# 14 Arrays

*Instructor*: Ke Wei（柯韋）

A319 Ext. 6452 wke@ipm.edu.mo

http://brouwer.ipm.edu.mo/COMP112/18/

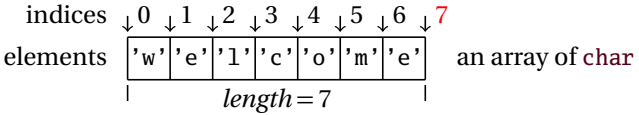
Bachelor of Science in Computing, School of Public Administration, Macao Polytechnic Institute

November 5, 2018

# Outline

# Arrays

- An array is a group of variables. These variables are the *elements* of the array.
- The elements are stored in *consecutive* memory cells, one next to another, without gaps.
- The elements are of the same type. This type is the *element type* of the array. We often say "an array of $T$", if the element type is $T$.
- An array has a fixed size, called its *length*. This length is specified when the array is created.
- An element of an array is accessed via an *index*. The index is an integer. The index must be $\geq 0$ and $<$ the length of the array.
- An element is an l-value. You can assign to an element.

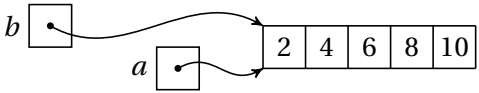


# Array Objects and Array Variables

- An array itself is an *object*.
- We can only access an array via a *reference* (pointer) to the array object.
- An array variable holds an array reference, but it cannot hold an array object.
- An array object is created by the `new` operator. The `new` operator returns a reference to the newly created object.
- An element at index $i$ of an array pointed to by an array variable $a$ is denoted by $a[i]$.
- The length of an array pointed to by variable $a$ is obtained by $a.length$.

```
int[] a = new int[5];
for ( int i = 0; i < a.length; ++i ) a[i] = 2*(i+1);
int[] b = a;
```

- We can have multiple array variables pointing to the same array object.



# Declaring and Initializing Array Variables

- A type name *T* followed by a pair of brackets `[]` results the type name *T*`[]` for the arrays of *T*, such as `int[]` for the arrays of `int` and *String*`[]` for the arrays of *String*.
- We declare an array variable just like declaring other variables, except that we use an array type name.

```
int[] a, b; char[] c; double[] d, e; // five array variables
```

- An array variable can be initialized by 1) a new array, 2) another array vairable, or 3) an *array initializer*.
- An array initializer is a comma-separated list of expressions, enclosed by braces { and }.
- A new array is created by the `new` operator, followed by the array type name with either the length specified in the brackets, or a further array initializer.

```
int[] a = new int[100], b = a;
double[] d = new double[] {1.0, 2.0, 3.0}, e = {1.0, 4.0, 9.0};
```

# Initializing Elements by a Function of Indices

- Often, the value of an element is a function of its index.
- An array of 100 odd numbers:

$$1, 3, 5, \ldots, 199$$

can be created by

```
int[] a = new int[100];
for ( int i = 0; i < a.length; ++i ) a[i] = 2*i+1;
```

- An array of 50 squares:

$$100^2, 200^2, 300^2, \ldots, 5000^2$$

can be created by

```
int[] a = new int[50];
for ( int i = 0; i < a.length; ++i ) { a[i] = (i+1)*100; a[i] *= a[i]; }
```

# Summing, Averaging and Counting Elements

- To compute the sum of all elements in an array is obvious.

```
double s = 0.0;
for ( int i = 0; i < a.length; ++i ) s += a[i];
```

- The average of all elements is $s/a.length$ when we have the sum in $s$.
- Be careful! We can have arrays of zero length!

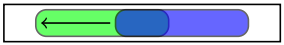
```
double[] a = new double[0]; // completely legal!
```

- Sometimes we need to count the number of elements by a condition, for example, the number of elements with a value over 50.

```
int n = 0;
for ( int i = 0; i < a.length; ++i )
    if ( a[i] >= 50.0 ) n++;
```

# Shifting Array Elements

- We may need to move a segment (at index *si* of length *n*) of an array to another segment (at index *di*) of the same array.
- The source segment and the destination segment may overlap each other.



- To prevent the moving from destroying the elements that have not yet been moved, when moving to the left, we move the left elements first; when moving to the right, we move the right elements first.

```
if ( si < di ) {
    int sj = si+n, dj = di+n;
    for ( int j = 0; j < n; ++j ) a[--dj] = a[--sj];
} else if ( si > di ) {
    int sj = si, dj = di;
    for ( int j = 0; j < n; ++j ) a[dj++] = a[sj++];
}
```

# Passing and Returning Arrays

- Array variables store references, so copying an array variable copies only the reference but not the array.

```
int[] a = {1,2,3}, b;
b = a; // b and a point to the same array.
b[1] = 100; // changing the array pointed to by b also changes the array pointed to by a.
System.out.println(a[1]); // prints 100.
```

- When we pass and return an array to and from a method, we transfers only the reference to the array.
- The method below returns a copy of an array.

```
public static int[] copyIntArray(int[] a) {
    int[] b = new int[a.length];
    for ( int i = 0; i < b.length; ++i ) b[i] = a[i];
    return b;
}
```

# Concatenating Two Arrays

We concatenate two arrays by copying the first array then copying the second array afterwards.

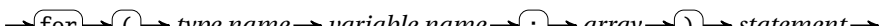
```
1  public static int[] concatIntArrays(int[] a, int[] b) {
2      int[] c = new int[a.length+b.length];
3
4      for ( int i = 0; i < a.length; ++i )
5          c[i] = a[i];
6      for ( int i = 0; i < b.length; ++i )
7          c[a.length+i] = b[i];
8
9      return c;
10 }
```

# *For-each* Loop

- Java supports a convenient `for` loop, known as a *for-each* loop, which enables you to traverse an array sequentially without using an index variable.

```
int[] a = new int[] {1,3,5,7,9,11,13,15,17,19};
for ( int u : a )
    System.out.println(u);
```

- You can read the code as "for each element *u* in *a*, do the following." The array is viewed as a collection of elements.
- Below is the syntax diagram for the for-each statement.




- You still have to use an index variable if you wish to traverse the array in a different order or change the elements in the array.

# Reading Homework

## Textbook

- Section 7.1–7.3, 7.5–7.7.
- Section 7.12.

## Internet

- Array data type (`https://en.wikipedia.org/wiki/Array_data_type`).
- Pointer (computer programming) (`https://en.wikipedia.org/wiki/Pointer_(computer_programming)`).
- Arrays (The Java™ Tutorials) (`https://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html`).