Arrays

CSGE601021 Dasar-Dasar Pemrograman 2

Fakultas Ilmu Komputer Universitas Indonesia

Motivation

Imagine storing a number of integers, what you might do:

```
int num0 = 8;
int num1 = 0;
int num2 = 9;
int num3 = 10;
...
```

This is not convenient!
Arrays make programming lives much easier.

Motivation

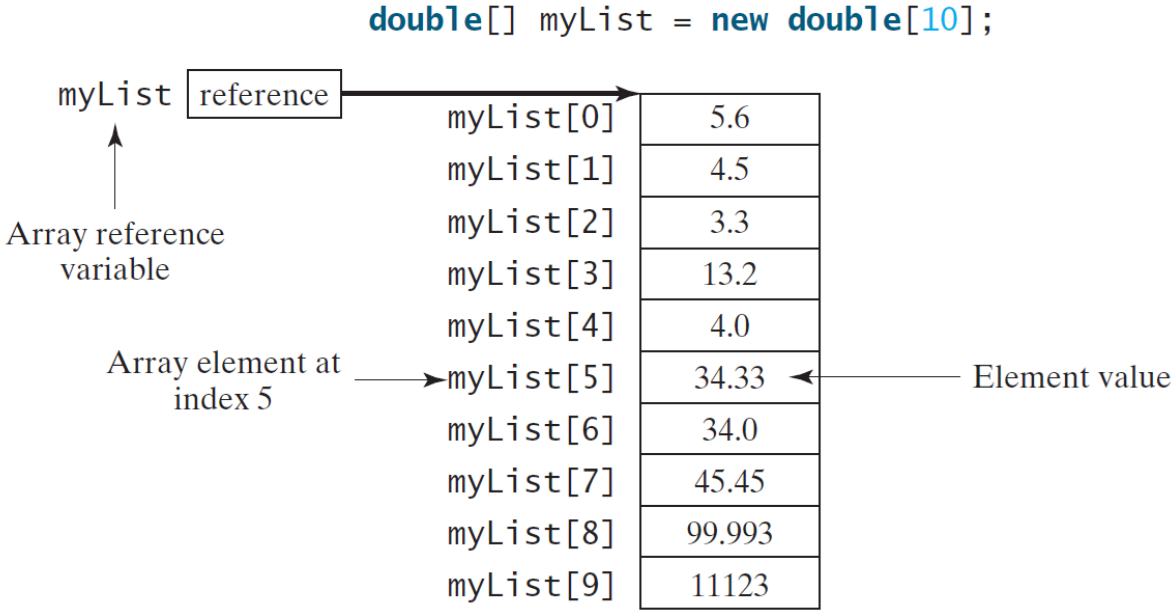
- The variables we have seen so far are for storing individual values, such as numbers, or strings.

- Now, what if we want to store multiple values of the same type?

Answer: Arrays!

Introducing Arrays

- Array is a data structure that represents a **collection** of the **same types** of data.
- The values in an array are called **elements**.



Declaring Arrays

- `datatype[] arrayVar;`

  Example: `double[] myList;`

- `datatype arrayVar[];` // This style is allowed, but not preferred

  Example: `double myList[];`

Creating Arrays

- To create the array itself, you have to use the **new** operator

```
arrayVar = new datatype[arraySize];
```

Example: `myList = new double[10];` → array index ranges from 0 to 9

`myList[0]` → the first element in the array.

`myList[9]` → the last element in the array.

Creating Arrays in One Step

- `datatype[] arrayVar = new datatype[arraySize];`

  Example: `double[] myList = new double[10];`

Quiz time

Create an array of 26 chars, and an array of 11 booleans!

```
char[] x = new char[26];
boolean[] y = new boolean[11];
```

Default Values

When an array is created, its elements are assigned the default value of:

- 0 for the **numeric primitive** data types,
- '\u0000' for **char** types, and
- false for **boolean** types.

