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

CSC3100 Data Structures
Lecture 3: Array

Li Jiang
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
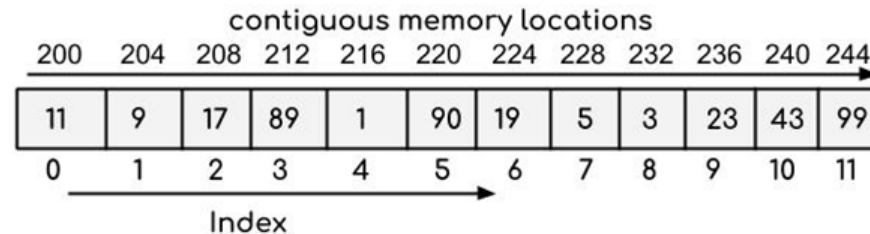- The simplest type of array is a linear array, or one-dimensional array

Array

▸ An array stores the same type of objects together, and we 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: The capacity is fixed, so if you insert an object in a place between two objects in an array, you have to move the objects first

Concepts

‣ An array is a linear data structure that hosts a collection of data elements stored at consecutive locations in a computer's memory

contiguous memory locations

200	204	208	212	216	220	224	228	232	236	240	244
11	9	17	89	1	90	19	5	3	23	43	99
0	1	2	3	4	5	6	7	8	9	10	11

Index

○ 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

‣ Key concepts
○ The individual values are called **elements**; all the elements are 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)

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

▸ 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_product;
2  for (i=0; i<n; i++){
3      i_dimension_product = retrieve(a_1,i)*retrieve(a_2,i);
4      store(a_3, i, i_dimension_product);
5  }
6  return a_3;
```

Java: array declaration

- **An array is characterized by**
  - Element type
  - Length: type[ ] identifier = new type[length];

    int[] numbers = new int[5];

- **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];
```

- Retrieve 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 i-th element array[i];
}
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",
    "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 swapElements(int[] a, int p1, int p2){
    int  tmp = a[p1];
    a[p1] = a[p2];
    a[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
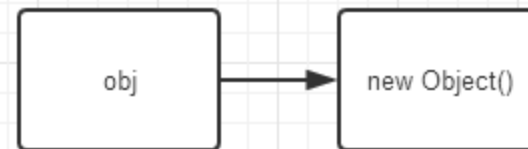
▸ What is Pass-by-Reference?
  ◦ The memory address is passed to that function, so the function gets access to the actual variable

