



香港中文大學 (深圳)  
The Chinese University of Hong Kong

# CSC3100 Data Structures

## Lecture 3: Array

Yixiang Fang  
School of Data Science (SDS)  
The Chinese University of Hong Kong, Shenzhen

---



# Outline

---

- ▶ Overview
- ▶ Concepts of arrays
- ▶ ADT of Arrays
- ▶ Implementation
- ▶ Examples



# Array

- ▶ Arrays are among the oldest and most important data structures
  - Arrays are supported by almost every programming language
  - Arrays are used for representing vectors/matrices
  - The simplest type of array is a linear array, or one-dimensional array



Array:

23	4	6	15	5	7
0	1	2	3	4	5

↑  
Array index





# Array

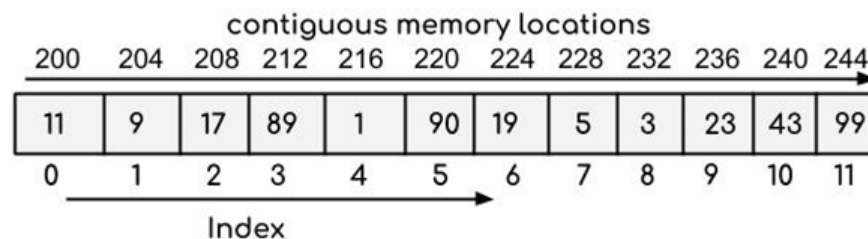
---

- ▶ An array **stores** the **same** type of objects together, and **access** the objects by their **indices**
- ▶ Pro & Con of arrays:
  - Pro: If you know the index (**where**), you can find the object (**content**) with one basic operation (**efficient**); Efficient search if array is sorted
  - Con: If you insert an object in a place between two objects in an array, you have to move the objects first; The capacity is fixed



# Concepts

- ▶ The individual values in an array are called **elements**; all the elements must be of the **same type**
- ▶ The number of elements is called the **length** of the array, which is fixed when the array is created
- ▶ Each element is identified by its position in the array, which is called **index** (In C/C++/Java, the index numbers begin with 0)
- ▶ An array is a linear data structure that hosts a collection of data elements stored at **consecutive locations** in a computer's memory
  - The memory address of the first element of an array is called first address, foundation address, or base address
  - The memory position of each element can be computed from its index





# Array ADT

1	ADT Array
2	Array createArray(n)
3	Item retrieve(arr, i)
4	Item store(arr, i, itemToStore)

- ▶ **createArray(n)**
  - Initialized an array of size  $n$  to store item
- ▶ **retrieve(arr, i)**
  - $arr[i]$ ,
  - Return the item stored in the  $i$ -th position of the array
- ▶ **store(arr, i, itemToStore)**
  - Store  $itemToStore$  to the  $i$ -th position of the array
  - $arr[i] = itemToStore$



# Array ADT: example

- ▶ We can design algorithms without knowing its underlying implementation
  - E.g., design a linear search algorithm with an array using ADT

## Algorithm 1: *Linear Search*

Input: An array *a* of integers with length *n*, an integer *searchnum*

Output: the index *i* such that the value stored in the *i*-th position equals *searchnum*, or -1 if no such *i* exists

```
1  int i;
2  for (i=0; i<n; i++){
3      if( retrieve(a,i) == searchnum )
4          return i;
5  }
6  return -1;
```



# Array ADT: example

- ▶ We can design algorithms without knowing its underlying implementation
  - E.g., implement the dimension-product given two arrays  $a_1$  and  $a_2$  and output to  $a_3$  such that  $a_3[i] = a_1[i] \cdot a_2[i]$

## Algorithm 2: *Dimension Product*

Input: Vector  $a_1, a_2, a_3$  of length  $n$

Output: Dimension-product of  $a_1$  and  $a_2$  which is stored in  $a_3$

```
1  int i,i_dimension_coordinate;
2  for (i=0; i<n; i++){
3      i_dimension_coordinate = retrieve( $a_1$ ,i)*retrieve( $a_2$ ,i);
4      store( $a_3$ , i, i_dimension_coordinate);
5  }
6  return  $a_3$ ;
```





# Java: array declaration

- ▶ An array is characterized by
  - Element type
  - Length: `type[ ] identifier = new type[length];`
- ▶ Default values in initialization
  - numerics 0
  - boolean false
  - objects null



Elements of an array can be objects of any Java class



Example: An array of 5 instances of the student class

```
Student [ ] topStudents = new Student[5];
```



