Introduction to OOP with Java

Instructor: AbuKhleif, Mohammad Noor  
Sep 2017  
www.abukhleif.com

Lecture 06:

Arrays

Instructor: AbuKhleif, Mohammad Noor  
Sep 2017  
www.abukhleif.com
Instructor

- AbuKhleif, ‘Mohammad Noor’
- Computer Engineer (JU 2012-2017)
- Software Automation Engineer @ Atypon – John Wiley and Sons Company - Jordan Branch

- Reach me at:
  - www.abukhleif.com
  - moh.noor94@gmail.com
  - facebook.com/moh.noor94
  - twitter.com/moh_noor94

Course

- Java SE Basics
- Object Oriented Programming
- **Course Page:**
- Or, go to: [www.abukhleif.com](http://www.abukhleif.com) → Courses → Java 101 Course – Sep 2017
- Course Facebook Group:
  - [www.facebook.com/groups/AKF2017Java](http://www.facebook.com/groups/AKF2017Java)
Let’s Start!

Opening Problem

- Read one hundred marks, compute their average, and find out how many numbers are above the average.
Introducing Arrays

- Array is a data structure that represents a collection of the same types of data.
Declaring Arrays

• To use an array in a program, you must declare a variable to reference the array and specify the array’s elements type.
  • All elements in the array have the same data type.
• The syntax for declaring an array variable is:
  • `datatype[] arrayRefVar;`
  Example:
  ```java
double[] myList;
```
• `datatype arrayRefVar[];` // This style is allowed, but not preferred
  Example:
  ```java
double myList[];
```

Declaring Arrays, cont.

• Unlike declarations for primitive data type variables, the declaration of an array variable does not allocate any space in memory for the array.
  • It creates only a storage location for the reference to an array.
Creating Arrays

• An array is created using the `new` operator with the following syntax:

```
arrayRefVar = new elementType [arraySize];
```

• This statement does two things:
  • It creates an array using `new elementType [arraySize]`;  
  • It assigns the reference of the newly created array to the variable `arrayRefVar`.

• Example:

```
double [] myList; //array declaration
mylist = new double [10]; //array creation
```

Creating Arrays, cont.

• Array declaration and creation can be combined in one statement:

```
elementType [] arrayRefVar = new elementType [arraySize];
```

• Example:

```
double [] myList = new double [10];
```
The Length of an Array

- Once an array is created, its size is fixed. It cannot be changed. You can find its size using
  \texttt{arrayRefVar.length}

- For example:
  \texttt{myList.length} returns 10

Default Values

- When an array is created, its elements are assigned the default value of:
  - \texttt{0} for the numeric primitive data types,
  - \texttt{'\u0000'} for char types, and
  - \texttt{false} for boolean types.
Indexed Variables

• The array elements are accessed through the index.
• The array indices are 0-based, i.e., it starts from 0 to arrayRefVar.length-1.
  • In the example in Slide 8, myList holds ten double values and the indices are from 0 to 9.

• Each element in the array is represented using the following syntax, known as an indexed variable:
  arrayRefVar[index];

Using Indexed Variables

• After an array is created, an indexed variable can be used in the same way as a regular variable.
• For example, the following code adds the value in myList[0] and myList[1] to myList[2].

  myList[2] = myList[0] + myList[1];
Array Initializers

• Declaring, creating, initializing in one step:
  double[] myList = {1.9, 2.9, 3.4, 3.5};

• This shorthand syntax must be in one statement.

Declaring, creating, initializing Using the Shorthand Notation

double[] myList = {1.9, 2.9, 3.4, 3.5};

• This shorthand notation is equivalent to the following statements:
  double[] myList = new double[4];
  myList[0] = 1.9;
  myList[1] = 2.9;
  myList[2] = 3.4;
  myList[3] = 3.5;
Caution

• Using the shorthand notation, you have to declare, create, and initialize the array all in one statement.

• Splitting it would cause a syntax error.

• For example, the following is wrong:

```java
double[] myList;

myList = {1.9, 2.9, 3.4, 3.5};
```

Processing Arrays

• When processing array elements, you will often use a for loop.
  • All elements in an array are of the same type and they are evenly processed in the same fashion repeatedly using a loop.
  • Since the size of the array is known, it is natural to use a for loop.