Examples

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



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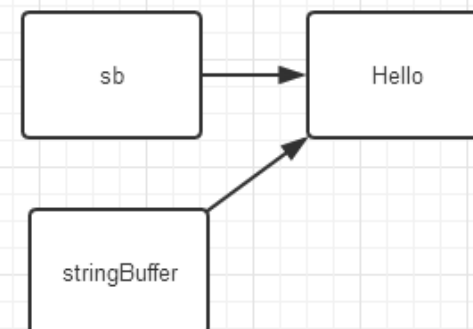
```java
public class test {
    public static void main(String[] args) {
        StringBuffer sb = new StringBuffer("Hello ");
        System.out.println("before change, sb is "+sb.toString());
        change(sb);
        System.out.println("after change, sb is "+sb.toString());
    }
    public static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

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

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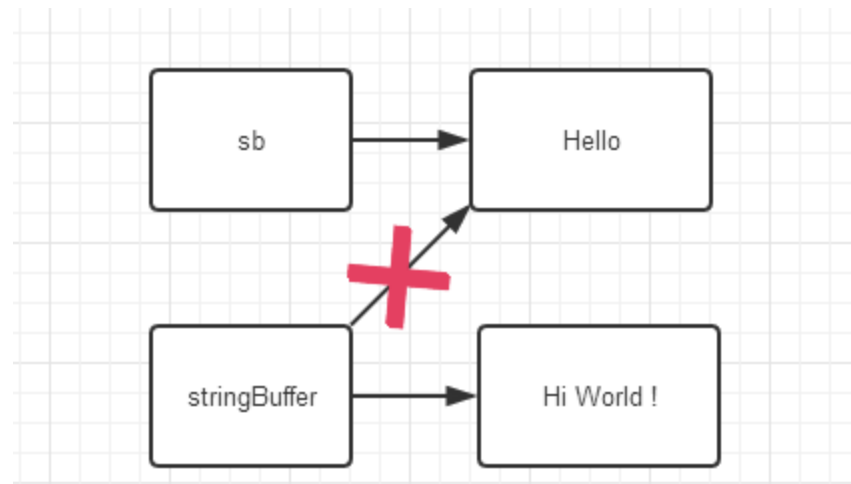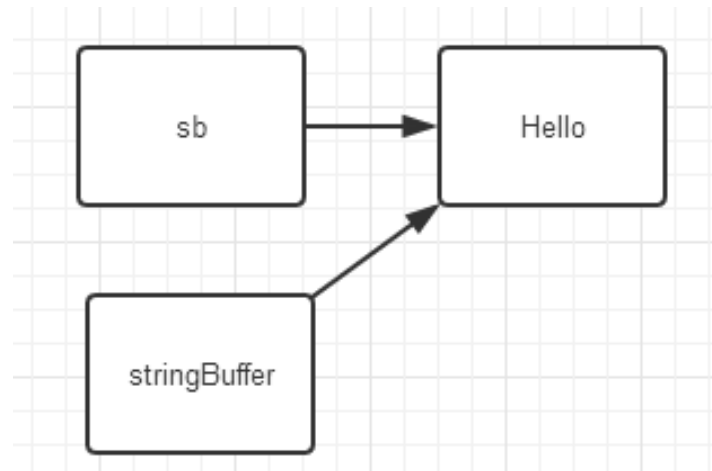
Examples

```
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
```


14

Left code:
```java
public class MyClass {
    public int x, y;

    public MyClass(int a, int b){
        x = a;
        y = b;
    }
    public static void swap(MyClass obj_a, MyClass obj_b){
        MyClass tmp = obj_a;
        obj_a = obj_b;
        obj_b = tmp;

        System.out.println("# obj_a: x=" + obj_a.x + " y=" + obj_a.y);
        System.out.println("# obj_b: x=" + obj_b.x + " y=" + obj_b.y);
        System.out.println();
    }
    public static void main(String args[]) {
        MyClass obj1 = new MyClass(1, 2);
        MyClass obj2 = new MyClass(10, 20);

        MyClass.swap(obj1, obj2);

        System.out.println("obj1: x=" + obj1.x + " y=" + obj1.y);
        System.out.println("obj2: x=" + obj2.x + " y=" + obj2.y);
    }
}
```

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

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

Right code:
```java
public class MyClass {
    public int x, y;

    public MyClass(int a, int b){
        x = a;
        y = b;
    }
    public static void swap(MyClass obj_a, MyClass obj_b){
        int tmp = obj_a.x;
        obj_a.x = obj_b.x;
        obj_b.x = tmp;

        tmp = obj_a.y;
        obj_a.y = obj_b.y;
        obj_b.y = tmp;

        System.out.println("# obj_a: x=" + obj_a.x + " y=" + obj_a.y);
        System.out.println("# obj_b: x=" + obj_b.x + " y=" + obj_b.y);
        System.out.println();
    }
    public static void main(String args[]) {
        MyClass obj1 = new MyClass(1, 2);
        MyClass obj2 = new MyClass(10, 20);

        MyClass.swap(obj1, obj2);

        System.out.println("obj1: x=" + obj1.x + " y=" + obj1.y);
        System.out.println("obj2: x=" + obj2.x + " y=" + obj2.y);
    }
}
```

Right output:
```
# 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?

