Chapter 6 Arrays

Motivations
- Suppose we want to write a program that computes the average midterm score of all the students and find out how many scored above the average.
- How many variables do we need for storing the scores? (30 students!)
- Solution: Arrays

Arrays
- Array is a data structure that represents a collection of the same types of data.
- How to use arrays?
  - Declaring array variables
  - Creating arrays
  - Initializing arrays
  - Accessing and using array elements
  - Traversing arrays using for loops

Declaring Array Variables
- `datatype[] arrayRefVar;` Preferred!
  - Example:
    ```java
double[] myList;
    
    datatype arrayRefVar[];
    
    Example:
    ```
    ```java
double myList[];
    ```

Creating Arrays
- `arrayRefVar = new datatype[arraySize];`
  - Example:
    ```java
    myList = new double[10];
    ```

Declaring and Creating an Array in One Step
- `Datatype[] arrayRefVar;` `arrayRefVar = new datatype[arraySize];`
  - Example:
    ```java
double[] myList = new double[10];
    ```
Default Values

- When an array is created, its elements are assigned the default value:
  - 0 for the numeric primitive data types
  - (char)0 for char types
  - false for boolean types

Assigning Values to Array Elements

arrayRefVar[index] = expression;

Example:

```java
myList[0] = 5.6;
myList[1] = 4.5;
...
myList[9] = 11123;
```

Inside Arrays

- The array elements are accessed through the index.
- Each element in the array is represented using:
  ```java
  arrayRefVar[index]
  ```
- The array indices start from 0 to array.length-1

Example:

```java
double[] myList = new double[10];
myList[0] = 5.6;
myList[1] = 4.5;
myList[10] = 3.33; // ArrayIndexOutOfBoundsException
```

Array Initializers - Declaring, creating, initializing an array using one statement

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

This shorthand notation is equivalent to the following statements:

```java
double[] myList = new double[4];
myList[0] = 1.9;
myList[1] = 2.9;
myList[2] = 3.4;
myList[3] = 3.5;
```

Note: this shorthand syntax must be in one statement

The Length of an Array

Once an array is created, its size is fixed. It cannot be changed. You can find its size using

```java
arrayRefVar.length
```

For example:

```java
int[] numbers = new int[10];
int len = numbers.length; // 10
```

Accessing Array Elements

- The array elements are accessed through the index.
- Each element in the array is represented using:
  ```java
  arrayRefVar[index]
  ```
- The array indices start from 0 to array.length-1

Example:

```java
double[] myList = new double[10];
myList[0] = 5.6;
myList[1] = 4.5;
myList[10] = 3.33; // ArrayIndexOutOfBoundsException
```
Using Array Elements

- Use elements in arrays as usual variables!!!

```java
int[] tmp = new int[10];

System.out.println("tmp[2]: " + tmp[2]);
System.out.println("tmp[9]: " + tmp[9]);
```

Example

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

```java
double[] myList = new double[5];
myList[0] = 3.0;
myList[1] = 4.5;
myList[2] = 2.0;
System.out.println(myList[2]);
System.out.println(myList[4]);
```

```
double[] myList = new double[5];
myList[0] = 3.0;
myList[1] = 4.5;
myList[2] = 2.0;
System.out.println(myList[2]);
System.out.println(myList[4]);
```

Example

```java
double[] myList = new double[5];
myList[0] = 3.0;
myList[1] = 4.5;
myList[2] = 2.0;
System.out.println(myList[2]);
System.out.println(myList[4]);
```

```
double[] myList = new double[5];
myList[0] = 3.0;
myList[1] = 4.5;
myList[2] = 2.0;
System.out.println(myList[2]);
System.out.println(myList[4]);
```
Using for loops to process arrays

- **Using for loops to traverse or process array elements**
  - Elements are of the same type and processed repeatedly
  - The size of the array is known
  ```java
  for (int i=0; i<myList.length; i++) {
      // process ith element myList[i]
      System.out.println(myList[i]);
  }
  ```

Enhanced for Loop (for-each loop)

