



# Arrays

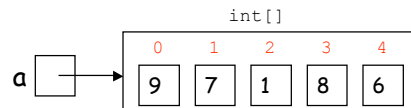
Wellesley College CS230  
Lecture 02  
Thursday, February 1  
Handout #9

Reading: Downey: Chapter 10  
Problem Set: Assignment #1 due Tuesday, February 13

02 - 1



# Array Operations



Java arrays are homogeneous: all slots must contain the same kind of element.

E.g., an array with type `int[]` is an array of integers.

Java arrays are fixed length: `a.length`  $\rightarrow$  5

Get the contents of a slot via `a[index]` (“a sub index”):

`a[0]`  $\rightarrow$  9, `a[4]`  $\rightarrow$  6, `a[5]`  $\rightarrow$  **ArrayIndexOutOfBoundsException**

Set the contents of a slot via `a[index] = expression;`

`a[0] = 2; a[1] = a[2] + a[3];`

`a[i]++` means `a[i] = a[i] + 1;` and `a[i]--` means `a[i] = a[i] - 1;`

Create an integer array with `num` slots via `new int[num];`

`int [] a = new int[5];` // All slots contain 0 by default

`a[0]=9; a[1]=7; ... ; a[4]=6;`

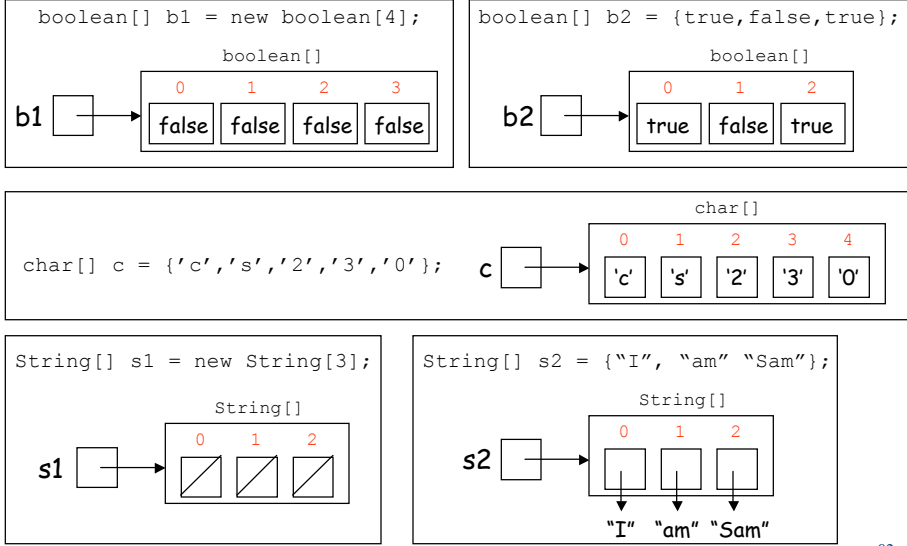
Shorthands: `int [] a = new int[] {9,7,1,8,6};`

