Comp 110
Variable-Sized Collections: Array Implementations

Instructor: Prasun Dewan

Prerequisite

- Interfaces
- Arrays

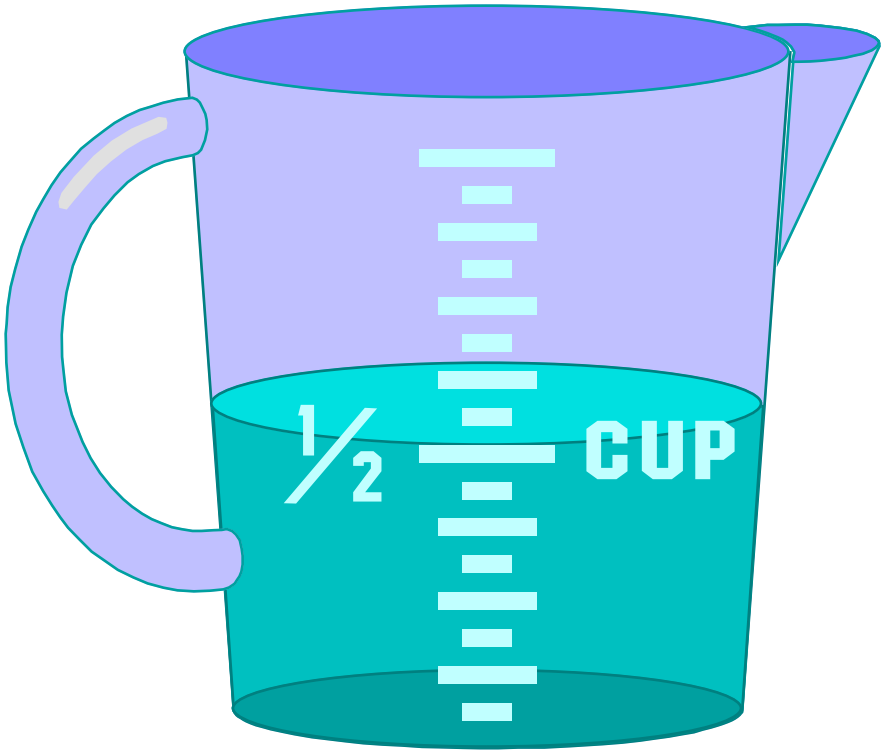
Topics

- Variable-sized collections
- Encapsulated arrays
  - History
  - Database

Example

```
<terminated> AHistoryDriver [Java Application]
James Dean
p
******************
James Dean
******************
Joe Doe
Jane Smith
p
******************
James Dean
Joe Doe
Jane Smith
******************
q
```