Creating Arrays with content in one statement

```
int[] myInts = {9,1,7,7};
```

Guess whether each of the following array initialization is OK or NOT.

```
int[] myInts;
myInts = {9,1,7,7};
```

ERROR!

```
int[] myInts;
myInts = new int[]{9,1,7,7};
```

OK

```
int[] myInts = new int[]{9,1,7,7};
```

OK

```
int[] myInts = new int [4];
myInts[0] = 9;
myInts[1] = 1;
myInts[2] = 7;
myInts[3] = 7;
```

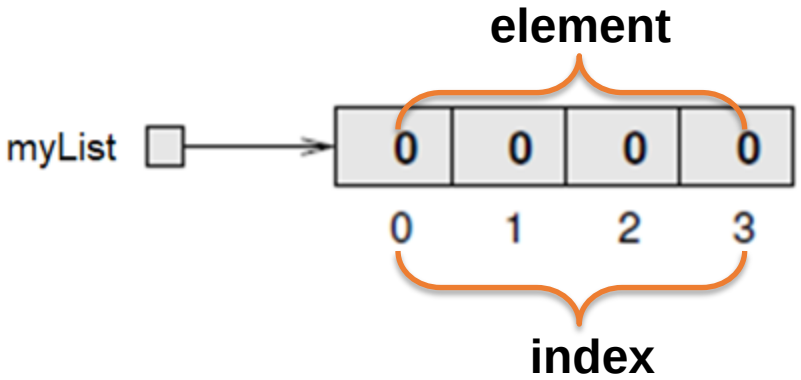
OK

The Length of an Array

Once an array is created, its size is **fixed** (**cannot be changed)**. You can find its size using `arrayVar.length`

Accessing elements

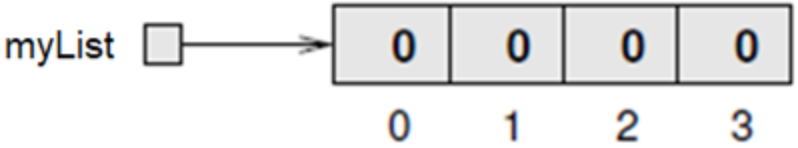
- When creating an array of ints, the elements are initialized to zero.



- The **[ ]** operator selects elements from an array.
- What's the output of:

```
System.out.println(“The zeroth element is ” + myList[0]);
```

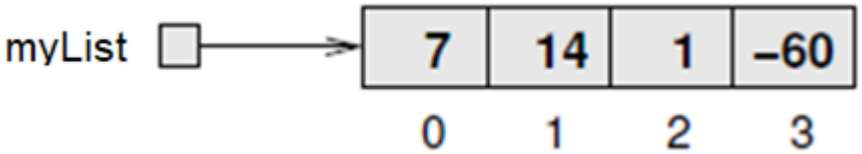
Manipulating array elements



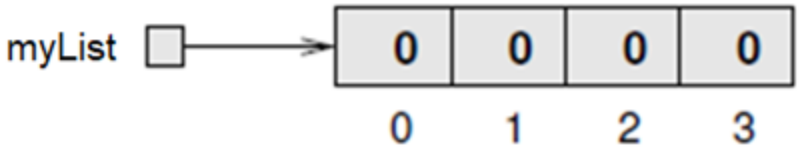
What are the contents of array myList after executing the following statements?

```
myList[0] = 7;
myList[1] = myList[0] * 2;
myList[2]++;
myList[3]-=60;
```

Answer:

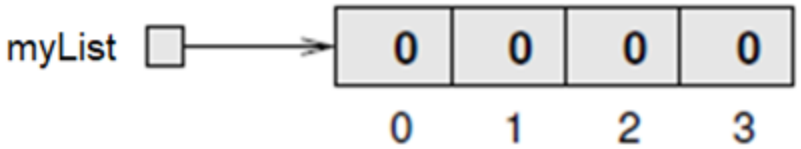


Visiting every element of an array




```
for (int i=0; i<4; i++){
        System.out.println(myList[i];)
}
```

Visiting every element of an array (oops)




```
for (int i=0; i<5; i++){
        System.out.println(myList[i];)
}
```

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 4 out of bounds for length 4

Foreach loop (enhanced for loop)

```
for (dataType value: arrayVar) {

  // Process the value

}
```

Example:

```
int[] myInts = new int[]{9,1,7,7};
for(int element:myInts) {
    System.out.println(element);
}
```

Output:

```
9
1
7
7
```

Foreach loop (enhanced for loop)

```
int[] myInts = new int[]{9,1,7,7};
for(int element:myInts) {
    System.out.println(element);
}
```

Output:

```
9
1
7
7
```

Foreach loop is best when:
- No need to manipulate array content.
- No need to use the indexes for something.