# Java: array operations

---

- ▶ Use named constant to declare the length of an array, or read the length of an array from the user

```
private static final int N_JUDGES = 5;  
double[ ] scores = new double[N_JUDGES];
```

- ▶ Identifying an element by an integer number or an expression  
array[index], array[(a+b)/2]

- ▶ Cycling through array elements

```
for (int i = 0; i < array.length; i++) {  
    operations involving the ith element;  
}  
for (int value:array)  
{  
    operations involving values;  
}
```



# Array initialization

---

- ▶ A convenient way of initializing an array:

```
int[ ] digits = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
```

```
String[ ] US_CITIES_OVER_ONE_MILLION = {  
    "New York",  
    "Los Angeles",  
    "Chicago",  
    "Huston",  
    "Philadelphia",  
    "Phoenix",  
}
```

- ▶ Starting index numbering at 0 can be confusing, so we use two standard ways:
  - Use Java's index number internally, and then add one when presenting to the user
  - Use index values beginning at 1, and ignore the first (0) element in each array



# Pass-by-Value vs Pass-by-Reference

swapElements(array[i], array[n - i - 1]) **[wrong]**

swapElements(array, i, n - i - 1)

```
private void swap(int[] array, int p1, int p2) {  
    int tmp = array[p1];  
    array[p1] = array[p2];  
    array[p2] = tmp;  
}
```

## ▶ What is **Pass-by-Value**?

- The value of a function parameter is copied to another location of the memory
- When accessing or modifying the variable within the function, it accesses only the copy, so there is no effect on the original value

## ▶ What is **Pass-by-Reference**?

- The memory address is passed to that function, so the function gets access to the actual variable



# Examples

```
Object obj = new Object();
```



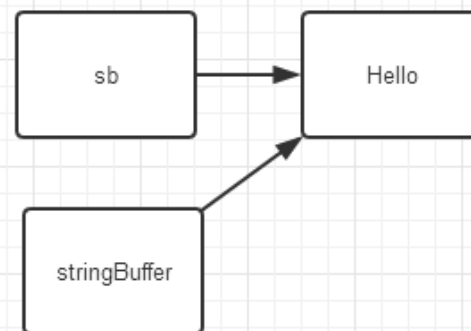
```
public class test {  
    public static void main(String[] args) {  
        int i = 1;  
        System.out.println("before change, i = "+i);  
        change(i);  
        System.out.println("after change, i = "+i);  
    }  
    public static void change(int i){  
        i = 5;  
    }  
}
```



```
before change, i = 1  
after change, i = 1
```



```
public class test {  
    public static void main(String[] args) {  
        StringBuffer sb = new StringBuffer("Hello ");  
        System.out.println("before change, sb is "+sb.toString());  
        change(sb);  
        System.out.println("after change, sb is "+sb.toString());  
    }  
    public static void change(StringBuffer stringBuffer){  
        stringBuffer.append("world !");  
    }  
}
```



```
before change, sb is Hello  
after change, sb is Hello world !
```



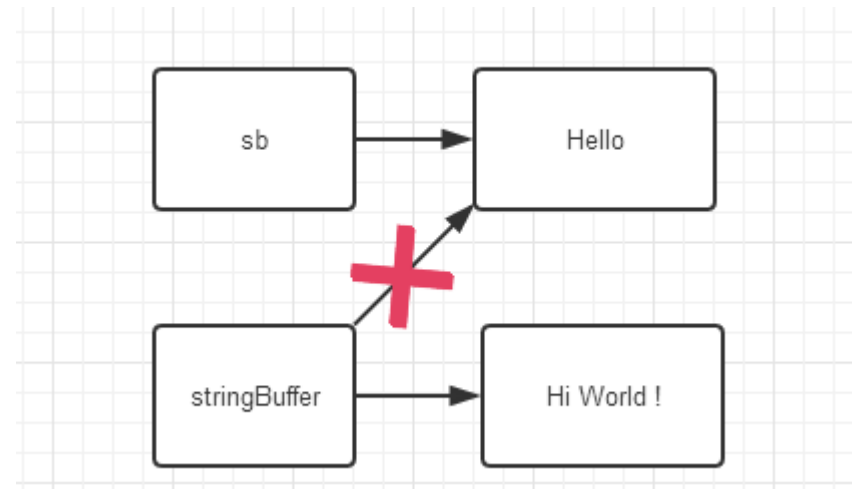
# Examples



```
public class test {  
    public static void main(String[] args) {  
        StringBuffer sb = new StringBuffer("Hello ");  
        System.out.println("before change, sb is "+sb.toString());  
        change(sb);  
        System.out.println("after change, sb is "+sb.toString());  
    }  
    public static void change(StringBuffer stringBuffer){  
        stringBuffer = new StringBuffer("Hi ");  
        stringBuffer.append("world !");  
    }  
}
```



```
before change, sb is Hello  
after change, sb is Hello
```