Variable-Size Collection



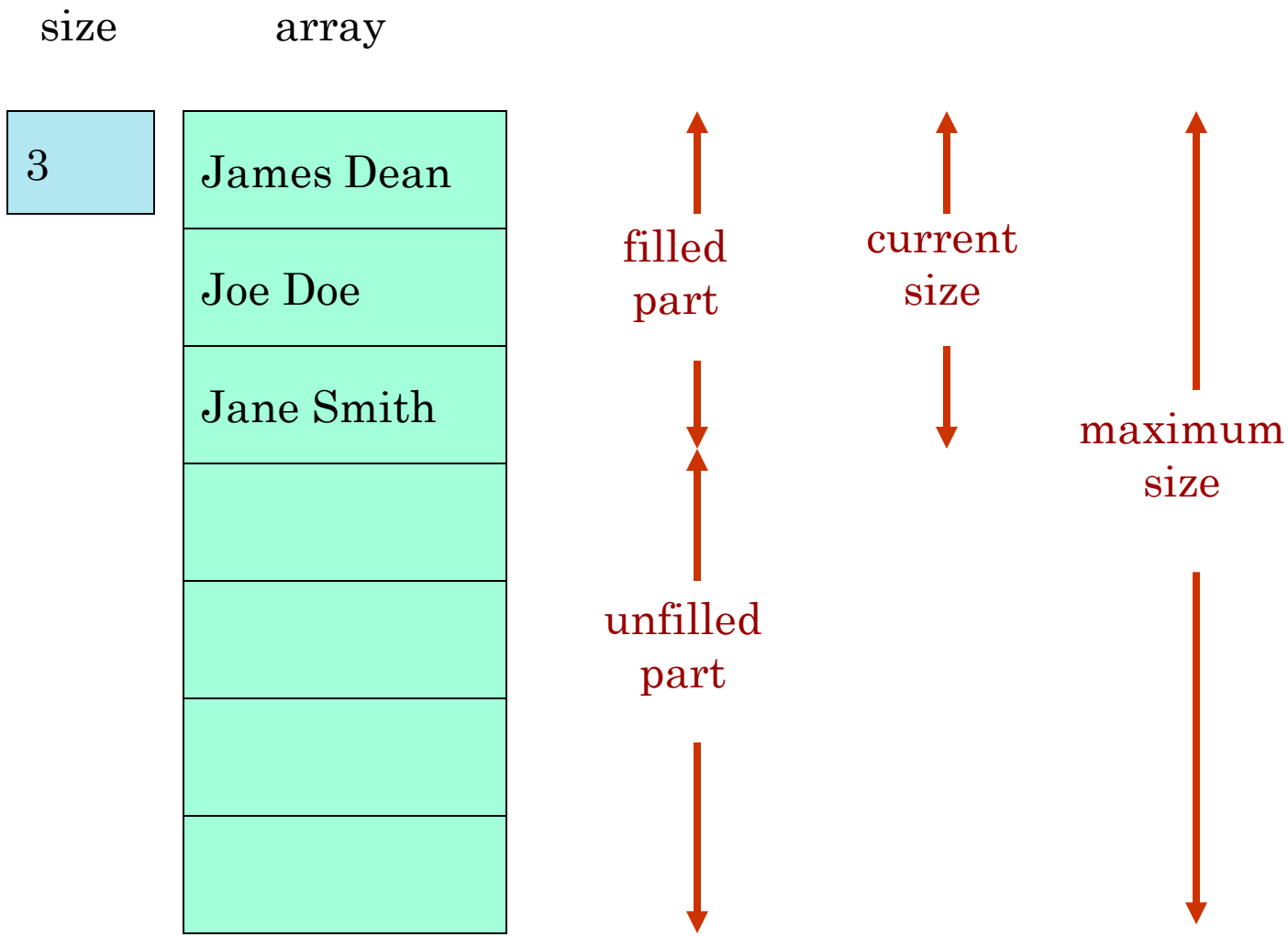
unfilled part

filled part

current size

maximum size

Variable-Size Collection



Using a History

```
public class HistoryMain{
        final static int HISTORY_MAX_SIZE = 50;
        String[] history = new String[HISTORY_MAX_SIZE];
        int historySize = 0;
        …
        static void printHistory() {
          System.out.println("********************");
          for (int index = 0; index < historySize; index++)
            System.out.println(history[index]);
          System.out.println("********************");
        }
        public static void main(String[] args) {
          …
        }

}
```

Variable-Size Collection

```
public class <ClassNeedingVariableSizeCollection> {
        …
          final static int A_MAX_SIZE = 50;
          String[] a = new String[A_MAX_SIZE];
          int aSize = 0;
          …
          //process a
          for (int index = 0; index < aSize; index++)
              System.out.println(a[index]);
          …
          final int B_MAX_SIZE = 50;
          String[] b = new String[B_MAX_SIZE];
          int bSize = 0;
          …
           //process b
          …
}
```

Monolithic Programs Using Variable Sized collection

- Variable-sized collection takes programmer effort.
  - Max size constant.
  - Current size.
  - Checking of max size.
  - Manipulation of size.
  - Deleting, replacing, searching takes more effort.
- Main program declares and uses (array-based implementation of) variable-sized collection
- Cannot reuse variable-size collection implementation.
- It is ok if program uses array as fixed-sized collection as no programmer effort required to implement it.

Special Type

```
public class <ClassNeedingVariableSizeCollection> {
    ...
    AVariableSizeCollection a = new AVariableSizeCollection();
    ...
    for (int index = 0; index < a.size; index++)
        System.out.println(a.contents[index]);
    …
    a.contents[a.size] = System.console().readLine();
    AVariableSizeCollection b = new AVariableSizeCo
    ...
}
```

Size Not Updated

Each collection must implement code for adding to history

```
public class AVariableSizeCollection {
        public static final int MAX_SIZE = 50;
        public String[] contents = new String [MAX_SIZE];
        public int size = 0;
}
```

No Encapsulation

Special Type (Review)

```
public class <ClassNeedingVariableSizeCollection> {
    ...
    AVariableSizeCollection a = new AVariableSizeCollection();
    ...
    for (int index = 0; index < a.size; index++)
        System.out.println(a.contents[index]);
    …
    a.contents[a.size] = System.console().readLine();
    AVariableSizeCollection b = new AVariableSizeCo
    ...
}
```

Size Not Updated

Each collection must implement code for adding to history

```
public class AVariableSizeCollection {
        public static final int MAX_SIZE = 50;
        public String[] contents = new String [MAX_SIZE];
        public int size = 0;
}
```

No Encapsulation

Operations?

```
<terminated> AHistoryDriver [Java Application]
James Dean
p
********************
James Dean
********************
Joe Doe
Jane Smith
p
********************
James Dean
Joe Doe
Jane Smith
********************
```

Object operations often correspond to user-interface commands

User commands are add string and print collection

```
public class AVariableSizeCollection {
        public static final int MAX_SIZE = 50;
        public String[] contents = new String [MAX_SIZE];
        public int size = 0;
}
```

Supporting Encapsulation

```
public interface StringHistory    {
        public static final int MAX_SIZE = 50;
        public void addElement(String element);
        public void print();}
```

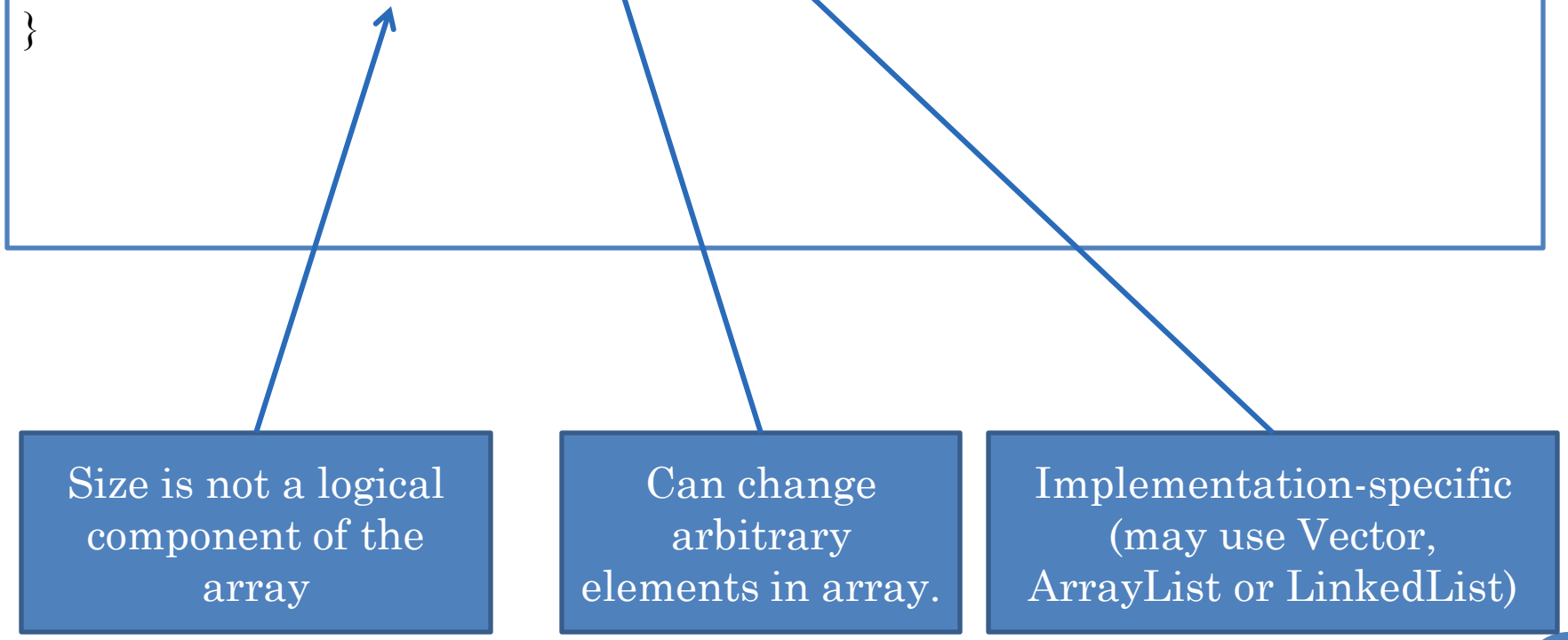
```
System.out.println("********************");
 for (int index = 0; index < historySize; index++)
    System.out.println(history[index]);
 System.out.println("********************");
```