- **Traditional for loop**
  ```java
  int[] array = new int[5];
  for (int i = 0; i < array.length; i++) {
      double value = array[i];
      System.out.println(value);
  }
  ```

  - **Generalized for loop**
  ```java
  for (double value: array) // for each value in array
      System.out.println(value);
  ```

Common Array Processing

- Initializing array elements
- Printing arrays
- Summing all elements
- Finding largest elements
- Finding smallest index of the largest element
- Shifting elements
- Flipping elements
- Counting occurrences
Initialize array elements using for loop

Initialize an array with random values

```java
for (int i=0; i<myList.length; i++) {
    myList[i] = Math.random() * 100;
}
```

Another example of initializing arrays

```java
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];
    }
}
```

Trace Program with Arrays

1. i=1, value[1] is 1
2. i=2, value[2] is 3
3. i=3, values[3] = 6 (3 + 3)
Simple Array Algorithms:
Finding the Maximum or Minimum
- Problem: finding the maximum/minimum
  - Algorithm:
    - Initialize the current maximum/minimum with the starting element
    - For each of the remaining elements
      - Update the current maximum/minimum with the element if it is larger or smaller
  ```java
  int max = myList[0];
  for (int i=1; i<myList.length; i++) {
    if (max < myList[i])
      max = myList[i];
  }
  ```

Simple Array Algorithms: Counting Matches
- Problem: counting number of a particular item in an array
  - Algorithm:
    - Initialize a counter
    - For each of the elements in the array
      - Increase the counter when there is a match
  ```java
  int count = 0;
  for (int i=0; i<myList.length; i++) {
    if (myList[i] == 'A')
      count ++;
  }
  ```

Simple Array Algorithms: Finding the smallest index of largest item
- Algorithm:
  - Initialize current maximum with the starting element and the index of current maximum with 0
  - For each of the remaining elements
    - Update current maximum and index if the element is larger
  ```java
  int max = myList[0];
  int indexOfMax = 0;
  for (int i=1; i<myList.length; i++) {
    if (max < myList[i]) {
      max = myList[i];
      indexOfMax = i;
    }
  }
  ```

Simple Array Algorithms: computing the sum
- Problem: computing the sum of all elements in an array
  - Algorithm:
    - Initialize the current sum with 0
    - For each of the elements
      - Add the element to the sum
  ```java
  int sum = 0;
  for (int i=0; i<myList.length; i++) {
    sum += myList[i];
  }
  ```
  - How to compute the average?
Analyzing Array Elements

- Write a program that:
  - Receives 6 numbers from the user and place them in an array
  - Finds the largest number and count of the occurrences of this number in the array

TestArray.java

Problem: Assigning Grades

- Objective: read student scores (int), get the best score, and then assign grades based on the following scheme:
  - Grade is A if score is $\geq$ best–10;
  - Grade is B if score is $\geq$ best–20;
  - Grade is C if score is $\geq$ best–30;
  - Grade is D if score is $\geq$ best–40;
  - Grade is F otherwise.

AssignGrades.java

Review Question

Analyze the following code:
```
int[] x = new int[5];
int i;
for (i = 0; i < x.length; i++)
x[i] = i;
System.out.println(x[i]);
```
A. It displays 0 1 2 3 4.
B. It displays 4.
C. It has a runtime error because the last statement causes ArrayIndexOutOfBoundsException.
D. It has a compile error because i is not defined in the last statement in the main method.

Review: Array basics

- Array basics
  - Declare an array
  - Create an array
  - Initialize an array
  - Traverse an array using for loops
  - Array processing algorithms
- Practice problems on textbook
  - 6.3, 6.4, 6.5, 6.9

Outline

- Copying arrays
- Passing arrays to methods
- Returning an array from methods
- The Array class
- Two dimensional arrays
### Inside Arrays

Double array `myList`:

<table>
<thead>
<tr>
<th>Index</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>8.5</td>
</tr>
<tr>
<td>2</td>
<td>3.5</td>
</tr>
<tr>
<td>3</td>
<td>12.2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>34.33</td>
</tr>
<tr>
<td>6</td>
<td>11123</td>
</tr>
<tr>
<td>7</td>
<td>99.993</td>
</tr>
<tr>
<td>8</td>
<td>45.45</td>
</tr>
<tr>
<td>9</td>
<td>0.0</td>
</tr>
</tbody>
</table>