15
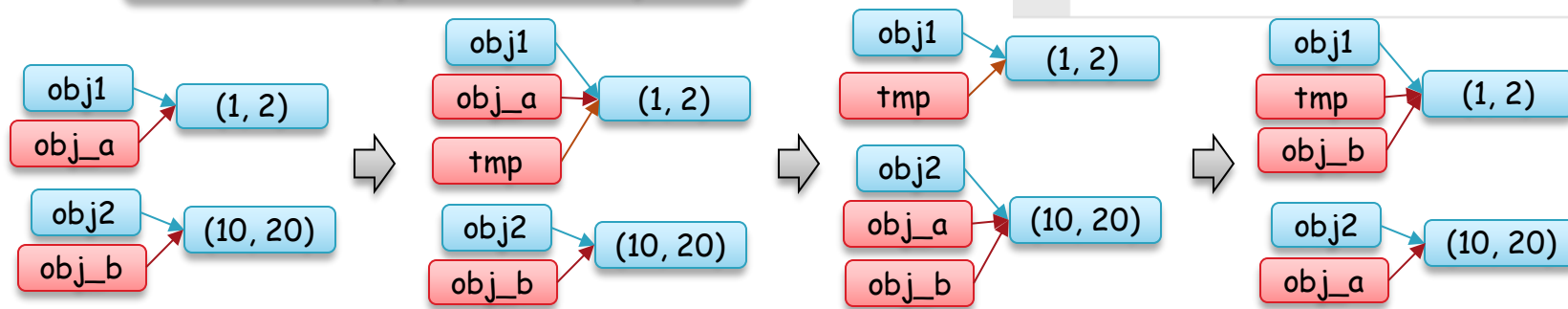
Exercise: using arrays

▸ 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 **ArrayList**s 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

https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html

More examples in Java

‣ **Number arrays**
  ◦ int, float, double, ...

‣ **String arrays**
  ◦ String array is an array holding a fixed number of strings or string values
  ◦ The arguments of "main" function is a string array
    • public static void main(String args[ ]){...}

‣ **Boolean arrays**
  ◦ Each element is either true or false
  ◦ By default, the elements are false

```java
public class Main
{
    public static void main(String[] args)
    {
        // create an array
        int[] age = {12, 4, 5};

        // loop through the array
        // using for loop
        System.out.println("Using for-each Loop:");
        for(int a : age)
        {
            System.out.println(a);
        }
    }
}
```

Output

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

```java
public class Main
{
    public static void main(String[] args)
    {

        int[] numbers = {2, -9, 0, 5, 12, -25, 22, 9, 8, 12};
        int sum = 0;
        Double average;

        // access all elements using for each loop
        // add each element in sum
        for (int number : numbers)
        {
            sum += number;
        }

        // get the total number of elements
        int arrayLength = numbers.length;

        // calculate the average
        // convert the average from int to double
        average =  ((double)sum / (double)arrayLength);

        System.out.println("Sum = " + sum);
        System.out.println("Average = " + average);
    }
}
```

Output:

```
Sum = 36
Average = 3.6
```

```java
public class MultidimensionalArray {
    public static void main(String[] args)
    {

        int[][] a =
        {
            {1, -2, 3},
            {-4, -5, 6, 9},
            {7},
        };

        for (int i = 0; i < a.length; ++i)
        {
            for(int j = 0; j < a[i].length; ++j)
            {
                System.out.println(a[i][j]);
            }
        }
    }
}
```

Output:

```
1
-2
3
-4
-5
6
9
7
```

```java
public class MultidimensionalArray
{
    public static void main(String[] args)
    {

        // create a 2d array
        int[][] a =
        {
            {1, -2, 3},
            {-4, -5, 6, 9},
            {7},
        };

        // first for...each loop access the individual array
        // inside the 2d array
        for (int[] innerArray : a)
        {
            // second for...each loop access each element inside the row
            for(int data : innerArray)
            {
                System.out.println(data);
            }
        }
    }
}
```