Specific to object using history

Implementation-specific

Supporting Encapsulation

```
public interface StringHistory    {
        public void addElement(String element);
        public String[] getStringArray();
        public int getSize();
}
```



History

```
public interface  StringHistory  {
        public void addElement(String element);
        public int size();
        public String elementAt(int index);
}
```

Implementing A History

```
public class AStringHistory implements StringHistory {
        public final int MAX_SIZE = 50;
        String[] contents = new String[MAX_SIZE];
        int size = 0;
        public int size() { return size;}
        public String elementAt (int index) {  return contents[index]; }
        boolean isFull() { return size == MAX_SIZE; }
        public void addElement(String element) {
            if (isFull())
                    System.out.println("Adding item to a full history");
            else {
                    contents[size] = element;
                    size++;
            }
    }
}
```

Using a History

```
public static void main(String[] args) {
        StringHistory names = new AStringHistory();
        while (true) {
                String input = System.console().readLine();
                if (input.length() > 0)
                 if (input.charAt(0) == 'q') break;
                 else if (input.charAt(0) == 'p' )
                        print(names);
                  else
                        names.addElement(input);
        }
}
```

Printing a History

```
static void print(StringHistory strings) {
        System.out.println("********************");
        for ( int elementNum = 0; elementNum < strings.size(); elementNum++)
            System.out.println(strings.elementAt(elementNum));
        System.out.println("********************");
}
```

History

```
<terminated> AHistoryDriver [Java Application]
James Dean
p
******************
James Dean
******************
Joe Doe
Jane Smith
p
******************
James Dean
Joe Doe
Jane Smith
******************
q
```

DATABASE

```
AStringDatabase [Java Application] C:\Program F
James Dean
Joe Doe
Jane Smith
p
********************
James Dean
Joe Doe
Jane Smith
**********
m Joe Doe
true
m Jane Doe
false
d Joe Doe
p
**********
James Dean
Jane Smith
********************
c
p
********************
********************
```

```
public interface  StringHistory  {
        public void addElement(String
        element);
        public int size();
        public String elementAt(int index);
}
```

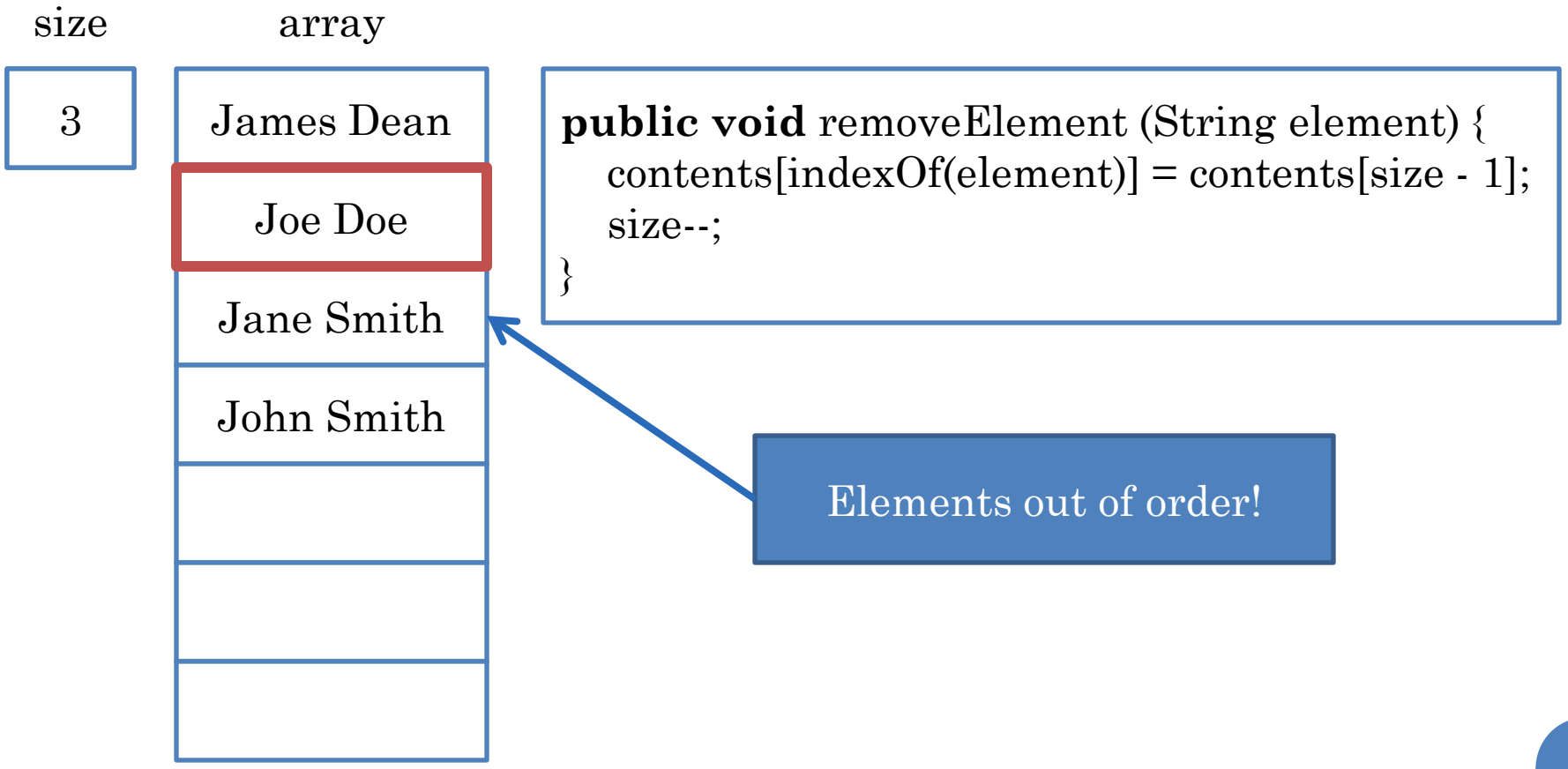
Database

```
public interface  StringDatabase  {
        //from history
        public int size();
        public void addElement(String element);
        public String elementAt(int index);

        //additional methods
         public void removeElement(String element);
         public boolean member(String element);
         public void clear();
}
```

