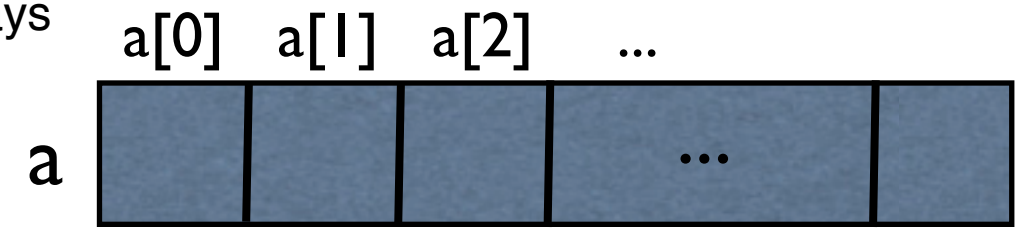
Computer Science 210: Data Structures

Arrays and Vectors

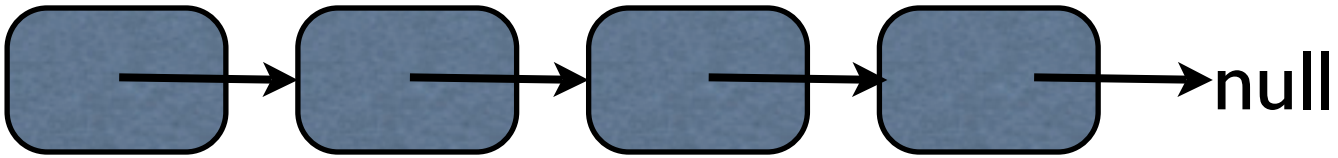
Collections of data

- The most common thing you want to do when writing algorithms/code is handle a bunch of data
  - get(i): give me the i-th element

- The fundamental structures
  - Arrays



  - Linked lists



Arrays

```
int[] a;
//declare a to be an array; a is null

a = new int[10];
//create a: allocate space to hold 10 integers and assign
//a reference to this memory to a
```

- Accessing an array:

```
a[0], a[1]...a[9]
a.length
```

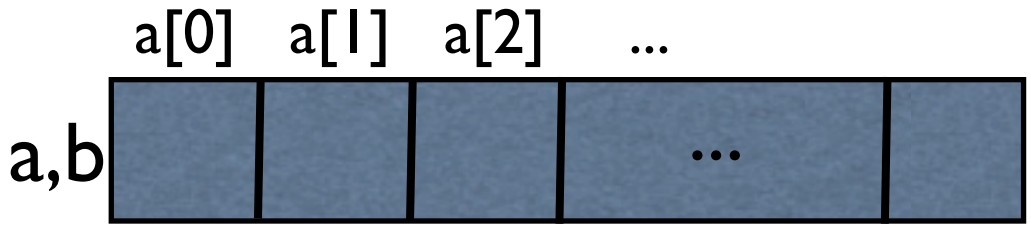
- In general, you can have arrays of objects

```
Object[] a;
a = new Object[10];
//allocate space to hold 10 Object and assign
//a reference to this memory to a
```

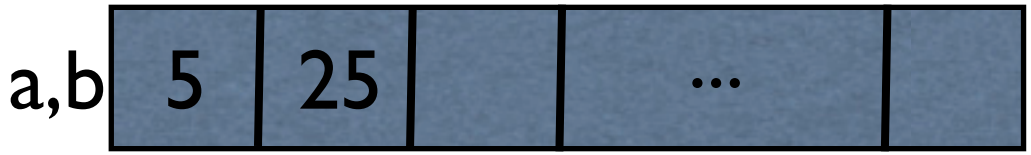
Assigning Arrays

```
int []a, []b;

a = new int[10];
b = a;
```




```
b[1] = 25;
a[0] = 3;
b[0] = 5;
```



Arrays in Java

- Java provides a number of built-in methods for performing common tasks on arrays
- Java.util.Arrays
  - equals (a, b)
    - performs an element-by-element comparison of a and b and returns true if all elements are equal
  - binarySearch (a, val)
    - searches for val in a using binary search, and returns the result
  - toString(a)
    - converts each element to a String, concatenates them and returns the result
  - sort(a)
    - sorts a in < order
- Note: all static methods
  - called without instantiating an object