Displaying arrays

Be careful when printing arrays:

```
int[] myInts = new int[]{9,1,7,7};
System.out.println(myInts);
```

Output: `[I@3ac3fd8b`

Displaying arrays

Instead, do this:

```
import java.util.Arrays;

int[] myInts = new int[]{9,1,7,7};
System.out.println(Arrays.toString(myInts));
```

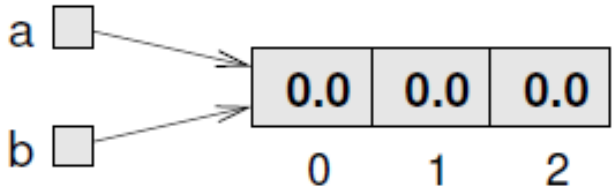
Output: `[9, 1, 7, 7]`

Copying arrays

Or copying references? Have a look below:

```
double[] a = new double[3];
double[] b = a;
```

Here's what really happens:



Copying arrays

Or copying references? Have a look below:

```
int[] myInts = new int[]{9,1,7,7};
int[] copyInts = myInts;

System.out.println(myInts);
System.out.println(copyInts);
```

Copying references of `myInts` to `copyInts`

Output:

```
[I@3ac3fd8b
[I@3ac3fd8b
```

Quiz time: Guess the output

```
int[] myInts = new int[]{9,1,7,7};
int[] copyInts = myInts;
copyInts[2] = 66;
System.out.println(Arrays.toString(myInts));
System.out.println(Arrays.toString(copyInts));
```

Output:

[9, 1, 66, 7]
[9, 1, 66, 7]

Copying arrays, now for real

```
int[] myInts = new int[]{9,1,7,7};
int[] copyInts = new int[4];

for(int i = 0; i<myInts.length; i++)
    copyInts[i] = myInts[i];

copyInts[2] = 66;
System.out.println(Arrays.toString(myInts));
System.out.println(Arrays.toString(copyInts));
```

Copying elements of `myInts` to `copyInts`

What is the output? [9, 1, 7, 7]
[9, 1, 66, 7]

A better way to copy arrays

```
int[] myInts = new int[]{9,1,7,7};
int[] copyInts = Arrays.copyOf(myInts, myInts.length);
```

The second parameter is the number of elements you want to copy, so you can also use `copyOf` to copy just part of an array.

Alternative copying arrays

What is the major difference between `System.arraycopy(...)` with `Arrays.copyOf(...)` ?

```
int[] myInts = new int[]{9,1,7,7};
int[] copiedInts = new int[3];
System.arraycopy(myInts, 1, copiedInts, 0, 2);
System.out.println(Arrays.toString(copiedInts));
```

Parameters for `System.arraycopy(sourceArr, sourcePos, destArr, destPost, length)`:

- `sourceArr`: array to be copied from
- `sourcePos`: starting position in source
- `destArr`: array to be copied in
- `destPos`: starting position in destination
- `length`: length of array to be copied

Quiz time: What goes wrong?

```
int[] myInts = new int[]{9,1,7,7};
System.out.println(myInts[myInts.length]);

int[] myInts2 = null;
System.out.println(myInts2[1]);

Object[] myInts3 = new Integer[3];
myInts3[2] = new String("a");
```

Quiz time: What's this code doing?

```
for (int i = 0; i < a.length; i++) {
    a[i] = Math.pow(a[i], 2.0);
}
```

Quiz time: What's this code doing?

```
public static int s(int[] a, int target) {
    for (int i = 0; i < a.length; i++) {
        if (a[i] == target) {
            return i;
        }
    }
    return -1;
```

Exercise

- Write a method **printEven** to print only array elements of even indexes.
- Write a method **sum** that returns the sum of an array of integers.
- Write a method **maxArray** to find the largest element in an array of int!
- Write a method **shiftLeftArray** to shift each element in an array of int to the left. The first element will be moved to the last element!