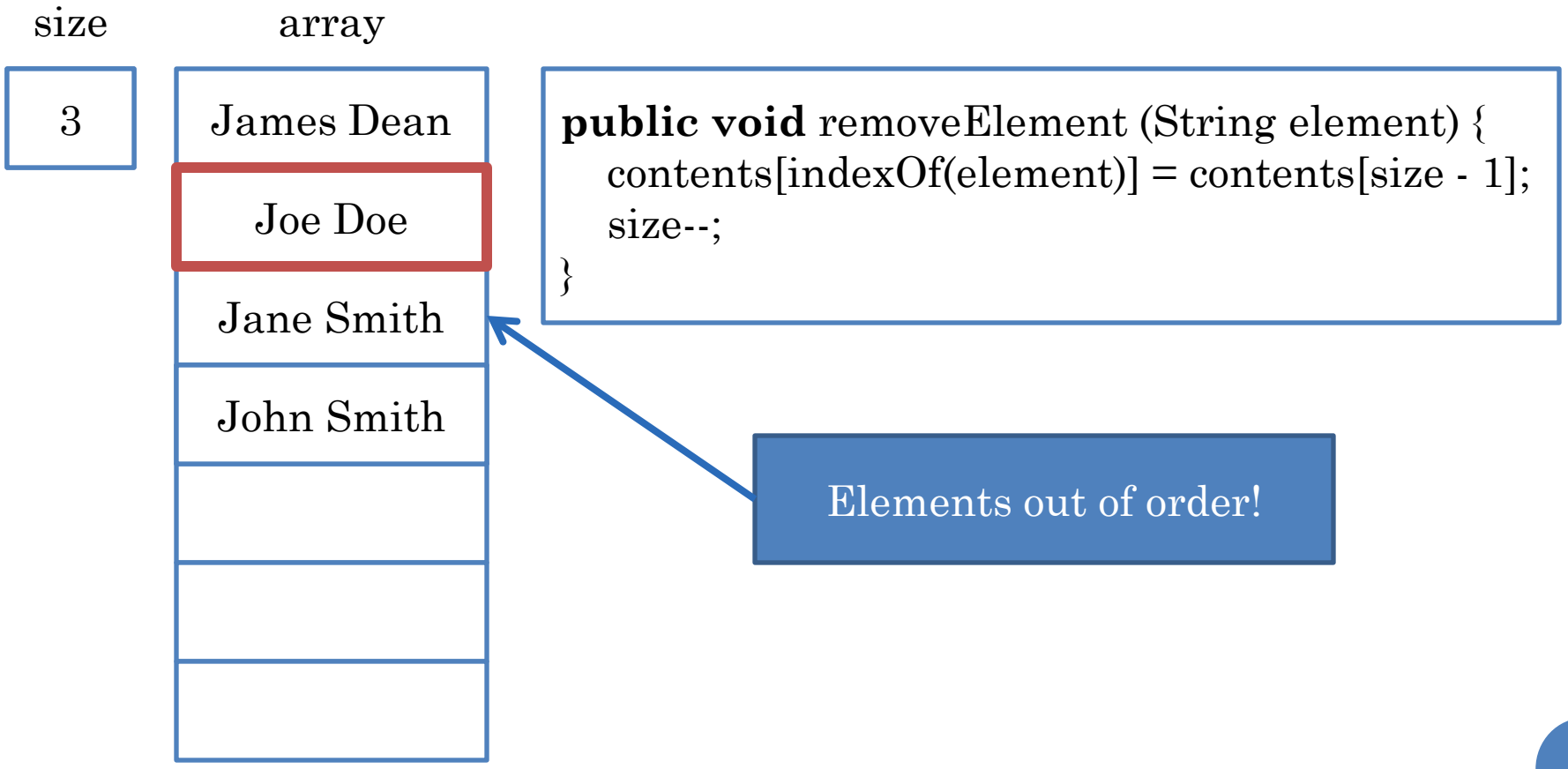
removeElement(String element)

removeElement("Joe Doe");