• For example, to print an array, you have to print each element in the array using a loop like the following:

```java
for (int i = 0; i < myList.length; i++)
    System.out.print (myList[i] + “ “);
```
Example

Declare array variable values, create an array, and assign its reference to values

public class Test {
    public static void main(String[] args) {
        int[] values = new int[5];
        for (int i = 1; i < 5; i++) {
            values[i] = i + values[i-1];
        }
        values[0] = values[1] + values[4];
    }
}

Let’s Code

• Solve the ‘Opening Problem’ of this chapter.

Enough Talk Let’s Code
Enhanced for Loop
(for-each Loop)

- JDK 1.5 introduced a new for loop that enables you to traverse the complete array sequentially without using an index variable.
- For example, the following code displays all elements in the array myList:

```java
for (double value: myList)
    System.out.println(value);
```

- In general, the syntax is

```java
for (elementType value: arrayRefVar) {
    // Process the value
}
```

- You still have to use an index variable if you wish to traverse the array in a different order or change the elements in the array.
Let’s Code

• Resolve the ‘Opening Problem’ of this chapter using the enhanced for loop.
Copying Arrays

• The assignment operator does not copy the contents of an array into another, it instead merely copies the reference values.

Before the assignment
list2 = list1;

After the assignment
list2 = list1;

Contents of list1

Contents of list2

Copying Arrays, cont.

• To copy the contents of one array into another, you have to copy the array’s individual elements into the other array.
• Use a loop to copy every element from the source array to the corresponding element in the target array.
• Example:

```java
int[] sourceArray = {2, 3, 1, 5, 10};
int[] targetArray = new int[sourceArray.length];
for (int i=0; i < sourceArray.length; i++)
    targetArray[i] = sourceArray[i];
```
Copying Arrays, cont.
(The arraycopy Utility)

```java
arraycopy(sourceArray, src_pos, targetArray, tar_pos, length);
```

- Example:

```java
System.arraycopy(sourceArray, 0, targetArray, 0, sourceArray.length);
```

Passing Arrays to Methods
Passing Arrays to Methods

• Java uses *pass by value* to pass arguments to a method. There are important differences between passing a value of variables of primitive data types and passing arrays.

• For a parameter of a primitive type value, the actual value is passed.
  • Changing the value of the local parameter inside the method does not affect the value of the variable outside the method.

• For a parameter of an array type, the value of the parameter contains a reference to an array; this reference is passed to the method.
  • Any changes to the array that occur inside the method body will affect the original array that was passed as the argument.

Example

```java
public class Test {
    public static void main(String[] args) {
        int x = 1; // x represents an int value
        int[] y = new int[10]; // y represents an array of int values

        m(x, y); // Invoke m with arguments x and y

        System.out.println("x is " + x);
        System.out.println("y[0] is " + y[0]);
    }

    public static void m(int number, int[] numbers) {
        number = 1001; // Assign a new value to number
        numbers[0] = 5555; // Assign a new value to numbers[0]
    }
}
```
• When invoking \( m(x, y) \), the values of \( x \) and \( y \) are passed to \texttt{number} and \texttt{numbers}.

• Since \( y \) contains the reference value to the array, \texttt{numbers} now contains the same reference value to the same array.

Self Study

Passing Arrays to Methods, cont.

• Please study the following example:

\[ \text{http://www.cs.armstrong.edu/liang/intro9e/html/TestPassArray.html} \]

• Objective: Demonstrate differences of passing primitive data type variables and array variables.
Anonymous Array

- The statement
  ```java
  printArray(new int[]{3, 1, 2, 6, 4, 2});
  ```
  creates an array using the following syntax:
  ```java
  new dataType[]{literal0, literal1, ..., literalk};
  ```
- There is no explicit reference variable for the array.
- Such array is called an *anonymous array*.

Returning an Array from a Method
Returning an Array from a Method

• When a method returns an array, the reference of the array is returned.

• Example:

```java
public static int[] copy(int[] list) {
    int[] result = new int[list.length];
    for (int i = 0; i < list.length; i++)
        result[i] = list[i];
    return result;
}
```

Example of this method invocation:
```java
int[] list1 = {1, 2, 3, 4, 5};
int[] list2 = copy(list1);
```
Searching Arrays

• Searching is the process of looking for a specific element in an array; for example, discovering whether a certain score is included in a list of scores.
• Searching is a common task in computer programming.
• There are many algorithms and data structures devoted to searching.
• In this section, two commonly used approaches are discussed, *linear search* and *binary search*.