Sorting arrays

```
int[] myInts = new int[]{9,1,7,7};
Arrays.sort(myInts);
System.out.println(Arrays.toString(myInts));
```

The sorting is done **in-place,**
no need to return to any variable.

Quiz time: Sorting arrays in descending order

```
public static int[] sortDescending(int[] arr) {
    int[] copyArr = Arrays.copyOf(arr,arr.length);
    Arrays.sort(copyArr);
    return reverse(copyArr);
}

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

Filling arrays

```
int[] intArr = new int[3];
Arrays.fill(intArr, 100);
System.out.println(Arrays.toString(intArr));
Arrays.fill(intArr, 1,3,7);
System.out.println(Arrays.toString(intArr));
```

Output:

[100, 100, 100]
[100, 7, 7]

Check array content equality

```
int[] intArr = new int[3];
Arrays.fill(intArr, 100);
int[] intArrAnother = new int[3];
Arrays.fill(intArrAnother, 100);
System.out.println(intArr == intArrAnother);
System.out.println(Arrays.equals(intArr, intArrAnother));
```

Output: false
true

Pass by Value

- 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

```
public static void main(String[] args){
    int intA = 5;
    int[] intArr = {1,3,2};
    mystery(intA, intArr);
    System.out.println(intA)
    System.out.println(Arrays.toString(intArr));
}

public static void mystery(int a, int[] arr){
    a = 10;
    arr[1] = 899;
}
```

What is the output ? 5

[1, 899, 2]

Enlarge array capacity

You can't. What you can do is to copy your array to a larger array.

```
int[] arr = {5,1,2,1,3};
int[] bigArr = new int[10];
for(int i = 0; i < arr.length; i++) {
    bigArr[i] = arr[i];
}
arr = bigArr;
System.out.println(Arrays.toString(arr));
```

Command-Line Parameters

```
class TestMain {
  public static void main(String[] args) {
  ...
  }
}

java TestMain arg0 arg1 arg2 ... argn
```

In the main method, get the arguments from `args[0], args[1], ..., args[n]`, which corresponds to `arg0, arg1, ..., argn` in the command line.


Multidimensional
Arrays

Motivation

- So far our arrays are one-dimensional:

```
double[] arr = new double[]{0.5,2.5,2.0,5.0};
```

- Suppose you want to store the content of a numeric **table or** a **matrix**, then we have to go deeper, adding more dimensions!

```
double[][] twoDArr = {{0.5,2.5,2.0,5.0},
                      {1.5,0.5,1.0,7.0},
                      {3.5,1.5,3.0,1.0}};
```

Two dimensional array

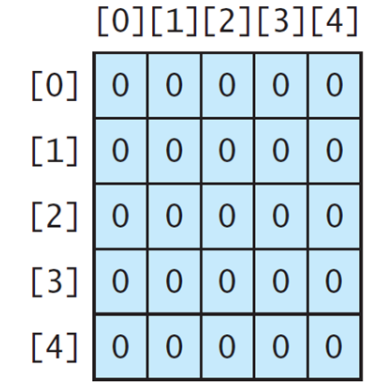
Declare/Create Two-dimensional Arrays

```
// Declare array variable
dataType[][] arrayVar;

// Create array and assign its reference to variable
arrayVar = new dataType[10][10];

// Combine declaration and creation in one statement
dataType[][] arrayVar = new dataType[10][10];