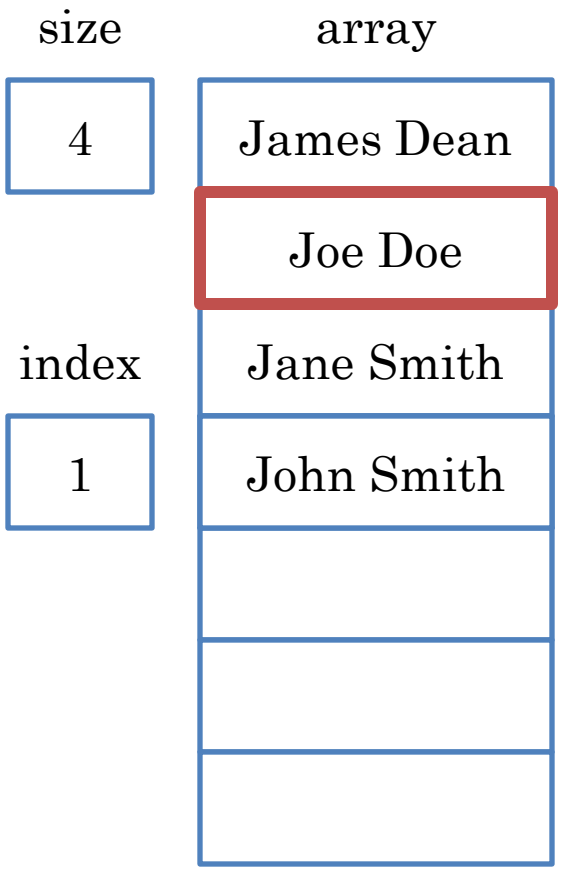
removeElement(String element)

removeElement("Joe Doe");



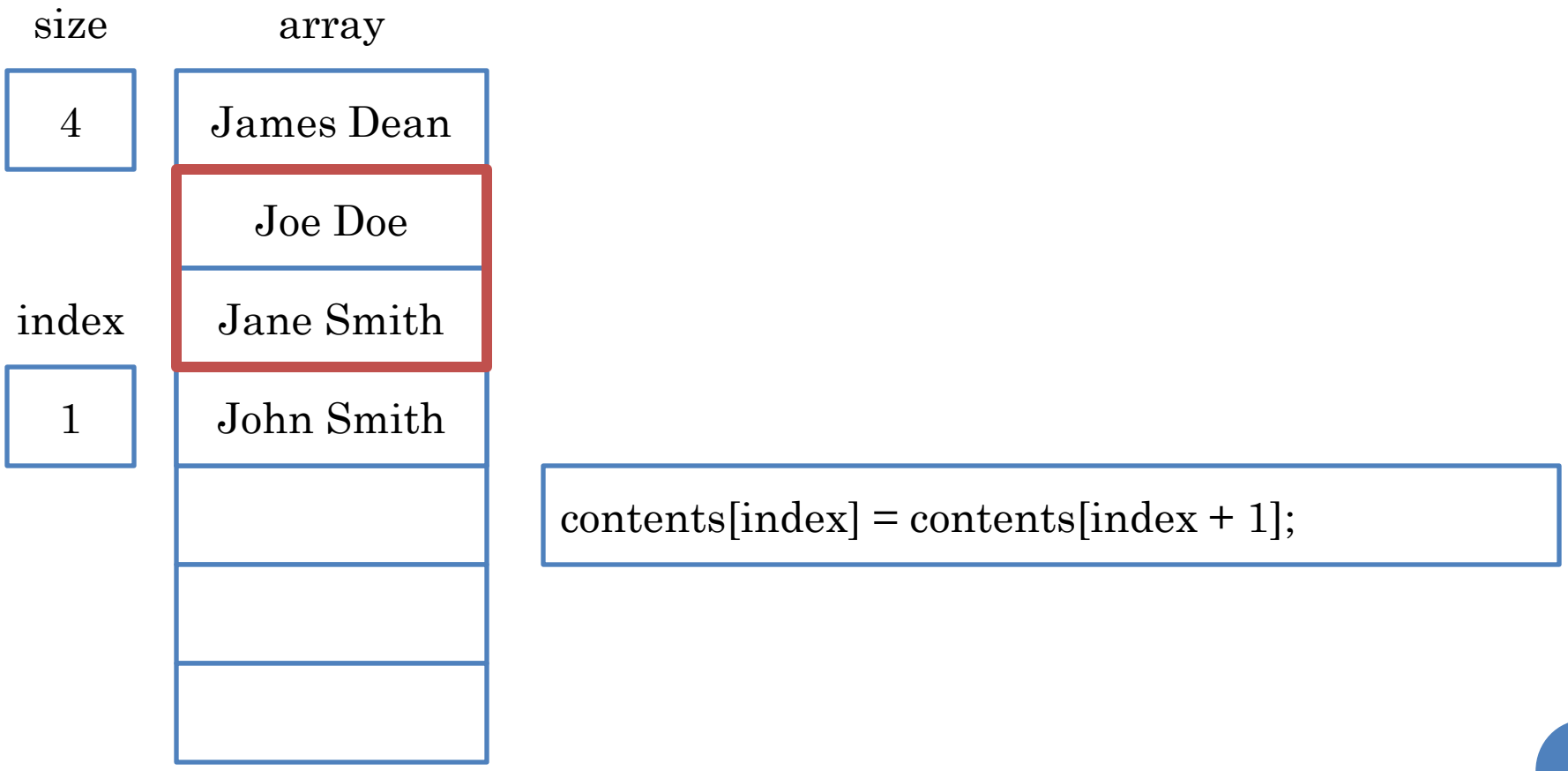
removeElement(String element)

removeElement("Joe Doe");