# Examples

```
1 public class MyClass {
2     public int x, y;
3
4     public MyClass(int a, int b){
5         x = a;
6         y = b;
7     }
8     public static void swap(MyClass obj_a, MyClass obj_b){
9         MyClass tmp = obj_a;
10        obj_a = obj_b;
11        obj_b = tmp;
12
13        System.out.println("# obj_a: x=" + obj_a.x + " y=" + obj_a.y);
14        System.out.println("# obj_b: x=" + obj_b.x + " y=" + obj_b.y);
15        System.out.println();
16    }
17    public static void main(String args[]) {
18        MyClass obj1 = new MyClass(1, 2);
19        MyClass obj2 = new MyClass(10, 20);
20
21        MyClass.swap(obj1, obj2);
22
23        System.out.println("obj1: x=" + obj1.x + " y=" + obj1.y);
24        System.out.println("obj2: x=" + obj2.x + " y=" + obj2.y);
25    }
26 }
```

```
# obj_a: x=10 y=20
# obj_b: x=1 y=2

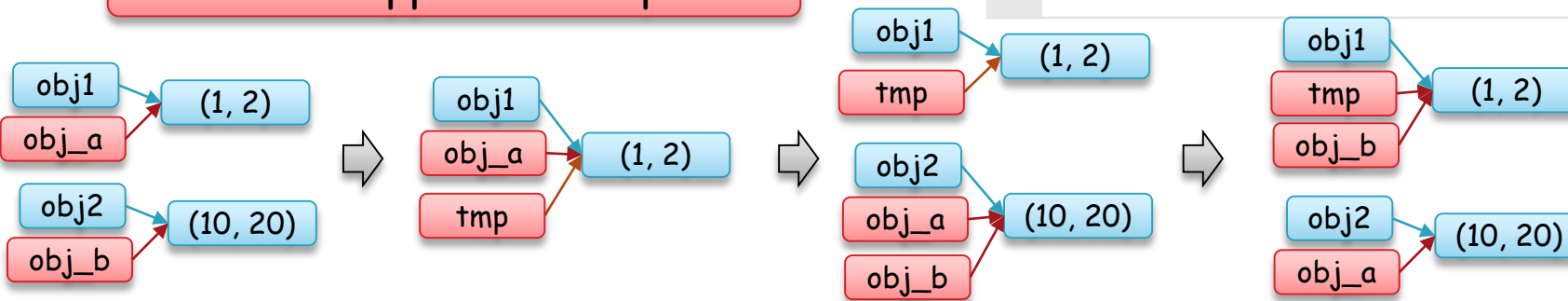
obj1: x=1 y=2
obj2: x=10 y=20
```

```
1 public class MyClass {
2     public int x, y;
3
4     public MyClass(int a, int b){
5         x = a;
6         y = b;
7     }
8     public static void swap(MyClass obj_a, MyClass obj_b){
9         int tmp = obj_a.x;
10        obj_a.x = obj_b.x;
11        obj_b.x = tmp;
12
13        tmp = obj_a.y;
14        obj_a.y = obj_b.y;
15        obj_b.y = tmp;
16
17        System.out.println("# obj_a: x=" + obj_a.x + " y=" + obj_a.y);
18        System.out.println("# obj_b: x=" + obj_b.x + " y=" + obj_b.y);
19        System.out.println();
20    }
21    public static void main(String args[]) {
22        MyClass obj1 = new MyClass(1, 2);
23        MyClass obj2 = new MyClass(10, 20);
24
25        MyClass.swap(obj1, obj2);
26
27        System.out.println("obj1: x=" + obj1.x + " y=" + obj1.y);
28        System.out.println("obj2: x=" + obj2.x + " y=" + obj2.y);
29    }
30 }
```

```
# obj_a: x=10 y=20
# obj_b: x=1 y=2

obj1: x=10 y=20
obj2: x=1 y=2
```

What happens in swap ?





# Exercise: using arrays

---

- ▶ Letter frequency table:

- Given an array of **upper-case letters**,  
e.g., `A[ ]={'A', 'B', 'C', 'B', 'A'}`

How to count the frequency of each letter **efficiently**?

- ▶ How to use an array to store a set of elements whose size is larger than the maximum number of elements in Java array?

