Arrays (cont); **`ArrayList`**

- Review: array syntax and arrays of objects
- Using arrays for random-access applications
- Partially filled array
- Java **`ArrayList`** class
- Autoboxing
- [Time permitting] Example: **`Names`** class (over next few lectures)

Announcements

- New TA: Ben Mathews
- My office hours CANCELLED tomorrow only
  (Ben will have 3:30 – 4:30 as a replacement)
- PA1 due Wednesday night
- this week's lab: programming with ArrayList
- Later this week / weekend:
  - PA2 will be published
  - Sample MT 1 exams will be available
- MT 1: Tue. 2/23 9:30am-10:50am
  - online exam
  - we'll have a rehearsal exam beforehand
  - alternate time will be provided for time-zone issues (TBA)

Review Arrays

```
int[] intArr;
```
array reference only

```
intArr = new int[100];
```
create array object

valid indices are?

```
int val= intArr[10];
```
access an array elmt

its value is?

```
intArr[10] = 59;
```
change value of array element

```
int val2 = intArr [100];
```
what does this do?

complete a loop to add 10 to all the elements in the array:

```
for (int i = 0;                                    ; i++) {


}
String[] words = new String[10];
int len = words[3].length();
```
what value does ***len*** *have?*

Review: applications where we use random access

- Ex: count how many people got each score (histogram)

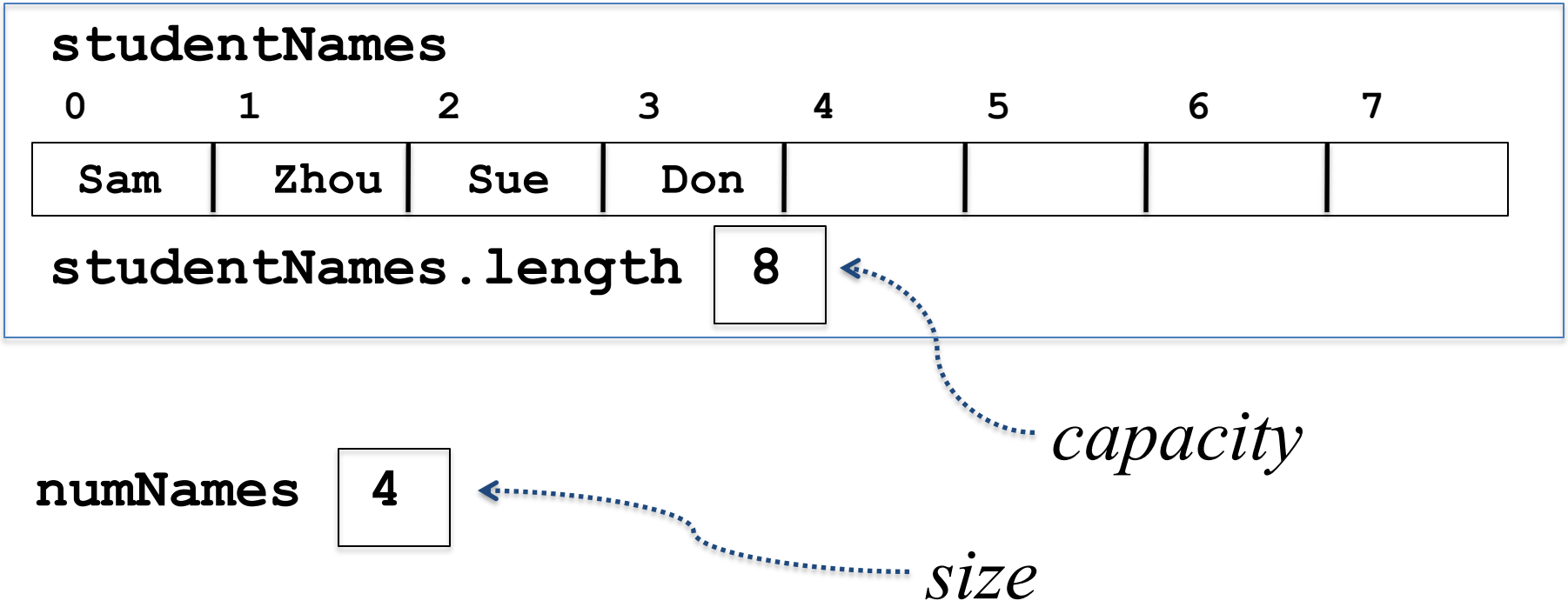
- Use random-access
- Array size known ahead of time and doesn't change
- Uses the whole array
- All elements have to be initialized ahead of time

Partially filled array

- Ex: store data about all students in the class
- Characteristics…
  - Don't know how many students there will be ahead of time
  - Students may add or drop
  - Uses mostly sequential access

