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 will be published soon (link to Sample Exams page on left side of web page).
• PA2 will be available soon
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>namesArr</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>Don</td>
</tr>
</tbody>
</table>

namesArr.length = 8

numNames = 4
Lookup test cases

- Returns true iff \texttt{target} is present in names

<table>
<thead>
<tr>
<th>namesArr</th>
<th>Test cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>\textbf{description}</td>
</tr>
<tr>
<td>Anne</td>
<td>present</td>
</tr>
<tr>
<td>Bob</td>
<td>not present</td>
</tr>
<tr>
<td>Carol</td>
<td></td>
</tr>
<tr>
<td>Don</td>
<td></td>
</tr>
<tr>
<td>Ed</td>
<td></td>
</tr>
</tbody>
</table>

numNames 5

Names ex [Bono]
Lookup code notes

- Returns true iff \texttt{target} is present in names

\begin{verbatim}
namesArr
\end{verbatim}

\begin{verbatim}
0  Anne
1  Bob
2  Carol
3  Don
4  Ed
\end{verbatim}

\begin{verbatim}
numNames 5
\end{verbatim}

Names ex \{Bono\}
Remove test cases

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}
Test cases

namesArr

\begin{tabular}{|l|}
\hline
0  Anne \\
1  Bob \\
2  Carol \\
3  Don \\
4  Ed \\
\hline
\end{tabular}

numNames 5
\end{verbatim}
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 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) {
    Names ex
    Anne
    Bob
    Carol
    Don
    Ed
    namesArr
    numNames 5
```
Minimize amount of code

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

/**
    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

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) {
    // Code implementation...
}
```
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

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

Names ex [Bono]
Insert code notes

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

```
namesArr

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