Linear Search

• The linear search approach compares the key element, *key*, *sequentially* with each element in the array *list*.
• The method continues to do so until the key matches an element in the list or the list is exhausted without a match being found.
• If a match is made, the linear search returns the index of the element in the array that matches the key.
• If no match is found, the search returns `-1`. 
Linear Search, cont.

- Try different examples by animation here:
  [http://www.cs.armstrong.edu/liang/animation/web/LinearSearch.html](http://www.cs.armstrong.edu/liang/animation/web/LinearSearch.html)
Linear Search, From Idea to Solution

```java
/** The method for finding a key in the list */
public static int linearSearch(int[] list, int key) {
    for (int i = 0; i < list.length; i++)
        if (key == list[i])
            return i;
    return -1;
}
```

Trace the method:
```
int[] list = {1, 4, 4, 2, 5, -3, 6, 2};
int i = linearSearch(list, 4); // returns 1
int j = linearSearch(list, -4); // returns -1
int k = linearSearch(list, -3); // returns 5
```

Binary Search

- For binary search to work, the elements in the array must already be ordered.
- Without loss of generality, assume that the array is in ascending order.
  - e.g., 2 4 7 10 11 45 50 59 60 66 69 70 79
- The binary search first compares the key with the element in the middle of the array.
Binary Search, cont.

- Consider the following three cases:
  - If the key is less than the middle element, you only need to search the key in the first half of the array.
  - If the key is equal to the middle element, the search ends with a match.
  - If the key is greater than the middle element, you only need to search the key in the second half of the array.

<table>
<thead>
<tr>
<th>Key</th>
<th>List</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1 2 3 4 6 7 8 9</td>
</tr>
<tr>
<td>8</td>
<td>1 2 3 4 6 7 8 9</td>
</tr>
<tr>
<td>8</td>
<td>1 2 3 4 6 7 8 9</td>
</tr>
</tbody>
</table>
Binary Search, cont.

- Try this example with animation:
  [Link](http://www.cs.armstrong.edu/liang-animation/web/BinarySearch.html)
Example 2

Binary Search, cont.

- The `binarySearch` method returns the index of the element in the list that matches the search key if it is contained in the list. Otherwise, it returns -1.
**Use binary search to find the key in the list */

```java
public static int binarySearch(int[] list, int key) {
    int low = 0;
    int high = list.length - 1;

    while (high >= low) {
        int mid = (low + high) / 2;
        if (key < list[mid])
            high = mid - 1;
        else if (key == list[mid])
            return mid;
        else
            low = mid + 1;
    }

    return -1 - low;
}
```

The `Arrays.binarySearch` Method

- Since binary search is frequently used in programming, Java provides several overloaded `binarySearch` methods for searching a key in an array of `int`, `double`, `char`, `short`, `long`, and `float` in the `java.util.Arrays` class.
- For example, the following code searches the keys in an array of numbers and an array of characters.
  ```java
  int[] list = {2, 4, 7, 10, 11, 45, 50, 59, 60, 66, 69, 70, 79};
  System.out.println("Index is " +
                     java.util.Arrays.binarySearch(list, 11));

  char[] chars = {'a', 'c', 'g', 'x', 'y', 'z'};
  System.out.println("Index is " +
                     java.util.Arrays.binarySearch(chars, 't'));
  ```
- For the `binarySearch` method to work, the array must be pre-sorted in increasing order.
Let’s Code

- Revisit your ‘Opening Problem’ solution, edit it by letting the user enter the mark and the name for each student.
- Then let the user enter any mark and return the name of first student with such a mark.
- If the mark does not found, print appropriate message clarifying that.

Sorting Arrays
Sorting Arrays

• Sorting, like searching, is also a common task in computer programming.
• Many different algorithms have been developed for sorting.
• This section introduces two simple, intuitive sorting algorithms: **selection sort** and **insertion sort**.

