Names example

- Example: `Names` class
- practice with
  - coding array algorithms
  - implementing classes
  - and using good development techniques
- incremental development
- for `lookup, remove, insert:`
  - design test cases first
  - implement code
    - code refactoring
  - test code
Announcements

• Sample midterm exams have been published (link to Sample Exams page on left side of web page).

• PA2 available sometime on Friday
Names class

- Stores a list of unique names in alphabetical order.
- Allows look-up, insert, and removal of elements in the list.
- Uses partially-filled array representation

- Names.java has a partial implementation
- MinNamesTester.java is a program to test that subset.
# Names representation

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Don</td>
<td>Joe</td>
<td>Sam</td>
<td>Sue</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **namesArr**: An array of names.
- **namesArr.length**: The length of the array, which is 8.
- **numNames**: The number of names, which is 4.
- **capacity**: The maximum number of elements the array can hold.
- **size**: The number of elements currently in the array.
Lookup test cases

• Returns true iff `target` is present in names

<table>
<thead>
<tr>
<th>namesArr</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

Test cases

numNames 5
• Returns true iff `target` is present in names
Remove test cases

Removes target from names object, and returns true.
If target wasn't present in names, returns false and no change made to names.

Test cases

namesArr

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Anne</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Bob</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Carol</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Don</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Ed</td>
<td></td>
</tr>
</tbody>
</table>

numNames 5
public static void testRemove() {
    Names names = new Names();
    names.loadNames();
    System.out.println("Attempt remove: Scotty");
    boolean removed = names.remove("Scotty");
    if (!removed) {
        System.out.println("Scotty was not present");
    }
    System.out.println(
        "Names in list [exp: Anne Bob Carol Don Ed]: ");
    names.printNames();
    System.out.println(
        "Number of names in list [exp: 5]: 
        " + names.numNames());
}
Implementing remove: outline

Removes \texttt{target} from names object, and returns \texttt{true}. If \texttt{target} wasn't present in names, returns \texttt{false} and no change made to names.

\begin{verbatim}
public boolean remove(String target) {

    namesArr

    0  Anne
    1  Bob
    2  Carol
    3  Don
    4  Ed

    numNames 5

\end{verbatim}

Minimize amount of code

• Reuse lookup loop?
• It returns boolean
• Refactor!
**New helper function**

```java
/**
 * lookupLoc returns index of target in namesArr
 * or NOT_FOUND if it is not present
 */

private int lookupLoc(String target)
```
Refactored `lookup` that uses `lookupLoc`

```java
public boolean lookup(String target)
```
Implementing remove

Removes `target` from names object, and returns `true`. If `target` wasn't present in names, returns `false` and no change made to names.

```java
public boolean remove(String target) {

    namesArr
    0     Anne
    1     Bob
    2     Carol
    3     Don
    4     Ed

    numNames 5
```
Insert test cases

Inserts newName into alphabetical names list. Returns false and no change is made to names if newName is already present.

Test cases

namesArr

<table>
<thead>
<tr>
<th></th>
<th>Anne</th>
<th>Bob</th>
<th>Carol</th>
<th>Don</th>
<th>Ed</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

numNames 5
Insert code notes

Inserts `newName` into alphabetical names list.
Returns `false` and no change is made to names if `newName` is already present.

```
namesArr

0  Anne
1  Bob
2  Carol
3  Don
4  Ed
```