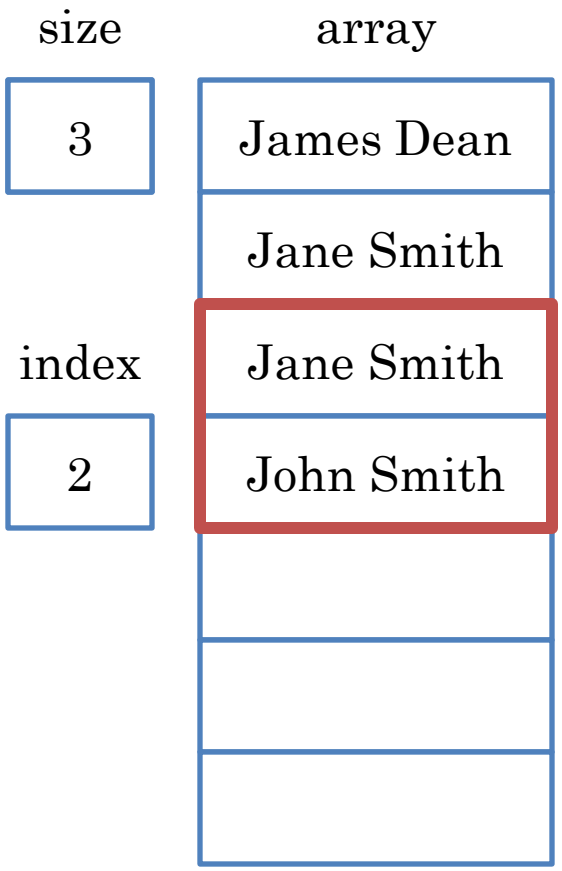
Multi-Element Window

removeElement("Joe Doe");



Multi-Element Window

removeElement("Joe Doe");



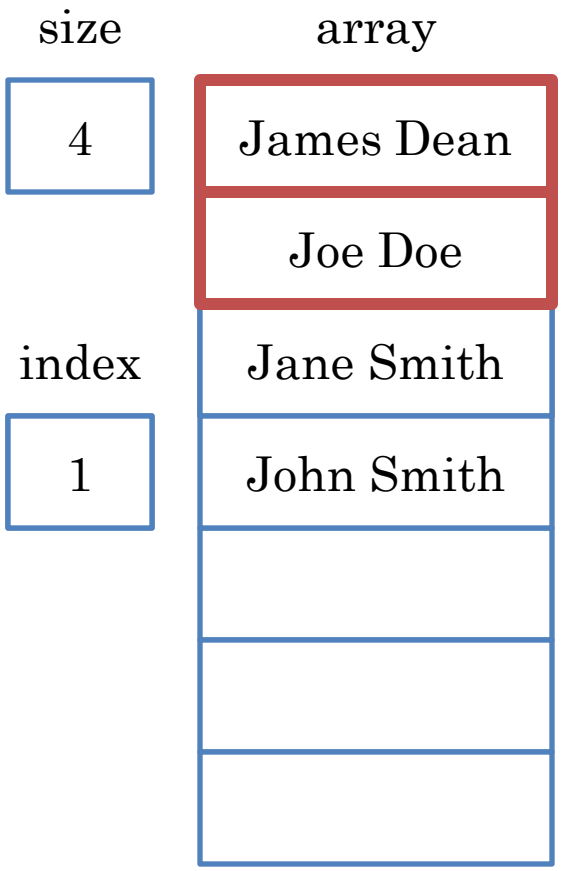

```
public void removeElement(String element) {
    removeElement(indexOf(element));
}
```

```
void removeElement(int startIndex) {
    shiftUp(startIndex);
    size--;
}
```

```
void shiftUp(int startIndex) {
    for (int index = startIndex; index + 1 < size;
                                                index++)
        contents[index] = contents[index + 1];
}
```

indexOf(String element)

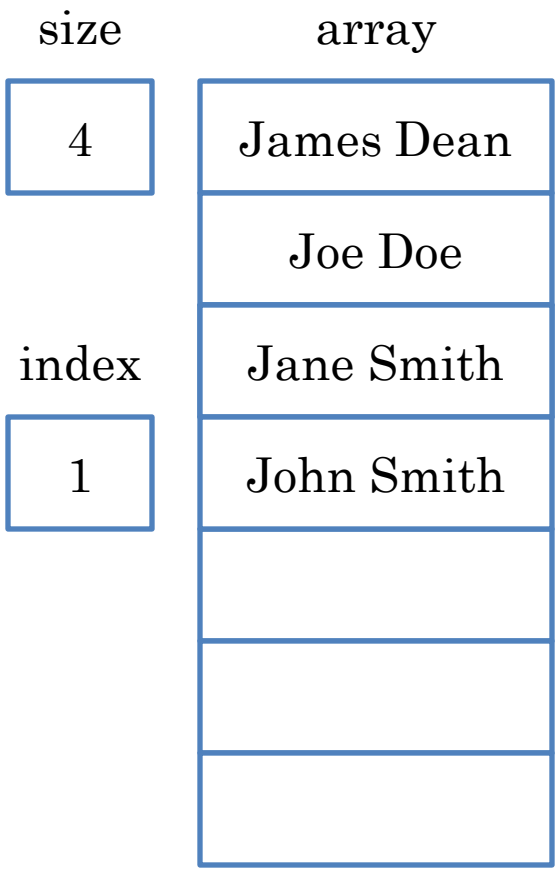
indexOf("Joe Doe");




```
public int indexOf (String element) {
    int index;
    for (index = 0;
            index < size &&
                !element.equals(contents[index]);
            index++);
    return index;
}
```

PUBLIC BOOLEAN MEMBER(STRING ELEMENT)

member("Joe Doe");