### The arraycopy method

System.arraycopy(sourceArray, src_pos, targetArray, tar_pos, length);

Example:

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

### Review question

In the following code, what is the printout for list2?

```java
class Test {
    public static void main(String[] args) {
        int[] list1 = {1, 2, 3};
        int[] list2 = {1, 2, 3};
        list2 = list1;
        list1[0] = 0; list1[1] = 1; list2[2] = 2;
        for (int i = 0; i < list2.length; i++)
            System.out.print(list2[i] + "");
    }
}
```

A. 1 2 3  
B. 1 1 1  
C. 0 1 2  
D. 0 1 3

### Passing Arrays to Methods

```java
public static void printArray(int[] array) {
    for (int i = 0; i < array.length; i++)
        System.out.print(array[i] + "");
}
```

```java
public static void main(String[] array) {
    int[] list = {3, 1, 2, 6, 4, 2};
    printArray(list);
}
```
Pass By Value

- **What is pass by value?**

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

Simple 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]
    }
}
```

Returning an Array from a Method

```java
public static int[] reverse(int[] list) {
    int[] result = new int[list.length];
    for (int i = 0; i < list.length; i++) {
        result[result.length - i - 1] = list[i];
    }
    return result;
}
```

Call Stack

- When invoking m(x, y), the values of x and y are passed to number and numbers
- y contains the reference value to the array!

Returning an Array from a Method

```java
public static int[] reverse(int[] list) {
    int[] result = new int[list.length];
    for (int i = 0; i < list.length; i++) {
        result[result.length - i - 1] = list[i];
    }
    return result;
}
```

Trace the reverse Method

```java
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```
```java
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);

public static int[] reverse(int[] list) {
    int[] result = new int[list.length];
    for (int i = 0; i < list.length; i++) {
        result[result.length - i - 1] = list[i];
    }
    return result;
}
```

**Trace the reverse Method, cont.**

```
public static int[] reverse(int[] list) {
    int[] result = new int[list.length];
    for (int i = 0; i < list.length; i++) {
        result[result.length - i - 1] = list[i];
    }
    return result;
}
```

```java
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {
    int[] result = new int[list.length];
    for (int i = 0; i < list.length; i++) {
        result[result.length - i - 1] = list[i];
    }
    return result;
}
```

- **i = 0**
- **Assign list[0] to result[5]**

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

```java
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {
    int[] result = new int[list.length];
    for (int i = 0; i < list.length; i++) {
        result[result.length - i - 1] = list[i];
    }
    return result;
}
```

- **i = 1**
- **Assign list[1] to result[4]**

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

```java
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {
    int[] result = new int[list.length];
    for (int i = 0; i < list.length; i++) {
        result[result.length - i - 1] = list[i];
    }
    return result;
}
```

- **i = 2**
- **Assign list[2] to result[3]**

```
list [1 2 3 4 5 6]
result [0 0 3 2 1]
```
```java
public static int[] reverse(int[] list) {
    int[] result = new int[list.length];
    for (int i = 0; i < list.length; i++) {
        result[result.length - i - 1] = list[i];
    }
    return result;
}
```

```java
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

Trace the reverse Method, cont.

i = 3
Assign list[3] to result[2]

```
list 1 2 3 4 5 6
result 0 0 0 0 0 0
```

i = 4
Assign list[4] to result[1]

```
list 1 2 3 4 5 6
result 0 0 4 3 2 1
```

After this, i becomes 6

```
list 1 2 3 4 5 6
result 0 5 4 3 2 1
```

i(=6) < 6 is false. So exit the loop.

```
list 1 2 3 4 5 6
result 0 5 4 3 2 1
```

Return result
Review question

What does the following program print?

