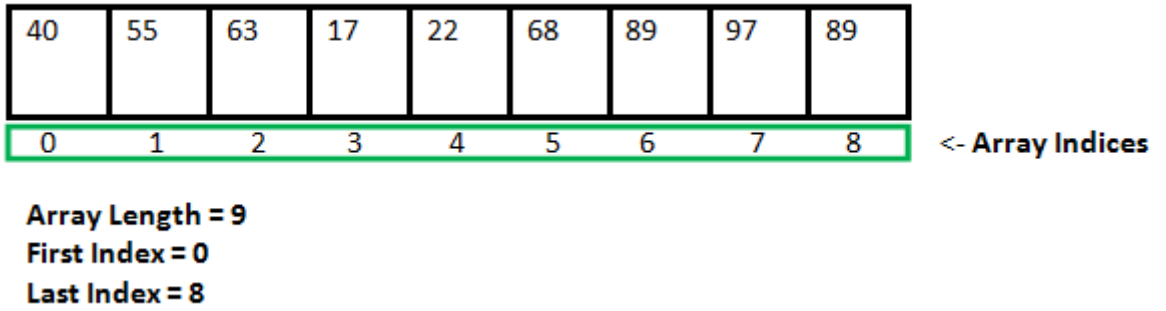
Arrays

Mengkang Li (122090264@link.cuhk.edu.cn)

Overview

- Data Structure
  - Why is data structure important?
- Array
  - Creating an array
  - Accessing data in an array
- Practice



Why we need arrays?

- Group data: use a single name to access several data.
- Sequential access: those data is associated with indices.
- Efficient storage: Java assigns fixed space for an array.

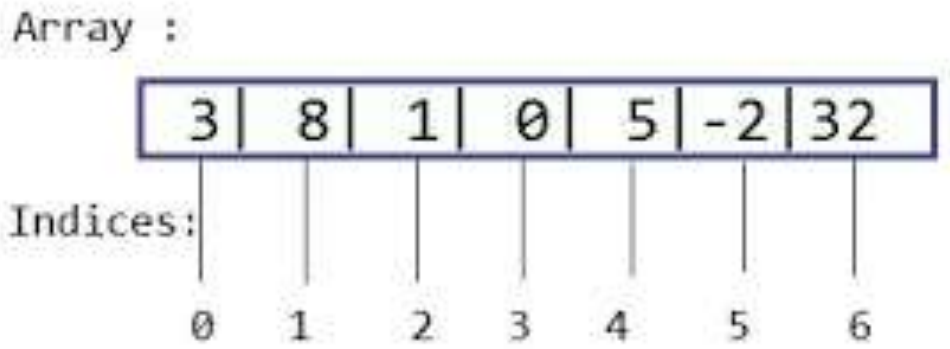
```java
int[] fibonacci_series = new int[n];
for (int i=0; i<n; i++) {
    fibonacci_series[i] = fibonacci_series[i-1]+fibonacci_series[i-2];
}
for (int i=n-1; i>n-d; i--){
    System.out.println(fibonacci_series[i]);
}
```

Arrays

A new variable type is an object that represents an ordered, homogeneous list of data.

– Arrays have many el

you can access using indic



Creating an array

type[] ***name*** = new ***type***[***length***];

int[] numbers = new int[5]; //Create an array with five integer elements

index	0	1	2	3	4
value	0	0	0	0	0

Creating an array

```
double[] results = new double[5];

String[] names = new String[3];

boolean[] switches = new boolean[4];
```

- Java initializes each element of a new array to its default value, which is **0** for int and doubles, **‘\0’** for char, **false** for boolean, and **nu**ll for objects.

Initialization of Java Arrays

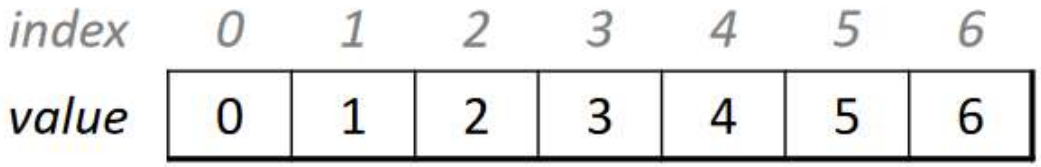
- Java initialize arrays in different ways corresponding to different arrays.
- int: 0
- double: 0.0
- char: ‘\0’
- boolean: false
- object: null

Accessing Data In An Array

```
name[index] // get element at index
```

- Indices go from 0 to the array's length - 1.

```
for (int i = 0; i < 7; i++) {
    println(numbers[i]);
}
println(numbers[9]); // exception
println(numbers[-1]); // exception
```



Putting data in an array

name[index] = value; // set element at index

- Like Strings, indices go from 0 to the array's length - 1.

```
int[] numbers = new int[7];
for (int i = 0; i < 7; i++) {
    numbers[i] = i;
}
```

Index	0	1	2	3	4	5	6
Value	0	1	2	3	4	5	6

Array length

You can get the length of an array by saying
myArray.length

Practice:

What is the index of the last element of an array in terms of its length?

What is the index of the middle element of an array in terms of its length

A tricky example : what is the output of the following program and why?

```java
public class Test {
    Run | Debug
    public static void main(String[] args) {
        int[] a = {1, 2, 3, 4};
        int[] b = new int[4];
        int[] c = new int[5];
        b = a;
        c = a;
        b[1] = 0;
        c[2] = 0;
        c = new int[3];
        for (int i : a) {
            System.out.println(i);
        }
        for (int i : c) {
            System.out.println(i);
        }
    }
}
```

Practice1: Swapping Elements

Let’s write a method called swapElements that swaps two elements of an array. How can we do this?

What parameters should it take (if any)? What should it return (if anything)?

```
public void run() {
      int[] array = new int[5];
      ...
      swapElements(array[0], array[1]);
      ...
}

private void swapElements(int x, int y) {
      int temp = x;
      x = y;
      y = temp;
}
```

Why it doesn’t work?

Ints are primitives, so they are passed by value!

Their variable boxes store their actual values.

So changes to the parameter do not affect the original.

```
private void swapElements(int x, int y) {
      int temp = x;
      x = y;
      y = temp;
}
```

Another method

```
public void run() {
	int[] array = new int[5];
	...
	swapElements(array, 0, 1);
	...
}


private void swapElements(int[] arr, int pos1, int pos2) {
	int temp = arr[pos1];
	arr[pos1] = arr[pos2];
	arr[pos2] = temp;
}
```

Why this method works?

Arrays are all objects in the java. And java passes all objects by value. So, the method shouldn’t change the value of the array.

But When you pass an array to another method, actually the reference to that array is copied.

- Any changes in the content of the array through that reference will affect the original array.
- But changing the reference to point to a new array will not change the existing reference in the original method.

Practice2: Reverse an array.

Let’s write a method called “reverse” that takes an int array, and return the reverse array (element-wise) of the array.

Practice3: Merge two sorted array.

Write a method called “merge” that merges two sorted increasing arrays of integers into a single sorted increasing array.

Practice4: Sort an array.

Write a method called “sort” that sort an int array increasingly.

Q&A