UNIVERSITAS INDONESIA

Veritas, Probitas, Iustitia

FAKULTAS ILMU KOMPUTER

Arrays

Dasar – Dasar Pemrograman 2

Dinial Utami Nurul Qomariah

Credits

- Liang, Introduction to Java Programming, 11th Edition, Ch. 2
- Downey & Mayfield, Think Java: How to Think Like a Computer Scientist, Ch. 2
- Slide Kuliah Dasar-dasar Pemrograman 2 Semester Genap 2021/2022

Motivations

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

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

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**!

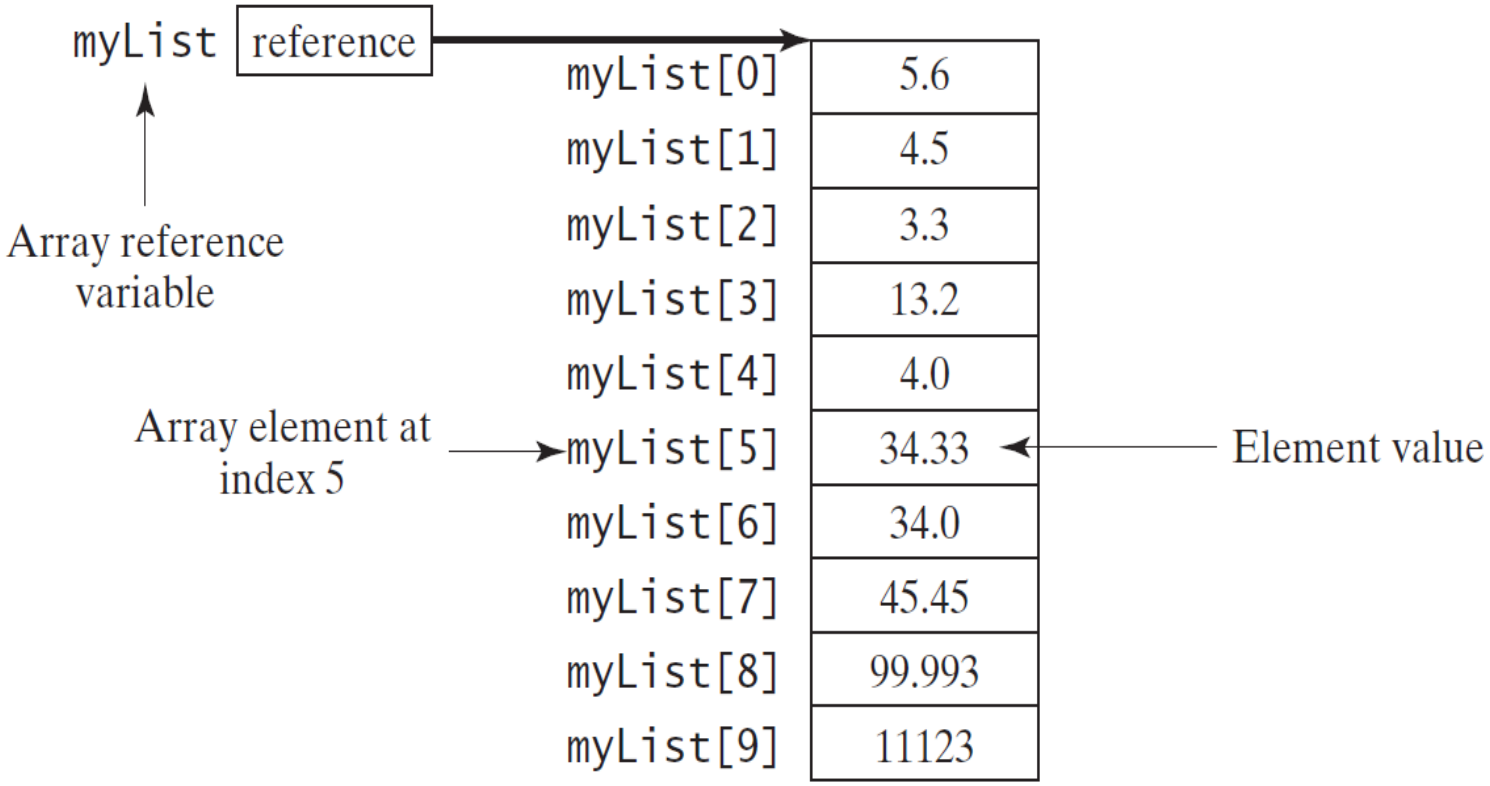
- This is not convenient!
  Arrays make programming lives much easier.