```java
public class Test {
    public static void main(String[] args) {
        int[] oldList = {1, 2, 3, 4, 5};
        int[] newList = reverse(oldList);
        for (int i = 0; i < oldList.length; i++)
            System.out.print(oldList[i] + " ");
        for (int i = 0; i < newList.length; i++)
            System.out.print(newList[i] + " ");
    }
    public static int[] reverse(int[] list) {
        int[] newList = new int[list.length];
        for (int i = 0; i < list.length; i++)
            newList[i] = list[list.length - 1 - i];
        list = newList;
        return newList;
    }
}
```

The Arrays.sort Method

- **How do we sort an array?**
- Java provides several overloaded sort methods in the `java.util.Arrays` class for sorting an array of int, float, double, char … in ascending order.
  - `public static void sort(int[] a)`
  - `public static void sort(double[] a)`
  - `public static void sort(char[] a)`

**Example**

```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);
```

Accessing items in 2D array

```java
int[][] matrix = new int[10][10];
matrix[0][0] = 3;
for (int i = 0; i < matrix.length; i++)
    for (int j = 0; j < matrix[i].length; j++)
        matrix[i][j] = (int)(Math.random() * 1000) + 1000;
```

Declarations, Create, and Initializing Using Shorthand Notations

- You can also use an array initializer to declare, create and initialize a two-dimensional array.

```java
int[][] array = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9},
    {10, 11, 12}
};
```
Lengths of Two-dimensional Arrays

- int[][] x = new int[3][4];

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

Ragged Arrays, cont.

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

```
1 2 3 4 5
2 3 4 5
3 4 5
4 5
5
```

Two-dimensional Array Illustration

- matrix = new int[5][5];

```
0   1    2    3    4
0
1
2
3
4
```

Problem: Grading Multiple-Choice Test

- Objective: write a program that grades multiple-choice test.

```
0 1 2 3 4 5 6 7

matrix = new int[][] {{1, 2, 3, 4, 5},
                      {2, 3, 4, 5},
                      {3, 4, 5},
                      {4, 5},
                      {5}};
```

Problem: Finding Two Points Nearest to Each Other

```
-1 3
-1 -1
1 1
2 0.5
3 0.5
4 -1
5 3
6 4.5
7 4 -0.5
```
Review question

Assume `double[][][] x = new double[4][5][6]`, what are `x.length`, `x[2].length`, and `x[0][0].length`?

A. 4, 5, and 6
B. 6, 5, and 4
C. 5, 5, and 5
D. 4, 5, and 4

Practice problems

- Copying arrays
  - 6.11
- Passing arrays to and returning arrays from methods
  - 6.13
- Two-dimensional arrays
  - 6.27, 6.28

Lab7: Array Processing

- Reading a list of numbers
- Computing and printing
  - Max
  - Min
  - Mean
  - Median of the array
  - Number of elements smaller than and greater than a given value

Hw4: Image (2-D Array) Processing

- An image is represented as a 2-dimensional array `int[][]`
- Each element is an intensity value which can be converted into three components: RGB

Provided Utilities in Image class

- `public static int[][] loadImage(String imageFile)`
  - converts an image file to a 2-d array representing pixels of the image
- `public static void saveImage(int[][] array, String imageFile)`
  - saves a 2-d array representing pixels of the image to a file
- `public static int colorToRGB(int pixel)`
  - converts each pixel value represented as a single integer to its RGB representation, an array of three integers. Intensity of each color is represented as number in range 0-255, where 0 is the lowest intensity and 255 is the highest.
- `public static int colorFromRGB(int[] rgb)`
  - Converts an array of three integers representing intensity of red, green and blue to a single integer.

Your task

- Process the 2-dimensional array for cool image effects!
Review Question