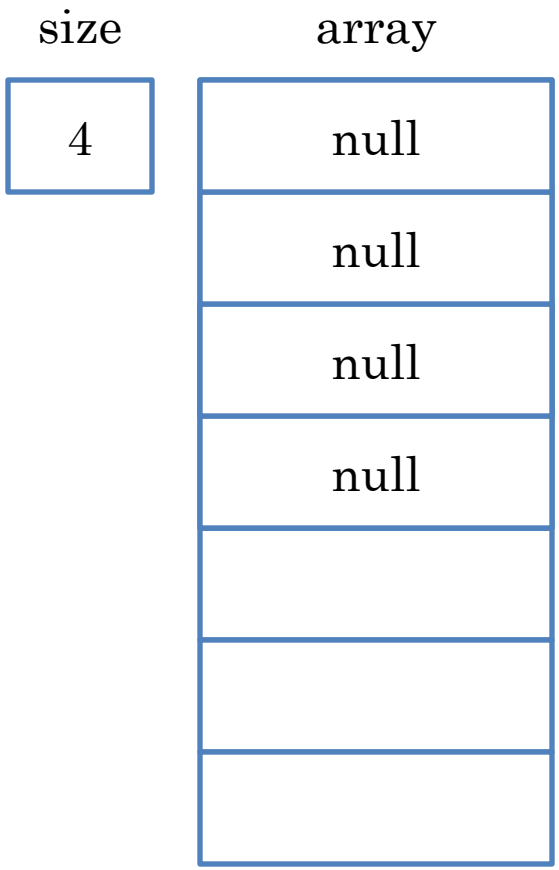



```
public int indexOf (String element) {
    int index;
    for (index = 0;
            index < size &&
                !element.equals(contents[index];
            index++);
    return index;
}
```

```
public boolean member (String element) {
    return indexOf (element) < size;
}
```

PUBLIC VOID CLEAR()

clear();

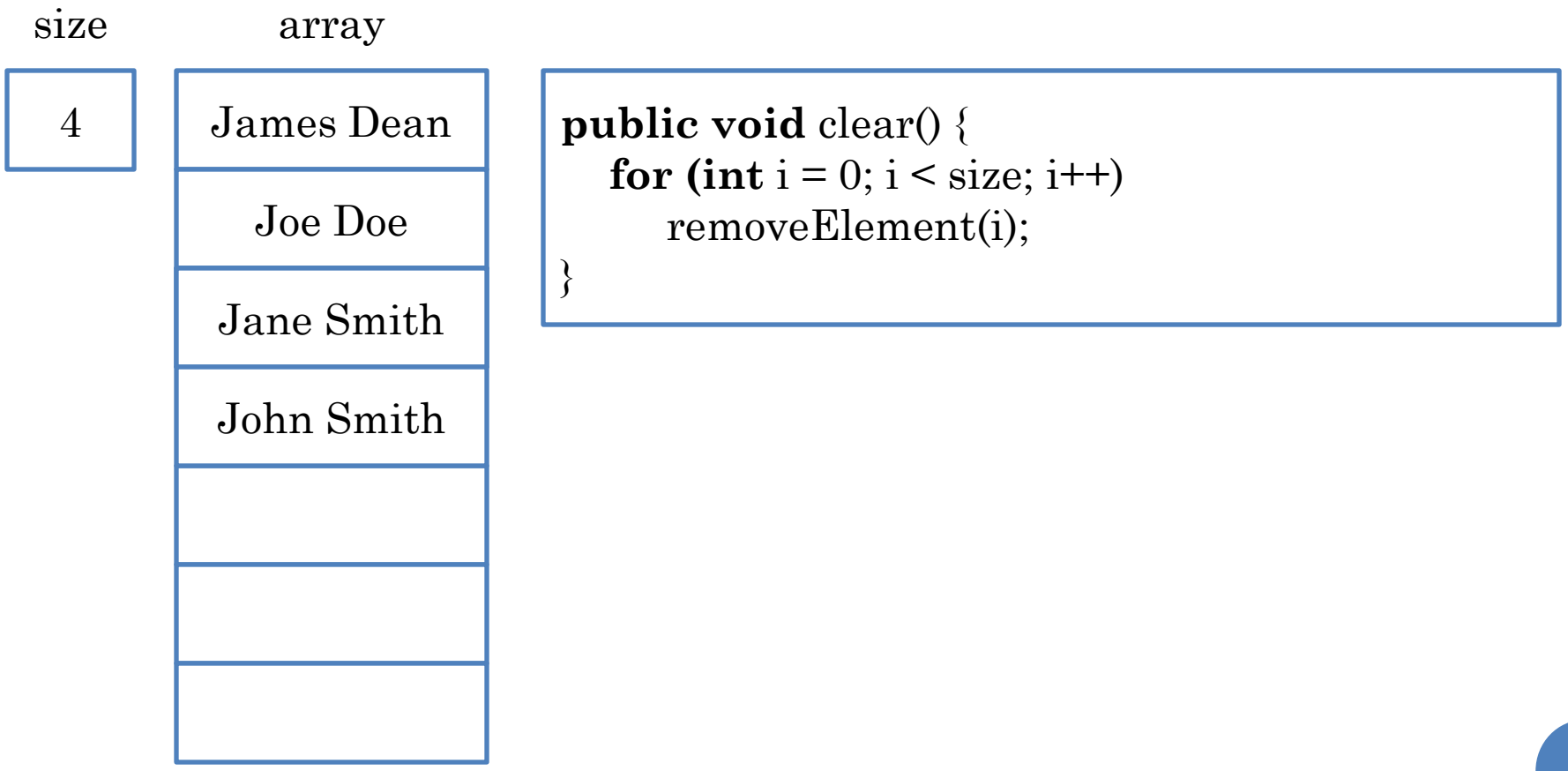



```
public void clear() {
    for (int i = 0; i < size; i++)
       contents[i] = null;
 }
```

Collection size remains the same, now null elements in the collection

PUBLIC VOID CLEAR()