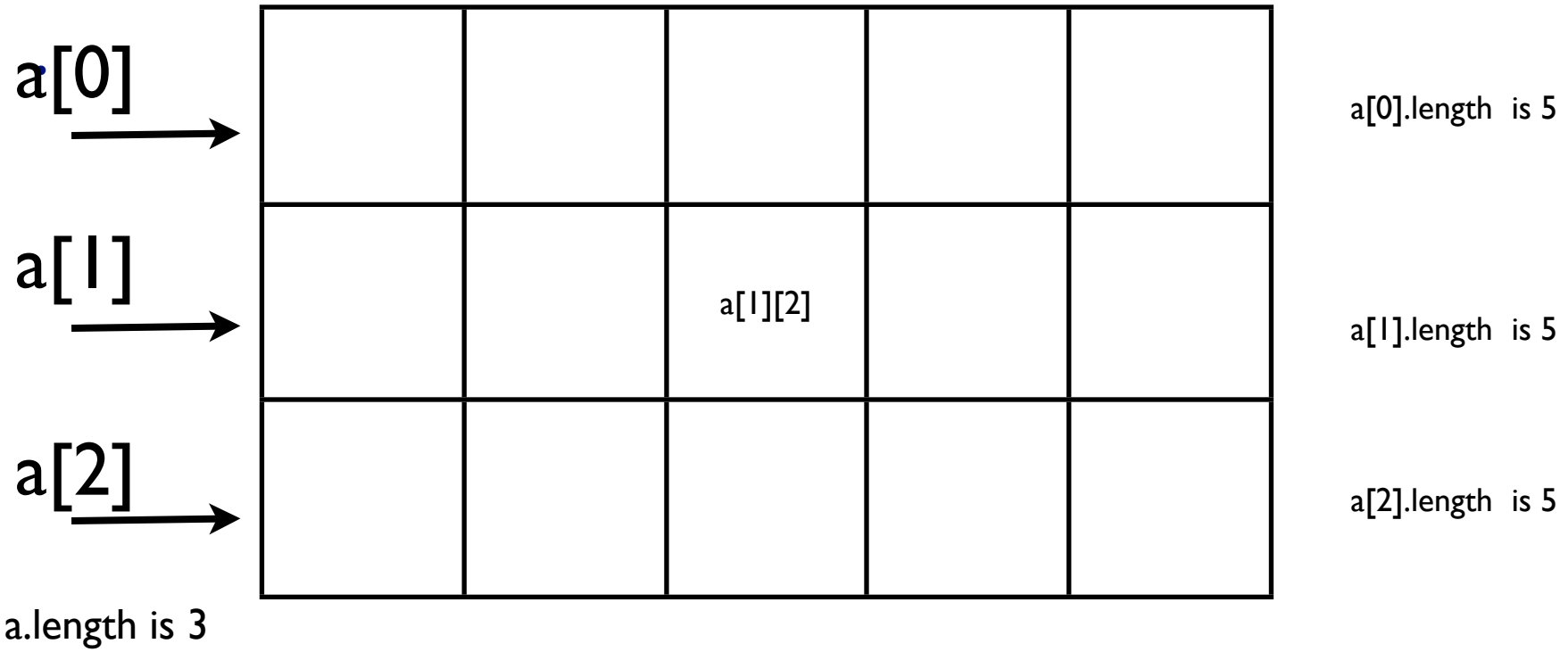
Arrays in Java: Example

- Java.util.Arrays
  - equals (a, b);
  - binarySearch (a, val)
  - toString(a)
  - sort(a)

```
import java.util.Arrays;
...
int[] a = new int[100];
for (int i=0; i< a.length; i++) a[i] = i;
System.out.print("the arrays is: " + Arrays.toString(a));
Arrays.sort(a);
System.out.print("The sorted arrays is: " +
Arrays.toString(a));
```

2D-arrays

```
int [][] a;
a = new int [3][5];
//a is an array of 3 rows ; each row is an array of 5 columns
```



3D-arrays

```
int [][][] a;
a = new int [3][4][5];
//a is an array of 3 elements; each element of a is a 2D-array [4][5]
```

- a.length is 3
- a[0].length is 4
- a[0][0].length is 5

Vectors and ArrayLists in Java

- "Smart" array classes provided by Java
  - Java.util.ArrayList
  - Java.util.Vector

- Practically identical

- Provide support for "smart" arrays
  - allow variable size of array
  - support useful methods
    - get(i)
    - set(i,e)
    - add(i,e)
    - remove(i)
    - add(e)
    - size()
    - isEmpty()

Vectors and ArrayLists in Java

- classes: ArrayList, Vector
  - get(i)
  - set(i,e)
  - add(i,e)
  - remove(i)
  - add(e)
  - size()
  - isEmpty()

- Exercise: think about implementation
  - Notation
    - N is the maximum capacity of the array
    - n is the current size

Performance

- get(i) : O(1)
- set(i,e): O(1)
- add(i,e): O(n)
- remove(i): O(n)
- size(): O(1)
- isEmpty(): O(1)
- add(e): O(1) unless overflow

  O(n) if overflow

- ArrayLists and Vectors do NOT have a fixed maximum size (like arrays); they grow the array
  - Whenever an element needs to be added and the array is full, the array is re-allocated with a larger size (default: double size)

- let's say N is the current max capacity of the array A
- allocate B[] of size 2N
- copy A[i] into B[i] for all i
- [free the space of A: note: this does not happen in Java, the garbage collector will find out that the space of A is not in use anymore and will free it]
- A = B
- add e to A as usual

Exercises

- see VectorExample.java
  - in the Examples folder on class website

- Write code to insert 100 random integers in an array, one at a time, at the beginning.