# Arrays (review)

CSE 2011
Winter 2011



## Arrays

**A common programming task is to keep track of a group of related objects!**

**Array – sequence of indexed components with the following properties:**

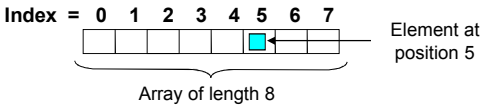
- **array size is fixed at the time of array's construction**

```
int[] numbers = new numbers [10];
```

- **array elements are placed contiguously in memory** ⇒ address of any element can be calculated directly as its offset from the beginning of the array

- consequently, **array components can be efficiently inspected or updated in O(1) time, using their indices**

```
randomNumber = numbers[5];
numbers[2] = 100;
```



## Arrays (cont.)

### Arrays in Java – Properties

(1) For an array of length n, the index bounds are 0 to (n-1).

(2) Java arrays are homogeneous - all array components must be of the same (object or primitive) type.

- but, an array of an object type can contain objects of any respective subtype

(3) An array is itself an object.

- it is allocated dynamically by means of "new", it is automatically deallocated when no longer referred to

(4) when an array is first created, all values are initialized with

- 0 – for an array of int[] or double[] type
- false – for a boolean[] array
- null – for an array of objects

**Example 1** [ common error – uninitialized arrays ]

```
int[] numbers;
numbers[2] = 100;
```

3

## Arrays (cont.)

### Arrays in Java – Properties (cont.)

(5) The length of any array (object) can be accessed through its instance variable 'length'.

- the cells of an array A are numbered: 0, 1, .., (A.length-1) !!!

(6) ArrayIndexOutOfBoundsException - thrown at an attempt to index into array A using a number larger than (A.length-1).

- helps Java avoid 'buffer overflow attacks'

**Example 2** [ declaring, defining and determining the size of an array ]

```
int[] A={12, 24, 37, 53, 67}
for (int i=0; i < A.length; i++) {
   … }
```

Array is defined at the time of its 'declaration'.

4

## Arrays (cont.)

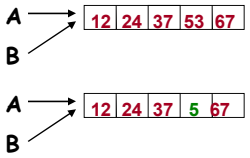
### Arrays in Java – Properties (cont.)

(7) **Since an array is an object, the name of the array is actually a reference (pointer) to the place in memory where the array is stored.**

- **reference to an object holds the address of the actual object**

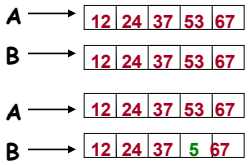
**Example 3** [ arrays as objects ]

```
int[] A={12, 24, 37, 53, 67};
int[] B=A;
B[3]=5;
```




**Example 4** [ cloning an array ]

```
int[] A={12, 24, 37, 53, 67};
int[] B=A.clone();
B[3]=5;
```






## Arrays (cont.)

### Arrays in Java: a few useful methods (java.util.Arrays)

**equals(A,B)** – **returns true if A and B have an equal number of elements and every corresponding pair of elements in the two arrays are equal**

**fill(A,x)** – **store element x into every cell of array A**

**sort(A)** – **sort the array A in the natural ordering of its elements**

**binarySearch([int] A, int key)** – **search the specified array of ints for the specified value using the binary search algorithm**

## Arrays (cont.)

**Example** [ What does get printed out ?! ]

```
...
        int[] A={12, 24, 37, 53, 67};

        int[] B=A.clone();

        if (A==B)  System.out.println(" Superman ");

        if (A.equals(B))  System.out.println(" Snow White ");
...
```

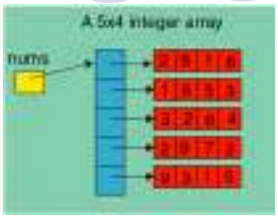
7

## Arrays (cont.)

**Example 4** [ 2D array in Java = array of arrays !!! ]

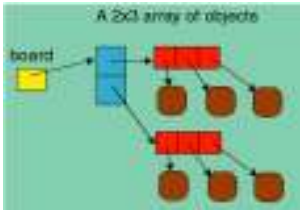
```
int[][] nums = new int[5][4];
```

```
int[][] nums;
nums = new int[5][];
for (int i=0; i<5; i++) {
        nums[i] = new int[4]; }
```




**Example 5** [ 2D array of objects in Java = an array of arrays of references !!! ]

```
Square[][] board = new Square[2][3];
```




8

http://www.willamette.edu/~gorr/classes/cs231/lectures/notes.htm

## Arrays (cont.)

### Arrays in General – Major Limitations

(1) **static data structure** – size must be fixed at the time the program creates the array – once set, array size cannot be changed

- if: *number of entered items > declared array size* ⇒ out of memory
- fix 1: use *array size > number of expected items* ⇒ waste of memory
- fix 2: increase array size to fit the number of items ⇒ extra time

**Example 5** [ time complexity of "growing" an array ]

```
if (numberOfItems > numbers.length) {
    int[] newNumbers = new int[2*numbers.length];
    System.arraycopy(numbers, 0, newNumbers, 0, numbers.length);
    numbers = newNumbers;
}
```

Number of elements to be copied.

Starting position in source array.

Starting position in destination array.

(2) **insertion / deletion in an array is time consuming** – all the elements following the inserted element must be shifted appropriately

9

## Next topic ...

- Linked lists (3.2, 3.3)