`int [] a = {9,7,1,8,6};`

02 - 2



## Arrays Elements can be of Any Type



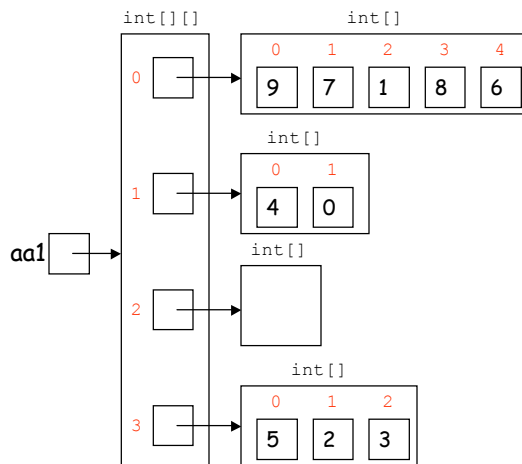
02 - 3



## Arrays of Arrays

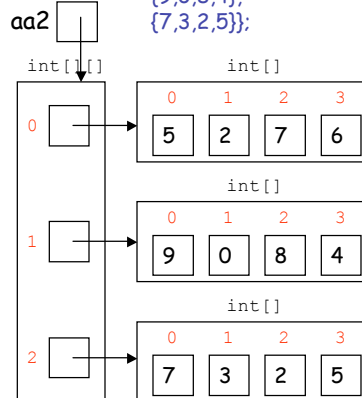
### An irregular 2-dimensional array

```
int[][] aa1 = {{9,7,1,8,6}, {4,0}, {}, {5,2,3}};
```



### A regular 2D array

```
int[][] aa2 = {{5,2,7,6}, {9,0,8,4}, {7,3,2,5}};
```

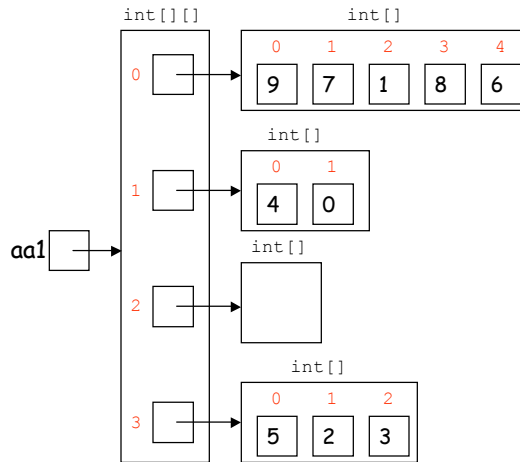


OR `int[][] aa2 = new int[3][4];  
aa2[0][0] = 5; aa2[0][1] = 2; ...02-4`



## Arrays Diagrams Illustrate Sharing

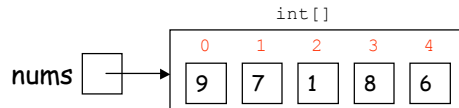
```
aa1[3] = aa1[0];  
aa1[0][2] = aa1[0][3] + aa1[0][4];  
aa1[3][0] = aa1[3][1] + aa1[3][2];  
System.out.println(aa1[3][0]);
```



02 - 5



## A Simple Array Iteration: Summation



Want `IntArray.sum(nums)` → 31

```
// Assume this is in the class IntArray  
public static int sum (int [] a) {
```

Iteration Table

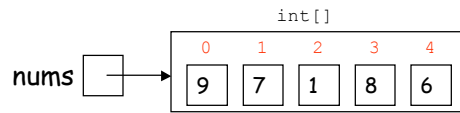
i	ans
0	0
1	9
2	16
3	17
4	25
5	31

```
}
```

02 - 6



## What does mystery() do?



IntArray.mystery(nums) → ???

```
// Assume this is in the class IntArray
public static String mystery (int [] a) {
    if (a.length == 0) // The empty array is a special case
        return "{}";
    else {
        String result = "{" + a[0];
        for (int i = 1; i < a.length; i++) { // Why is the first index 1, not 0?
            result = result + "," + a[i];
        }
        return result + "}";
    }
}
```

02 - 7



## Testing sum(): Approach 1

Suppose we want `java IntArray sum` to give the following output:

```
sum({9,7,1,8,6}) = 31
sum({4,0}) = 4
sum({}) = 0
sum({5,2,3}) = 10
```

```
// Assume this is in the class IntArray
public static void testSum (int [] a) {
    System.out.println("sum(" + toString(a) + ") = " + sum(a));
}

public static void main (String [] args) {
    if (args.length == 1) {
        if (args[0].equals("sum")) {
            testSum(new int[] {9,7,1,8,6}); testSum(new int[] {4,0});
            testSum(new int[] {}); testSum(new int[] {5,2,3});
        } else ... // Testing of other methods goes here
    } else System.out.println("usage: IntArray <name>");
}
```