### Selection Sort

1. Select 1 (the smallest) and swap it with 2 (the first) in the list

\[ 2 \quad 9 \quad 5 \quad 4 \quad 8 \quad 1 \quad 6 \]

2. **swap**

Select 1 (the smallest) and swap it with 2 (the first) in the list

\[ 1 \quad 9 \quad 5 \quad 4 \quad 8 \quad 2 \quad 6 \]

3. **swap**

The number 1 is now in the correct position and thus no longer needs to be considered.

\[ 1 \quad 2 \quad 5 \quad 4 \quad 8 \quad 9 \quad 6 \]

4. **swap**

Select 2 (the smallest) and swap it with 9 (the first) in the remaining list

\[ 1 \quad 2 \quad 4 \quad 5 \quad 8 \quad 9 \quad 6 \]

5. The number 2 is now in the correct position and thus no longer needs to be considered.

\[ 1 \quad 2 \quad 4 \quad 5 \quad 6 \quad 9 \quad 8 \]

6. **swap**

Select 4 (the smallest) and swap it with 5 (the first) in the remaining list

\[ 1 \quad 2 \quad 4 \quad 5 \quad 8 \quad 9 \quad 6 \]

7. The number 4 is now in the correct position and thus no longer needs to be considered.

\[ 1 \quad 2 \quad 4 \quad 5 \quad 6 \quad 8 \quad 9 \]

8. **swap**

Select 6 (the smallest) and swap it with 8 (the first) in the remaining list

\[ 1 \quad 2 \quad 4 \quad 5 \quad 6 \quad 8 \quad 9 \]

9. The number 6 is now in the correct position and thus no longer needs to be considered.

\[ 1 \quad 2 \quad 4 \quad 5 \quad 6 \quad 8 \quad 9 \]

10. **swap**

The number 8 is now in the correct position and thus no longer needs to be considered.

\[ 1 \quad 2 \quad 4 \quad 5 \quad 6 \quad 8 \quad 9 \]

11. Since there is only one element remaining in the list, sort is completed

The number 5 is now in the correct position and thus no longer needs to be considered.
Selection Sort, cont.

• Try this example with animation:
  http://www.cs.armstrong.edu/liang/animation/web/SelectionSort.html

Selection Sort
From Idea to Solution

```java
for (int i = 0; i < list.length; i++)
{
    // 1. select the smallest element in list[i..listSize-1];
    // 2. swap the smallest with list[i], if necessary;
    // list[i] is in its correct position.
    // The next iteration apply on list[i..listSize-1]
}
```
/** The method for sorting the numbers */
public static void selectionSort(double[] list) {
    for (int i = 0; i < list.length; i++) {
        // Find the minimum in the list[i..list.length-1]
        double currentMin = list[i];
        int currentMinIndex = i;
        for (int j = i + 1; j < list.length; j++) {
            if (currentMin > list[j]) {
                currentMin = list[j];
                currentMinIndex = j;
            }
        }
        // Swap list[i] with list[currentMinIndex] if necessary;
        if (currentMinIndex != i) {
            list[currentMinIndex] = list[i];
            list[i] = currentMin;
        }
    }
}

Selection Sort Code

Insertion Sort

- The insertion sort algorithm sorts a list of values by repeatedly inserting an unsorted element into a sorted sublist until the whole list is sorted.
Insertion Sort – Example

```java
int[] myList = {2, 9, 5, 4, 8, 1, 6}; // Unsorted
```

Step 1: Initially, the sorted sublist contains the first element in the list. Insert 9 into the sublist.

Step 2: The sorted sublist is {2, 9}. Insert 5 into the sublist.

Step 3: The sorted sublist is {2, 5, 9}. Insert 4 into the sublist.

Step 4: The sorted sublist is {2, 4, 5, 9}. Insert 8 into the sublist.

Step 5: The sorted sublist is {2, 4, 5, 8, 9}. Insert 1 into the sublist.

Step 6: The sorted sublist is {1, 2, 4, 5, 8, 9}. Insert 6 into the sublist.

Step 7: The entire list is now sorted.
Insertion Sort, cont.

- Try this example with animation:

---

Insertation Sort

How to Insert?