// Alternative syntax
dataType arrayVar[][] = new dataType[10][10];
```

FAKULTAS ILMU KOMPUTER

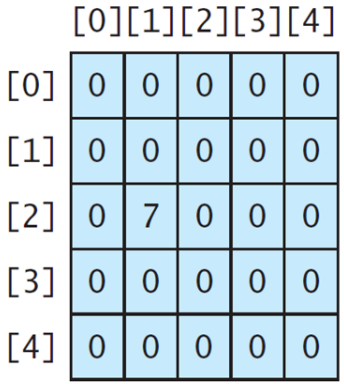
Two-dimensional Array Illustration

```
int[][] matrix;
```



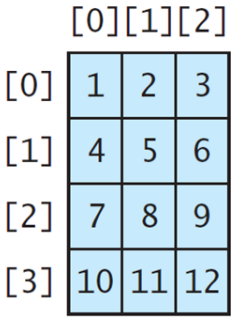

```
matrix = new int[5][5];
```

(a)




```
matrix[2][1] = 7;
```

(b)




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

(c)

matrix.length ? 5

matrix[0].length ? 5

array.length ? 4

array[0].length ? 3

Declaring, Creating, and Initializing Using Shorthand Notations

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

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

Same as

```
int[][] array = new int[4][3];
array[0][0] = 1; array[0][1] = 2; array[0][2] = 3;
array[1][0] = 4; array[1][1] = 5; array[1][2] = 6;
array[2][0] = 7; array[2][1] = 8; array[2][2] = 9;
array[3][0] = 10; array[3][1] = 11; array[3][2] = 12;
```

```
array.length ?
array[0].length ?
array[3][0].length ?
array[4].length ?
```

2-D arrays: Guess the output?

```
double[][] twoDArr = {{0.5,2.5,2.0,5.0},
                      {1.5,0.5,1.0,7.0},
                      {3.5,1.5,3.0,1.0}};


System.out.println(Arrays.toString(twoDArr));
System.out.println(Arrays.toString(twoDArr[1]));
System.out.println(Arrays.toString(twoDArr[1][3]));
System.out.println(twoDArr[1][3]);
System.out.println(twoDArr[2][1]);
```

Quiz time

Create a method **print2D** to print the content of a 2D-array of doubles!

For example, given:

```
double[][] twoDArr = {{0.5,2.5,2.0,5.0},
                      {1.5,0.5,1.0,7.0},
                      {3.5,1.5,3.0,1.0}};
```

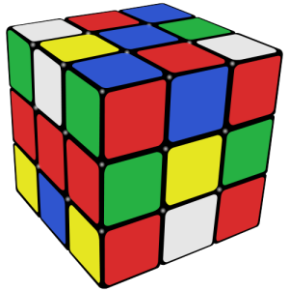
`print2D(twoDArr)` will print:

```
0.5 2.5 2.0 5.0
1.5 0.5 1.0 7.0
3.5 1.5 3.0 1.0
```

Quiz time

Create a method **sum2D** to sum all elements in a two dimensional array !

3-D arrays



- Yes, we can go further.
- Previously, our arrays contain arrays. Now, what if our arrays contain arrays of arrays?

```
int[][][] threeDArr1 =  new int[3][3][3];
int[][][] threeDArr2 = {{{1,2,3},{3,2,1},{2,1,5}},
                        {{5,2,5},{1,1,1},{7,1,0}},
                        {{4,6,7},{4,5,4},{4,6,6}}};
```

FAKULTAS ILMU KOMPUTER

Quiz time

What's the output?

```
int[][][] threeDArr1 = new int[3][3][3];
int[][][] threeDArr2 = {{{1,2,3},{3,2,1},{2,1,5}},
                        {{5,2,5},{1,1,1},{7,1,0}},
                        {{4,6,7},{4,5,4},{4,6,6}}};

System.out.println(threeDArr1[0][0][1]);
System.out.println(Arrays.toString(threeDArr1[0][0]));
System.out.println(threeDArr2[0][1][2]);
System.out.println(threeDArr2[2][1][1]);
System.out.println(Arrays.toString(threeDArr2[1][2]));
System.out.println(Arrays.toString(threeDArr2[0]));
```

Output:

0
[0, 0, 0]
1
5
[7, 1, 0]
[[I@3ac3fd8b, [I@5594a1b5, [I@6a5fc7f7]