02 - 8



## Testing sum(): Approach 2

---

```
public static void main (String [] args) {
    int[][] arrays = {{9,7,1,8,6}, {4,0}, {}, {5,2,3}};
    if (args.length == 1) {
        if (args[0].equals("sum")) {

        } else ... // Testing of other methods goes here
    } else System.out.println("usage: IntArray <name>");
}
```

02 - 9



## Testing sum(): Approach 3

---

Suppose we want `java IntArray sum 8 17 1 22 6` to give the following output:

```
sum({8,17,1,22,6}) = 54
```

```
// Assume this is in the class IntArray
// Returns an integer array based on strings in strs[1..(strs.length-1)]
public static int[] makeIntArray (String[] strs) {

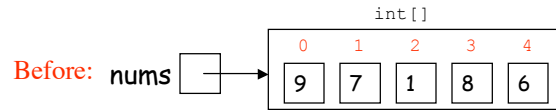
}

public static void main (String [] args) {
    if (args.length >= 1) { // Use >= to allow extra arguments
        if (args[0].equals("sum")) {
            testSum(makeIntArray(args));
        } else ... // Testing of other methods goes here
    } else System.out.println("usage: IntArray <name> <int1> ... <intr> ");
}
```

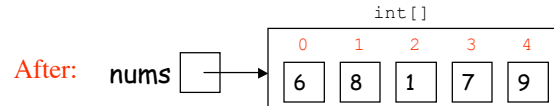
02 - 10



## Array Reversal



`IntArray.reverse(nums);`



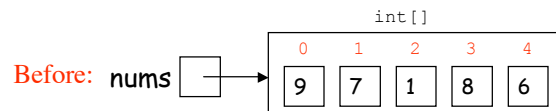
// Assume this is in the class `IntArray`  
`public static void reverse (int [] a) {`

`}`

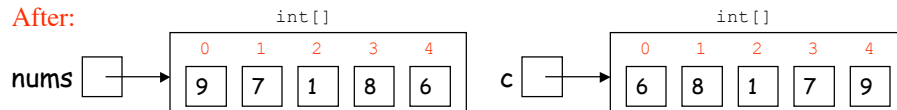
02 - 11



## Copying Array Reversal



`int[] c = IntArray.copyReverse(nums);`



// Assume this is in the class `IntArray`  
`public static int[] copyReverse (int [] a) {`

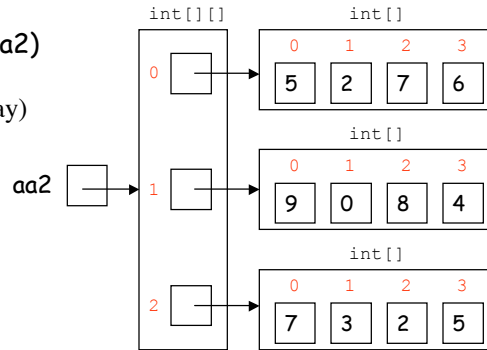
`}`

02 - 12



## Column Summation

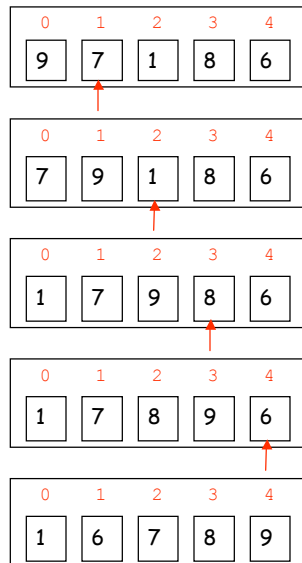
Want `IntArray.sumColumns(aa2)`  
→ {21, 5, 17, 15}  
(assume input is a regular 2D array)



02 - 13



## Insertion Sort



02 - 14