- The insertion sort algorithm sorts a list of values by repeatedly inserting an unsorted element into a sorted sublist until the whole list is sorted.

  ```
  [0] [1] [2] [3] [4] [5] [6]
  list 2 5 9 4
  ``

  **Step 1:** Save 4 to a temporary variable `currentEle`

  ```
  [0] [1] [2] [3] [4] [5] [6]
  list 2 5 9
  ``

  **Step 2:** Move list[2] to list[3]

  ```
  [0] [1] [2] [3] [4] [5] [6]
  list 2 5 9
  ``

  **Step 3:** Move list[1] to list[2]

  ```
  [0] [1] [2] [3] [4] [5] [6]
  list 2 4 5 9
  ``

  **Step 4:** Assign `currentEle` to list[1]

  ```
  [0] [1] [2] [3] [4] [5] [6]
  list 2 4 5 9
  ```
Insertion Sort
From Idea to Solution

for (int i = 1; i < list.length; i++) {
    // insert list[i] into a sorted sublist list[0..i-1] so that
    // list[0..i] is sorted
}

Self Study
Insertion Sort
From Idea to Solution

• Insertion sort solution can be found here:
  http://www.geeksforgeeks.org/insertion-sort/
Insertion Sort
The Arrays.sort Method

• Since sorting is frequently used in programming, Java provides several overloaded sort methods for sorting an array of int, double, char, short, long, and float in the java.util.Arrays class.
• For example, the following code sorts an array of numbers and an array of characters.

```java
double[] numbers = {6.0, 4.4, 1.9, 2.9, 3.4, 3.5};
java.util.Arrays.sort(numbers);

char[] chars = {'a', 'A', '4', 'F', 'D', 'P'};
java.util.Arrays.sort(chars);
```

Let’s Code

• Revisit your ‘Opening Problem’ solution, sort the marks of all students and print the results.
Introduction to Multi-Dimensional Arrays

Two-Dimensional Arrays

- Two dimensional arrays are used to represent data in a matrix or a table.
- The syntax for declaring and creating two dimensional arrays is:
  ```java
elementType [][] arrayRefVar;
arrayRefVar = new elementType [numRows][numCols];
```
- An element in a two-dimensional array is accessed through a row and column index: `arrayRefVar [rowIndex][colIndex];`
Two-Dimensional Arrays, cont.

- A two-dimensional array is actually an array in which each element is a one-dimensional array.

```
x[0][0]  x[0][1]  x[0][2]  x[0][3]  x[0].length is 4
x[1][0]  x[1][1]  x[1][2]  x[1][3]  x[1].length is 4
```

Two-Dimensional Arrays
Examples

```
[0] [1] [2] [3] [4]
[0] 0 0 0 0 0
[1] 0 0 0 0 0
[2] 0 0 0 0 0
[3] 0 0 0 0 0
[4] 0 0 0 0 0
matrix = new int[5][5];
```

```
[0] [1] [2] [3] [4]
[0] 0 0 0 0 0
[1] 0 0 0 0 0
[2] 0 0 0 0 0
[3] 0 0 0 0 0
[4] 0 0 0 0 0
matrix[2][1] = 7;
```

```
[0] [1] [2] [3]
[0] 1 2 3
[1] 4 5 6
[2] 7 8 9
[3] 10 11 12
```

```
int[][] array = {
{1, 2, 3},
{4, 5, 6},
{7, 8, 9},
{10, 11, 12}
};
```
Ragged Arrays

- Each row in a two-dimensional array is itself an array.
- So, the rows can have different lengths.
- Such an array is known as a **ragged array**.
- For example,

```java
int[][] triangleArray = {
    {1, 2, 3, 4, 5},
    {2, 3, 4, 5},
    {3, 4, 5},
    {4, 5},
    {5}
};
```

Processing Two-Dimensional Arrays

- When processing two-dimensional array elements, you will often use a nested for loop.
  - All elements in an array are of the same type and they are evenly processed in the same fashion repeatedly using a loop.
  - Since the size of the array is known, it is natural to use a for loop.

- For example, to print a two-dimensional array, you have to print each element in the array using a loop like the following:

```java
for (int i = 0; i < myList.length; i++)
    for (int j = 0; j < myList[i].length; j++)
        System.out.print (myList[i][j] + " ");
```
Processing Two-Dimensional Arrays Example

• Study and execute the following example in your machine: http://www.cs.armstrong.edu/liang/intro9e/html/PassTwoDimensionalArray.html

Multidimensional Arrays

• Occasionally, you will need to represent n-dimensional data structures. In Java, you can create n-dimensional arrays for any integer n.