Array Satu Dimensi

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**.
- Once an array is created, **its size is fixed**.
  An array reference variable is used to access the elements in an array using an index.

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



Declaring Array Variables

- `datatype[] arrayRefVar;`

  Example:

  ```
  double[] myList;
  ```

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

  Example:

  ```
  double myList[];
  ```

Creating Arrays

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

Example:

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

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

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

Declaring and Creating in One Step

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

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

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

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

Quiz time

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

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.

null for **reference** types.

Creating Arrays with element in one statement

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

Yang terjadi Error?

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

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

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

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

Creating Arrays with element in one statement

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

Yang terjadi Error?

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

Error

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

OK

```
int myInts = [] 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. It cannot be changed. You can find its size using

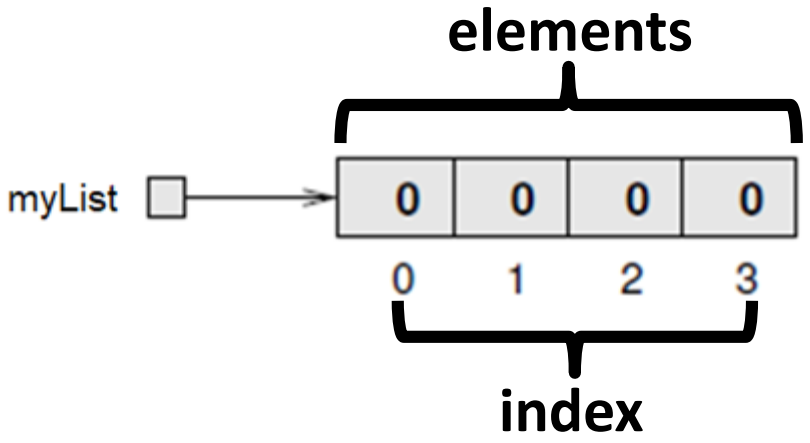
```
arrayRefVar.length
```

For example,

```
myList = new double[10];
myList.length → returns 10
```

Accessing elements of Arrays

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

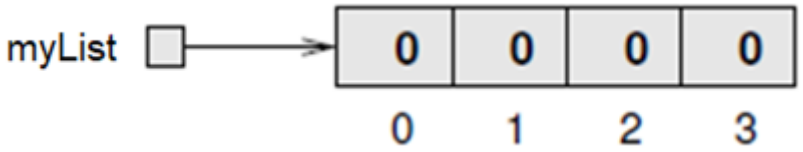


The **[ ]** operator selects elements from an array

What's the output of:

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

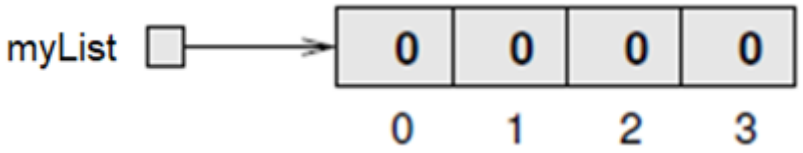
Manipulating array elements



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;
```

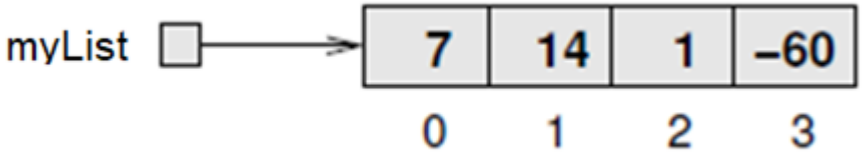
Manipulating array elements



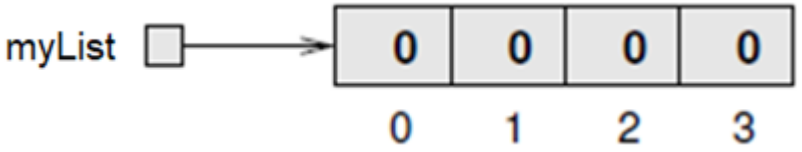
What are the contents of array myList after executing the following statements?