- Note that the maximum number of elements in a Java array is **`Integer.MAX_VALUE`**





# Two-dimensional arrays

---

- ▶ Each element of an array is an array (of the same dimension)
- ▶ E.g., a 3-by-2 matrix can be represented by a two-dimensional array:

```
int[][] A = new int[3][2];  
int A[3][2] = {{1, 4}, {2, 5}, {3, 6}};
```

The elements are  $A[0][0]$ ,  $A[0][1]$ ,  $A[1][0]$ ,  $A[1][1]$ ,  
 $A[2][0]$ ,  $A[2][1]$

The three one-dimensional arrays are  $A[0]$ ,  $A[1]$ ,  $A[2]$



# The ArrayList class

---

- ▶ The **java.util** package has a class called **ArrayList**
  - Provide standard array behaviors along with other useful operations
  - **ArrayList** is a Java class rather than a special form in the language
- ▶ All operations on **ArrayLists** are indicated using method calls
  - Create a new **ArrayList** by calling **ArrayList** constructor
  - Get the number of elements by calling the **size** method
  - Use the **get** and **set** methods to select individual elements



# More examples in Java

---

- ▶ Number arrays
  - int, float, double, ...
- ▶ String arrays
- ▶ Boolean arrays

```
1 public class Main
2 {
3     public static void main(String[] args)
4     {
5         // create an array
6         int[] age = {12, 4, 5};
7
8         // loop through the array
9         // using for loop
10        System.out.println("Using for-each Loop:");
11        for(int a : age)
12        {
13            System.out.println(a);
14        }
15    }
16 }
```

## Output

```
Using for-each Loop:
12
4
5
```

```

1  public class Main
2  {
3      public static void main(String[] args)
4      {
5
6          int[] numbers = {2, -9, 0, 5, 12, -25, 22, 9, 8, 12};
7          int sum = 0;
8          Double average;
9
10         // access all elements using for each loop
11         // add each element in sum
12         for (int number : numbers)
13         {
14             sum += number;
15         }
16
17         // get the total number of elements
18         int arrayLength = numbers.length;
19
20         // calculate the average
21         // convert the average from int to double
22         average = ((double)sum / (double)arrayLength);
23
24         System.out.println("Sum = " + sum);
25         System.out.println("Average = " + average);
26     }
27 }

```

## Output:

```

Sum = 36
Average = 3.6

```

```

1 public class MultidimensionalArray {
2     public static void main(String[] args)
3     {
4
5         int[][] a =
6         {
7             {1, -2, 3},
8             {-4, -5, 6, 9},
9             {7},
10        };
11
12        for (int i = 0; i < a.length; ++i)
13        {
14            for(int j = 0; j < a[i].length; ++j)
15            {
16                System.out.println(a[i][j]);
17            }
18        }
19    }
20 }

```

## Output:

```

1
-2
3
-4
-5
6
9
7

```

```

1 public class MultidimensionalArray
2 {
3     public static void main(String[] args)
4     {
5
6         // create a 2d array
7         int[][] a =
8         {
9             {1, -2, 3},
10            {-4, -5, 6, 9},
11            {7},
12        };
13
14        // first for...each loop access the individual array
15        // inside the 2d array
16        for (int[] innerArray : a)
17        {
18            // second for...each loop access each element inside the row
19            for(int data : innerArray)
20            {
21                System.out.println(data);
22            }
23        }
24    }
25 }

```

## Output:

```

1
-2
3
-4
-5
6
9
7

```

```

1 public class ThreeArray
2 {
3     public static void main(String[] args)
4     {
5
6         // create a 3d array
7         int[][][] test =
8         {
9             {
10                {1, -2, 3},
11                {2, 3, 4}
12            },
13            {
14                {-4, -5, 6, 9},
15                {1},
16                {2, 3}
17            }
18        };
19
20        // for..each loop to iterate through elements of 3d array
21        for (int[][] array2D : test)
22        {
23            for (int[] array1D : array2D)
24            {
25                for(int item : array1D)
26                {
27                    System.out.println(item);
28                }
29            }
30        }
31    }
32 }

```

Output:

```

1
-2
3
2
3
4
-4
-5
6
9
1
2
3

```





# Examples: String array

---

- ▶ String array is an array holding a fixed number of strings or string values
  - One structure commonly used in Java
  - Even the arguments of the Java 'main' function is a String Array
    - `public static void main(String args[ ]){...}`
- ▶ String array is an array of String objects
  - Each string is an object!

```

1 public class Main
2 {
3     public static void main(String[] args)
4     {
5
6         String[] myarray;      //declaration of string array without size
7         String[] strArray = new String[5]; //declaration with size
8
9         //System.out.println(myarray[0]); //variable myarray might not have been initialized
10        //display elements of second array
11        System.out.print(strArray[0] + " " + strArray[1] + " " + strArray[2] + " " +
12        strArray[3] + " " + strArray[4]);
13    }
14 }

