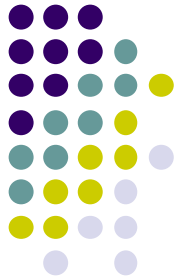
```
run:
8 42 8 64 21 51 54 79 87 12 97 9 57 67 76 2 73 23 47 27
Enter number to search : 27
Found in position 19
BUILD SUCCESSFUL (total time: 8 seconds)
```

Binary Search

- This method works well for **large and sorted arrays**.
- It eliminates half of the elements in the array being searched after each comparison.
- It locates the middle array element and compares it to the search key. If the search key not found, the binary search reduces the problem to **search for half** of the array.

```
int low = 0;   // low element subscript
int high = arrayName.length - 1;  // high element subscript
int middle;    // middle element subscript
```

Binary Search



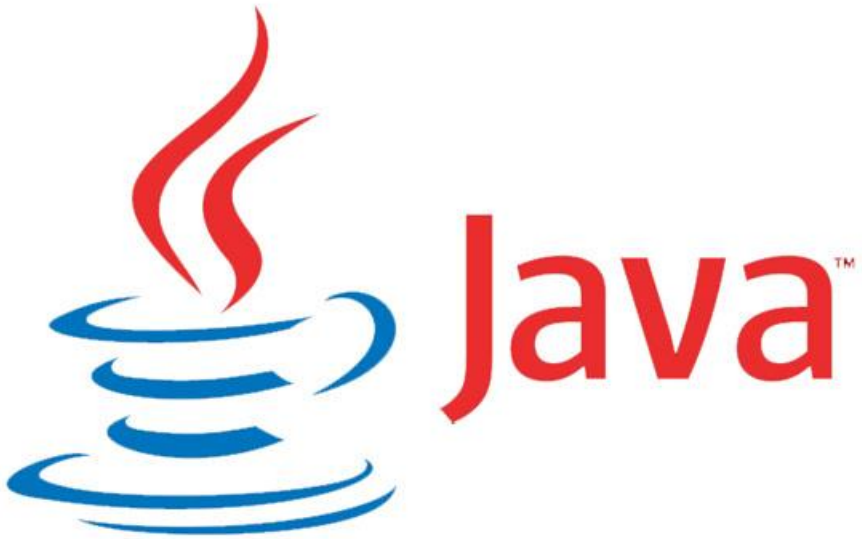
// loop until low subscript is greater than high subscript

```
while ( low <= high ) {
    middle = ( low + high ) / 2;
    if ( key == arrayName[ middle ] )
       return middle;
    else if ( key < arrayName[ middle ] )
       high = middle - 1;
    else
       low = middle + 1;
}
return -1;   // key not found
```

Exercise

- Create an array with a size of 20 and assign random number from 0 – 99 in the array
- Prompt the user to ‘enter number to search’
- Use binary search method to find the position
  - Note: the array need to be ascending order.

```
run:
2 70 60 46 82 38 29 36 54 88 45 98 4 51 42 94 74 36 18 11
Bubble Sort
2 4 11 18 29 36 36 38 42 45 46 51 54 60 70 74 82 88 94 98
Enter number to search : 29
Found in position 4
BUILD SUCCESSFUL (total time: 4 seconds)
```


Java™