Software and Programming I

More on Arrays

Roman Kontchakov

Birkbeck, University of London
Outline

- Arrays
- Enhanced for Loop
- Common Array Algorithms
- Using Arrays with Methods
  - Sections 6.1–6.4
- slides are available at www.dcs.bbk.ac.uk/~roman/sp1
Arrays

- An array collects a sequence of values of the same type.

```java
// empty array of 5 integers
Student[] students = new Student[5];

// list of initial values
String[] data = { "I", "V", "X", "L" };
```
Array Elements

- individual elements in an array are accessed by an integer index \( i \), using the notation \( \text{data}[i] \)
- an array element can be used in expressions like any other variable
- the elements of arrays are numbered starting at \( 0 \)
- use the expression \( \text{data.length} \) to find the number of elements in an array

```java
int[] data = { 2, 3, 5, 7, 11 };
for (int i = 0; i < data.length / 2; i++)
    data[data.length - 1 - i] = data[i];
```
Arrays: References are Copied

An array variable specifies the location of an array. Copying the reference yields another reference to the same array.

```java
1 int[] scores = { 10, 9, 7, 4, 5 };  
2 int[] values = scores;  
3 scores[3] = 10;  
4 // prints 10, not 4!  
5 System.out.println(values[3]);
```

NB: Just like object references!
Enhanced For Loop

use the enhanced for loop to visit all elements of an array

```java
public static double sum(double[] data) {
    double total = 0;
    // for(int i = 0; i < data.length; i++) {
    //    // e is a COPY of the current array element
    //    double e = data[i];
    //    // total += e;
    // }
    for (double e: data)
        total += e;
    return total;
}
```
The Length of Arrays is Fixed

come up with a guess on the maximum number of elements and keep a companion variable for the current size

```java
final int LENGTH = 100; // max number of elements
// partially filled array
double[] data = new double[LENGTH];
int currentSize = 0; // the actual number of elements
data[currentSize] = 42; // insert 42
currentSize++; // increase the number of elements
```

0
```
currentSize
```
data[0]
data[1]
data[2]
...
data[LENGTH-1]

1
```
currentSize
```
data[0] = 42
data[1]
data[2]
...
data[LENGTH-1]```
Partially Filled Arrays

```java
Scanner in = new Scanner(System.in);
while (in.hasNextDouble()) // read all doubles
    if (currentSize < data.length) {
        // currentSize is the first position available
        data[currentSize] = in.nextDouble();
        // update the actual number of elements
        currentSize++;
    }

    // the actual elements are indexed from 0 to currentSize-1
for (int i = 0; i < currentSize; i++)
    System.out.println(data[i]);
```
Partially Filled Arrays: Removing an Element

removing the element at position pos

```
for (int i = pos; i < currentSize - 1; i++)
    data[i] = data[i+1];
currentSize--; // update the actual number of elements
```
Partially Filled Arrays: Inserting an Element

inserting a newElement at position pos

```java
if (currentSize < data.length) { // check space
    for (int i = currentSize; i > pos; i--)
        data[i] = data[i-1];
    data[pos] = newElement; // place into array
    currentSize++; // update the number of elements
}
```
Bubble Sort: Idea

Repeatedly step through the list to be sorted, comparing each pair of adjacent items and **swapping** them if they are in the wrong order.

The pass through the list is repeated until no swaps are needed, which indicates that the list is **sorted**.
Bubble Sort: Example

pass 1: ( 5 1 4 2 8 ) → ( 1 5 4 2 8 ), swap
       ( 1 5 4 2 8 ) → ( 1 4 5 2 8 ), swap
       ( 1 4 5 2 8 ) → ( 1 4 2 5 8 ), swap
       ( 1 4 2 5 8 ) → ( 1 4 2 5 8 )

pass 2: ( 1 4 2 5 8 ) → ( 1 4 2 5 8 )
       ( 1 4 2 5 8 ) → ( 1 2 4 5 8 ), swap
       ( 1 2 4 5 8 ) → ( 1 2 4 5 8 )
       ( 1 2 4 5 8 ) → ( 1 2 4 5 8 )

The array is already sorted,
but the algorithm does not know if it is completed
It needs one whole pass without any swap to know it is sorted

NB: does it need to go until the very end on every pass?
NB: how many steps does the algorithm require?

best sorting algorithms require $O(n \log n)$ steps
```java
boolean swapped = false;
do {
    swapped = false;
    // start from 1, not 0!
    for (int i = 1; i < data.length; i++) {
        if (data[i-1] > data[i]) {
            double t = data[i-1];
            data[i-1] = data[i];
            data[i] = t;
            swapped = true;
        }
    }
} while (swapped);
```
for (int r = 8; r >= 1; r--) {
    for (char c = 'a'; c <= 'h'; c++)
        System.out.print(" "+c+r+" ");
    System.out.println();
}
Take Home Messages

- an array is a sequence of values of the same type
- an array index in an array data must be
  \[ \geq 0 \text{ and } < \text{data.length} \]
- an array variable specifies the location of an array; copying the reference yields a second reference to the same array
- arrays can occur as method parameters and return values (passing “by reference”)
- with a partially filled array, keep a companion variable for the current size