```

```

null null null null null

```

```

1  import java.util.*;
2
3  public class Main
4  {
5      public static void main(String[] args)
6      {
7          //original array
8          String[] colorsArray = {"Red", "Green", "Blue" };
9          System.out.println("Original Array: " + Arrays.toString(colorsArray));
10
11         //length of original array
12         int orig_length = colorsArray.length;
13         //new element to be added to string array
14         String newElement = "Orange";
15         //define new array with length more than the original array
16         String[] newArray = new String[ orig_length + 1 ];
17         //add all elements of original array to new array
18         for (int i = 0; i < colorsArray.length; i++)
19         {
20             newArray[i] = colorsArray [i];
21         }
22         //add new element to the end of new array
23         newArray[newArray.length - 1] = newElement;
24         //make new array as original array and print it
25         colorsArray = newArray;
26         System.out.println("Array after adding new item: " + Arrays.toString(colorsArray));
27     }
28 }

```

Original Array: [Red, Green, Blue]

Array after adding new item: [Red, Green, Blue, Orange]

```
1 import java.util.*;
2
3 class Main
4 {
5
6     public static void main(String[] args)
7     {
8         String[] colors = {"red", "green", "blue", "white", "orange"};
9         System.out.println("Original array: " + Arrays.toString(colors));
10        Arrays.sort(colors);
11        System.out.println("Sorted array: " + Arrays.toString(colors));
12    }
13 }
```

Original array: [red, green, blue, white, orange]

Sorted array: [blue, green, orange, red, white]

```

1  import java.util.*;
2  public class Main
3  {
4      public static void main(String[] args)
5      {
6          String[] strArray = { "Book", "Pencil", "Eraser", "Color", "Pen" };
7          boolean found = false;
8          int index = 0;
9          String searchStr = "Pen";
10         for (int i = 0; i < strArray.length; i++)
11         {
12             if(searchStr.equals(strArray[i]))
13             {
14                 index = i;
15                 found = true;
16                 break;
17             }
18         }
19         if(found)
20             System.out.println(searchStr + " found at the index " + index);
21         else
22             System.out.println(searchStr + " not found in the array");
23     }
24 }
25

```

```
Pen found at the index 4
```

```

1  import java.util.*;
2
3  public class Main
4  {
5      public static void main( String[] args )
6      {
7          //string arraya declaration
8          String [] str_Array = {"10", "20", "30", "40", "50"};
9          //print the string array
10         System.out.println("Original String Array:");
11         for(String val : str_Array)
12             System.out.print(val + " ");
13
14         System.out.println("\nThe integer array obtained from string array:");
15         //declare an int array
16         int [] int_Array = new int [str_Array.length];
17         //assign string array values to int array
18         for(int i = 0; i < str_Array.length; i++)
19         {
20             int_Array[i] = Integer.parseInt(str_Array[i]);
21         }
22         //display the int array
23         System.out.println(Arrays.toString(int_Array));
24     }
25 }

```

Original String Array:

10 20 30 40 50

The integer array obtained from string array:

[10, 20, 30, 40, 50]



# Examples: Boolean array

- ▶ Each element of array is a Boolean value (true, false)

```
1 import java.util.Arrays;
2 public class BooleanArrayTest
3 {
4     public static void main(String[] args)
5     {
6         Boolean[] boolArray = new Boolean[5]; // initialize a boolean array
7         for(int i = 0; i < boolArray.length; i++)
8         {
9             System.out.println(boolArray[i]);
10        }
11
12        Arrays.fill(boolArray, Boolean.FALSE);
13        // all the values will be false
14        for(int i = 0; i < boolArray.length; i++)
15        {
16            System.out.println(boolArray[i]);
17        }
18
19        Arrays.fill(boolArray, Boolean.TRUE);
20        // all the values will be true
21        for (int i = 0; i < boolArray.length; i++)
22        {
23            System.out.println(boolArray[i]);
24        }
25    }
26 }
27
28 }
```

What is the output?



# Recommended reading

---

- ▶ Reading this week
  - Chapter 1, textbook
  - Write and test the codes in the slides of this lecture
- ▶ Next lecture
  - Insertion/Merge sort: chapter 3, textbook