```java
public class Test {
    public static void main(String[] args) {
        int[] x = {1, 2, 3, 4};
        int[] y = x;
        x = new int[2];
        for (int i = 0; i < x.length; i++)
            System.out.print(x[i] + " ");
    }
}
```

A. The program displays 1 2 3 4
B. The program displays 0 0
C. The program displays 0 0 3 4
D. The program displays 0 0 0 0

Linear Search

- Compares the key element 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.

Outline

- What we have covered so far
  - Array basics
  - Copying arrays
  - Passing arrays to methods
  - Returning an array from methods
  - The Arrays class and Arrays.sort method
  - Two dimensional arrays
- Today
  - Searching arrays
  - Sorting arrays

Linear Search Animation

<table>
<thead>
<tr>
<th>Key</th>
<th>List</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>6 4 1 9 7 3 2 8</td>
</tr>
<tr>
<td>3</td>
<td>6 4 1 9 7 3 2 8</td>
</tr>
<tr>
<td>4</td>
<td>6 4 1 9 7 3 2 8</td>
</tr>
<tr>
<td>6</td>
<td>6 4 1 9 7 3 2 8</td>
</tr>
<tr>
<td>6</td>
<td>6 4 1 9 7 3 2 8</td>
</tr>
<tr>
<td>3</td>
<td>6 4 1 9 7 3 2 8</td>
</tr>
</tbody>
</table>

Searching Arrays

- You are given an array and value x
- Return the position of x in array (index)
- If x does not appear in the array, return -1

```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;
}
```

```java
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
```

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;
}
```
Binary Search

- Linear search works well, but has one disadvantage.
- Can we do better if the elements in the array are ordered (usually in ascending order)?

Binary Search, cont.

<table>
<thead>
<tr>
<th>key is 11</th>
<th>low</th>
<th>mid</th>
<th>high</th>
</tr>
</thead>
<tbody>
<tr>
<td>key &lt; 50</td>
<td>2</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>key &gt; 7</td>
<td>2</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>key == 11</td>
<td>11</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Implement Binary Search

- Implement a binarySearch method that returns the index of the element in the list that matches the search key if it is contained in the list
- Otherwise, it returns (-insertion point – 1)
  - The insertion point is the point at which the key would be inserted into the list.
From Idea to Solution

```java
/** Use binary search to find the key in the list */
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;
}
```

From Idea to Solution

Let's implement binary search!

```text
BinarySearch.java
```

The Arrays.binarySearch Method

Binary search is implemented in Java and ready to use!

```java
int[] list = {2, 4, 7, 10, 11, 45, 50, 60, 66, 69, 70, 79};
System.out.println("Index is "+
    java.util.Arrays.binarySearch(list, 13));
char[] chars = {'a', 'c', 'g', 'x', 'y', 'z'};
System.out.println("Index is "+
    java.util.Arrays.binarySearch(chars, 't'));
//Return is -4 (insertion point is 3, so return is -3-1)
```

For the binarySearch method to work, the array must be pre-sorted in increasing order.

Selection Sort

Problem: sort arrays so that items are in ascending (descending) order

How do we do this?

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

Selection Sort

```java
int[] mylist = {2, 9, 5, 4, 8, 1, 6}; // Unsorted
```
Wrap it in a Method

```java
/** The method for sorting the numbers */
public static void selectionSort(double[] list) {
    for (int i = list.length - 1; i >= 1; i--) {
        // Find the maximum in the list[0..i]
        double currentMax = list[0];
        int currentMaxIndex = 0;
        for (int j = 1; j <= i; j++) {
            if (currentMax < list[j]) {
                currentMax = list[j];
                currentMaxIndex = j;
            }
        }
        // Swap list[i] with list[currentMaxIndex] if necessary;
        if (currentMaxIndex != i) {
            list[currentMaxIndex] = list[i];
            list[i] = currentMax;
        }
    }
}
```

Selection Sort - implementation

- Let’s implement selection sort in Java!

SelectionSort.java

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

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

```
2 9 5 4 8 1 6
2 5 9 4 8 1 6
2 4 5 8 9 1 6
1 2 4 5 8 9
```

Insertion Sort - implementation

- Let's implement insertion sort in Java!

InsertionSort.java

Practice problems

- Searching and sorting
  - 6.17, 6.18, 6.19

Enjoy the spring break!

Summary

- Search
  - Linear search
  - Binary search
- Sort
  - Selection sort
  - Insertion sort

Demos for various sorting algorithms including selection sort and insertion sort

http://www.cs.uwaterloo.ca/~bwbecker/sortingDemo/
http://www.sorting-algorithms.com/