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

Output

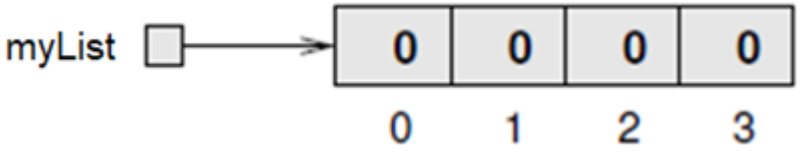


Printing every element of an arrays




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

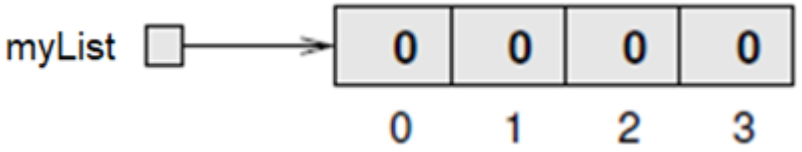
Printing every element of an arrays




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

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

Printing every element of an arrays Solution




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

Enhanced for Loop (for-each loop)

General syntax is

```
for (elementType value: arrayRefVar) {
  // 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
```

Enhanced for Loop (for-each loop)

Foreach loop is best when:

- No need to manipulate array content.
- No need to use the indexes for something.

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

Output

```
9
1
7
7
```

Exercise time

- Write a method **printEven** to print only array elements of even indexes.

Exercise time

- Write a method **printEven** to print only array elements of even indexes.

```
public static void printEven(int[] intArray) {

    for(int i = 0; i < intArray.length; i = i + 2)
        System.out.println(intArray[i]);

}
```

Displaying arrays

- Be careful when printing arrays:

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

Output

```
[I@3ac3fd8b
```

Displaying arrays

- Solution

```
import java.util.Arrays;

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

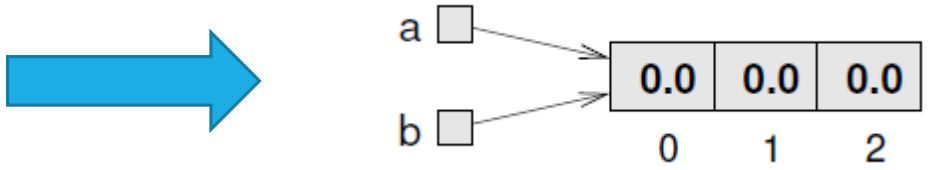
- Class **java.util.Arrays** berisi method method static yang mendukung manipulasi array, seperti perbandingan array, sorting, searching, dsb.

Copying arrays

Copying Array Or copying references? Have a look below:

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

Here's what really happens:



Saat mengcopy array maka yang dicopy bukan element arraynya tapi adress/lokasi memorinya. (pass by references). **Be Carefull**

Copying arrays

Ada tiga cara untuk menyalin isi suatu array ke dalam array lain supaya hubungan kedua array saling lepas:

- Assign nilai setiap elemen arr2 dengan nilai dari arr1.
- Menggunakan **Arrays.copyOf()**.
- Menggunakan **System.arrayCopy()**.

Copying arrays

`Copying Array` 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));
```

Copying arrays : Guess the output?

```
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`

Fast way to Copy Arrays

- use copyOf to copy just part of an array.

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

Alternative copying arrays

```
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

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

Quiz time: What goes wrong?

Case 1

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

Case 2

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

Case 3

```
int[] a = new int[]{9,1};
for (int i=0; i<a.length; i++){
a[i] = Math.pow(a[i], 2.0);
}
```

Quiz time: What The output?

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

Quiz time:

Create a method that returns the sum of an array of integers.

Quiz time: What The output?

```
public static int sum(int[] a) {
int total = 0;
for (int i = 0; i < a.length; i++) {
   total += a[i];

}
return total;
```

Create a **foreach loop** version of the above method!

Exercise

Write a method **maxArray** to find the largest element in an array of int!

Exercise

Write a method **maxArray** to find the largest element in an array of int!

```
public static int maxArray (int[] myList) {
   int max = myList[0];
   for (int i = 1; i < myList.length; i++) {
      if (myList[i] > max)
         max = myList[i];
   }
   return max;
}
```

Exercise

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!

Exercise

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!

```
public static void shiftLeftArray (int[] myList)
   int temp = myList[0];
   for (int i = 1; i < myList.length; i++) {
      myList[i-1]=myList[i];
   }
   myList[i]=temp;
}
```

Sorting arrays

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

Quiz time: Sorting arrays in descending order

```
public static int[] sortDescending(int[] arr) {
// Implement the method
}
```

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));
```

Membandingkan Alamatnya

Membandingkan Value/nilai elementnya

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.

Guess the output

```
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;
}
```

Guess the output

```
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;
}
```

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));
```

Method “main” pada main class

```
public class Test {
    public static void main(String[] args){
    //statements
    }
}
```

Elements dari array “args”
Tipe data berupa String

Pada method main, sebenarnya menampung argument-argument dari args[0] – args[n]