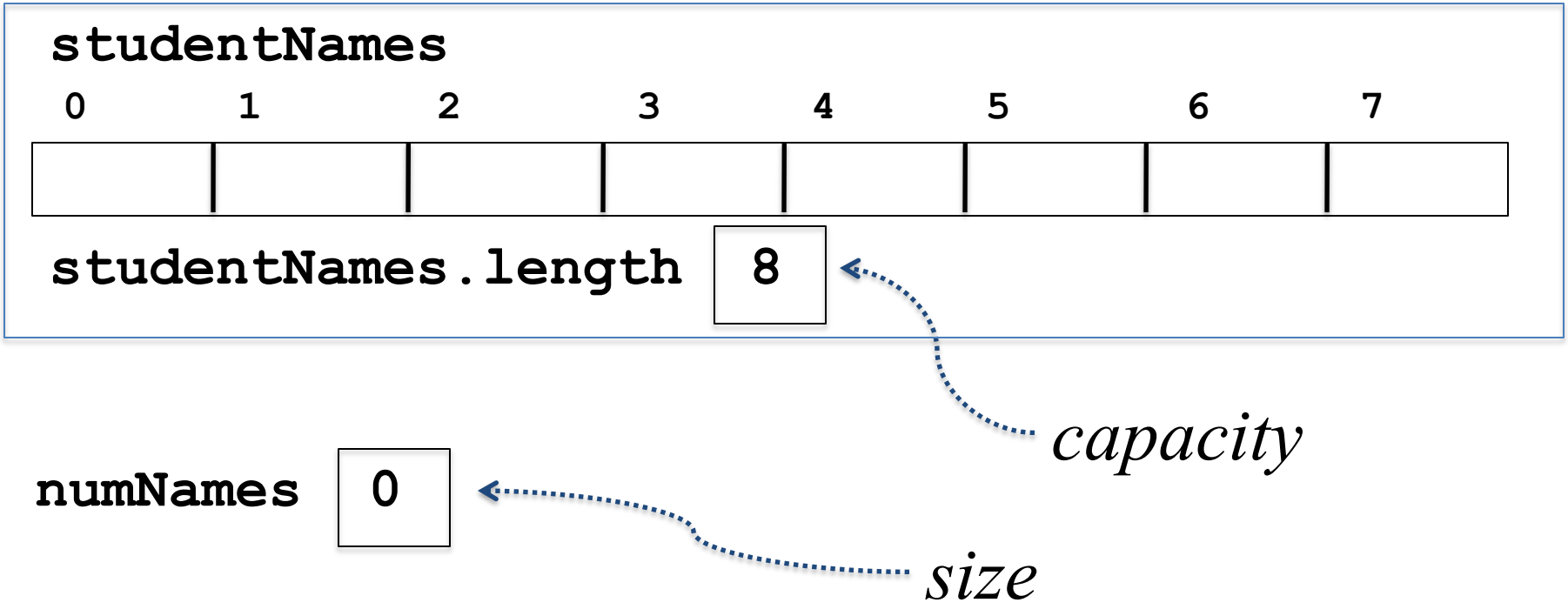
- Use a partially filled array

Ex: partially filled array of student names



code to add a new student to the end:

Empty partially filled array of student names



example initialization:

```
String[] studentNames = new String[8];
int numNames = 0;
```

POLL: partially filled array

- pollEv.com/cbono

Asynchronous participation: Link to Partially-filled Array poll

Difficulties of partially filled array

- have to guess necessary capacity ahead of time
- have to keep two variables in sync: **`numNames`** and **`studentNames`**
- What if we run out of space?
  - have to allocate a bigger array
  - copy all the elements from smaller array to bigger array
  - **`Arrays.copyOf`** (discussed in section 7.3.9 can help with this)
- Common use of arrays, so …

`ArrayList` class

- Hides the code to take care of messy details of partially-filled array:
- Keeps track of how full array is:

  `arrList.size()`
- Makes array bigger as necessary:

  `arrList.add("Zhou");`

  adds Zhou to the *end* of the partially-filled array
- Accessing individual elements by index still uses random access (fast): **`get`**, **`set`** methods

`ArrayList` basic syntax

```
ArrayList<String> names =
                    new ArrayList<String>();
```
create empty arraylist

```
int len = names.size();        // 0
```
```
names.add("Joe");
```
adds a new name to end of list

```
int len = names.size();        // 1
```

```
String name = names.get(0);
```
like ***`names[0]`***

```
names.set(0, "Suzy");
```
like ***`names[0]=`***

```
String name2 = names.get(1);
```
run-time error

Example: traversing an `ArrayList`

Review of syntax:

```
ArrayList<String> names =
                  new ArrayList<String>();
```
create empty arraylist

```
int len = names.size();       // 0
```
`names.add("Joe");` *adds a new name to end of list*

```
int len = names.size();       // 1
```
`String name = names.get(0);` *like* ***`names[0]`***

`names.set(0, "Suzy");` *like* ***`names[0]=`***

```
// turns all the names into nicknames, using this
// pattern, shown by example:
// "Sam" turns into "The Sam-inator"
public static void nickNamer(
                          ArrayList<String> names)
```

Traversing an **`ArrayList`**

```
// turns all the names into nicknames, using this
// pattern shown by example:
//       "Sam" turns in into "The Sam-inator"
public static void nickNamer(
                          ArrayList<String> names) {
```

`ArrayList` of ints

- With generics, must use a ***class*** as type parameter:

```
ArrayList<Integer> nums =
                    new ArrayList<Integer>();
```

create empty arraylist

```
nums.add(3);
nums.add(17);
nums.add(2);
int n = nums.get(1);
```

- Uses auto-boxing…

Wrapper classes / Autoboxing

```
int i = 10;
Integer iObj = new Integer(10);
Integer iObj2 = 12;      // ok
int j = iObj + i + 5;          // ok
```