Output:

```
1
-2
3
-4
-5
6
9
7
```

23

```java
public class ThreeArray
{
    public static void main(String[] args)
    {

        // create a 3d array
        int[][][] test =
        {
            {
                {1, -2, 3},
                {2, 3, 4}
            },
            {
                {-4, -5, 6, 9},
                {1},
                {2, 3}
            }
        };

        // for..each loop to iterate through elements of 3d array
        for (int[][] array2D : test)
        {
            for (int[] array1D : array2D)
            {
                for(int item : array1D)
                {
                    System.out.println(item);
                }
            }
        }
    }
}
```

Output:

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

24

```java
public class Main
{
    public static void main(String[] args)
    {

        String[] myarray;          //declaration of string array without size
        String[] strArray = new String[5];  //declaration with size

        //System.out.println(myarray[0]);   //variable myarray might not have been initialized
        //display elements of second array
        System.out.print(strArray[0] + " " + strArray[1] + " " + strArray[2] + " " +
                          strArray[3] + " " + strArray[4]);
    }
}
```

```
null null null null null
```

```java
import java.util.*;

public class Main
{
    public static void main(String[] args)
    {
        //original array
        String[] colorsArray = {"Red", "Green", "Blue" };
        System.out.println("Original Array: " + Arrays.toString(colorsArray));

        //length of original array
        int orig_length = colorsArray.length;
        //new element to be added to string array
        String newElement = "Orange";
        //define new array with length more than the original array
        String[] newArray = new String[ orig_length + 1 ];
        //add all elements of original array to new array
        for (int i = 0; i < colorsArray.length; i++)
        {
            newArray[i] = colorsArray [i];
        }
        //add new element to the end of new array
        newArray[newArray.length - 1] = newElement;
        //make new array as original array and print it
        colorsArray = newArray;
        System.out.println("Array after adding new item: " + Arrays.toString(colorsArray));
    }
}
```

```
Original Array: [Red, Green, Blue]

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

```java
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]
```

```java
import java.util.*;
public class Main
{
    public static void main(String[] args)
    {
        String[] strArray = { "Book", "Pencil", "Eraser", "Color", "Pen" };
        boolean found = false;
        int index = 0;
        String searchStr = "Pen";
        for (int i = 0; i < strArray.length; i++)
        {
            if(searchStr.equals(strArray[i]))
            {
                index = i;
                found = true;
                break;
            }
        }
        if(found)
            System.out.println(searchStr + " found at the index " + index);
        else
            System.out.println(searchStr + " not found in the array");

    }
}
```

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

```java
import java.util.*;

public class Main
{
    public static void main( String[] args )
    {
        //string arrya declaration
        String [] str_Array = {"10", "20", "30", "40", "50"};
        //print the string array
        System.out.println("Original String Array:");
        for(String val : str_Array)
            System.out.print(val + " ");

        System.out.println("\nThe integer array obtained from string array:");
        //declare an int array
        int [] int_Array = new int [str_Array.length];
        //assign string array values to int array
        for(int i = 0; i < str_Array.length; i++)
        {
            int_Array[i] = Integer.parseInt(str_Array[i]);
        }
        //display the int array
        System.out.println(Arrays.toString(int_Array));
    }
}
```

```
Original String Array:

10 20 30 40 50

The integer array obtained from string array:

[10, 20, 30, 40, 50]
```

```java
import java.util.Arrays;
public class BooleanArrayTest
{
    public static void main(String[] args)
    {
        Boolean[] boolArray = new Boolean[5]; // initialize a boolean array
        for(int i = 0; i < boolArray.length; i++)
        {
            System.out.println(boolArray[i]);

        }
        Arrays.fill(boolArray, Boolean.FALSE);
        // all the values will be false
        for(int i = 0; i < boolArray.length; i++)
        {
            System.out.println(boolArray[i]);

        }
        Arrays.fill(boolArray, Boolean.TRUE);
        // all the values will be true
        for (int i = 0; i < boolArray.length; i++)
        {
            System.out.println(boolArray[i]);

        }
    }
}
```

null
null
null
null
null
false
false
false
false
false
true
true
true
true
true

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