• The way to declare two-dimensional array variables and create two-dimensional arrays can be generalized to declare n-dimensional array variables and create n-dimensional arrays for n >= 3.
Multidimensional Arrays - Example

```java
double[][][] scores = {
    {{7.5, 20.5}, {9.0, 22.5}, {15, 33.5}, {13, 21.5}, {15, 2.5}},
    {{4.5, 21.5}, {9.0, 22.5}, {15, 34.5}, {12, 20.5}, {14, 9.5}},
    {{6.5, 30.5}, {9.4, 10.5}, {11, 33.5}, {11, 23.5}, {10, 2.5}},
    {{6.5, 23.5}, {9.4, 32.5}, {13, 34.5}, {11, 20.5}, {16, 7.5}},
    {{8.5, 26.5}, {9.4, 52.5}, {13, 36.5}, {13, 24.5}, {16, 2.5}},
    {{9.5, 20.5}, {9.4, 42.5}, {13, 31.5}, {12, 20.5}, {16, 6.5}}};
```

Which student

Which exam

Multiple-choice or essay

scores[i][j][k]

Tasks

All tasks should be well-documented, well-designed, and well-styled.
Task 01

*(Reverse the numbers entered)*

- Write a program that reads ten integers and displays them in the reverse of the order in which they were read.

Task 02

*(Average an array)*

- Write two overloaded methods that return the average of an array with the following headers:
  
  ```java
  public static int average(int[] array)
  public static double average(double[] array)
  ```

- Write a test program that prompts the user to enter ten double values, invokes this method, and displays the average value.
Task 03

(Eliminate duplicates)

• Write a method that returns a new array by eliminating the duplicate values in the array using the following method header:

  ```java
  public static int[] eliminateDuplicates(int[] list)
  ```

• Write a test program that reads in ten integers, invokes the method, and displays the result. Here is the sample run of the program:

  Enter ten numbers: 1 2 3 2 1 6 3 4 5 2
  The distinct numbers are: 1 2 3 6 4 5

Task 04 (Optional)

(Sorted?) **

• Write the following method that returns true if the list is already sorted in increasing order.

  ```java
  public static boolean isSorted(int[] list)
  ```

• Write a test program that prompts the user to enter a list and displays whether the list is sorted or not. Here is a sample run.

• Note that the first number in the input indicates the number of the elements in the list. This number is not part of the list.

• Next slide shows sample runs.
Task 04 (Optional) – Sample Runs

Enter list: 8 10 1 5 16 61 9 11 1
The list is not sorted

Enter list: 10 1 1 3 4 4 5 7 9 11 21
The list is already sorted

Task 05 (Optional)

(Locate the largest element) *

• Write the following method that returns the location of the largest element in a two-dimensional array.

public static int[] locateLargest(double[][] a)

• The return value is a one-dimensional array that contains two elements.
  • These two elements indicate the row and column indices of the largest element in the two-dimensional array.
• Write a test program that prompts the user to enter a two-dimensional array and displays the location of the largest element in the array.
• Next slide shows a sample run.
Task 05 (Optional) – Sample Run

Enter the number of rows and columns of the array: 3 4
Enter the array:
10 2 35 23.5
3.5 45 3 4.5
9.6 5.5 44 35
The location of the largest element is at (1, 2)

Tasks Submission

• Submit a zipped file contains all the .java files, 1 file for each task.
• Name your zipped file as follow [Lect6_YourName.zip].
• Upload your zipped file to the Facebook group.
• Submission due: Thursday, Sep 21 - 10:00 PM
• Late submission will not be reviewed by the instructor.
• Public solutions upload goal is to share knowledge, you can see other’s solutions, but, please, don’t cheat yourself!
• Don’t forget, all tasks should be well-documented, well-designed, and well-styled.
Test Yourself

• Answer all question:
  

And


References:

- Liang, Introduction to Java Programming 10/e
- Eng. Asma Abdel Karim Computer Engineering Department, JU Slides.

Instructor: AbuKhleif, Mohammad Noor
Sep 2017
www.abukhleif.com