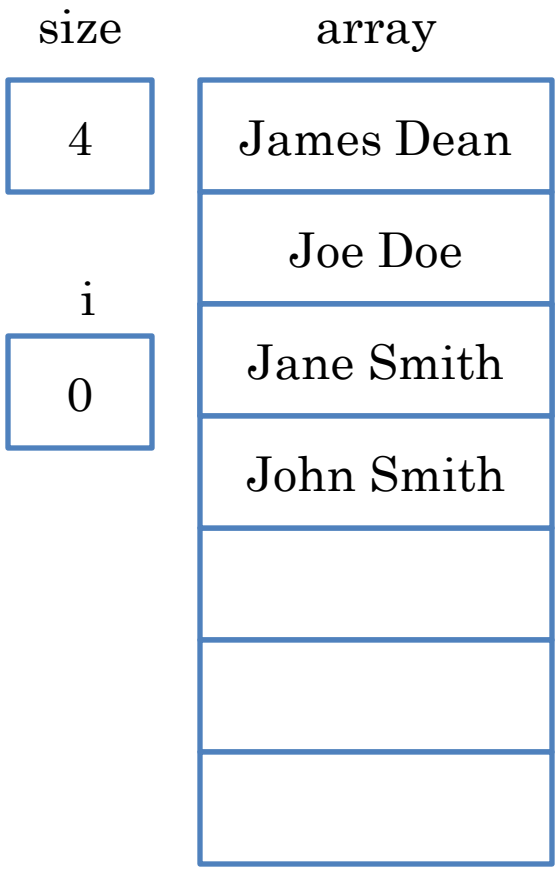
clear();




```
public void clear() {
    for (int i = 0; i < size; i++)
        removeElement(i);
}
```

PUBLIC VOID CLEAR()

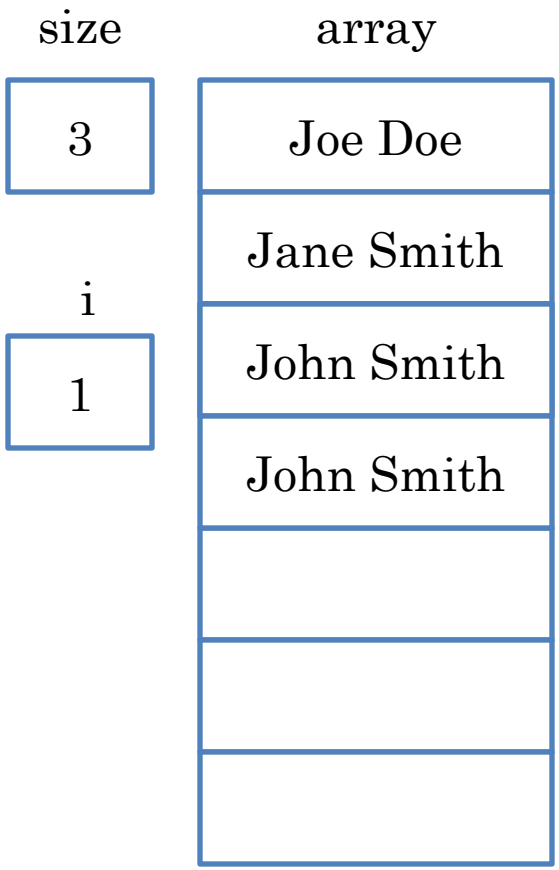
clear();




```
public void clear() {
    for (int i = 0; i < size; i++)
        removeElement(i);
}
```

PUBLIC VOID CLEAR()

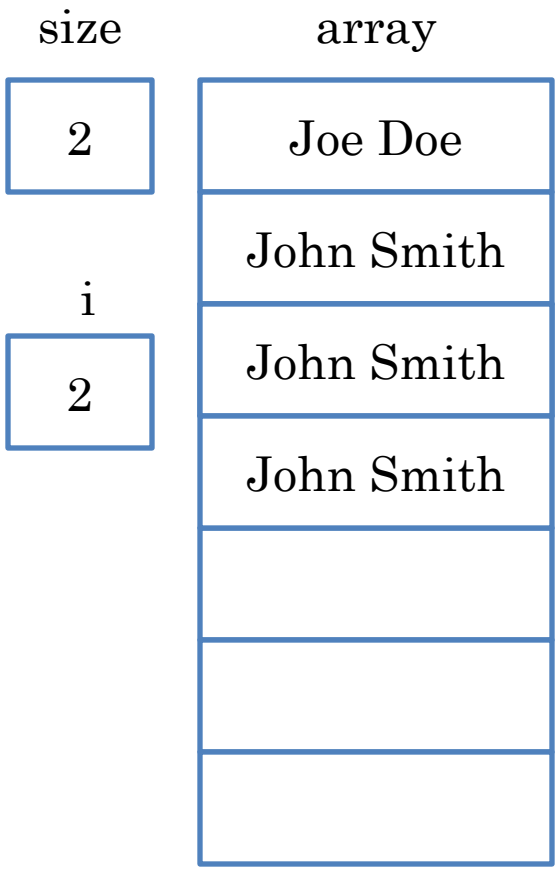
clear();




```
public void clear() {
    for (int i = 0; i < size; i++)
        removeElement(i);
}
```

PUBLIC VOID CLEAR()

clear();




```
public void clear() {
    for (int i = 0; i < size; i++)
        removeElement(i);
}
```

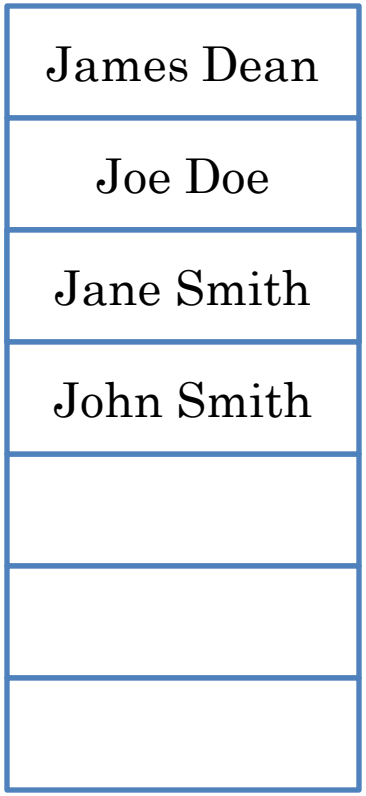
PUBLIC VOID CLEAR()

clear();

size

0

array

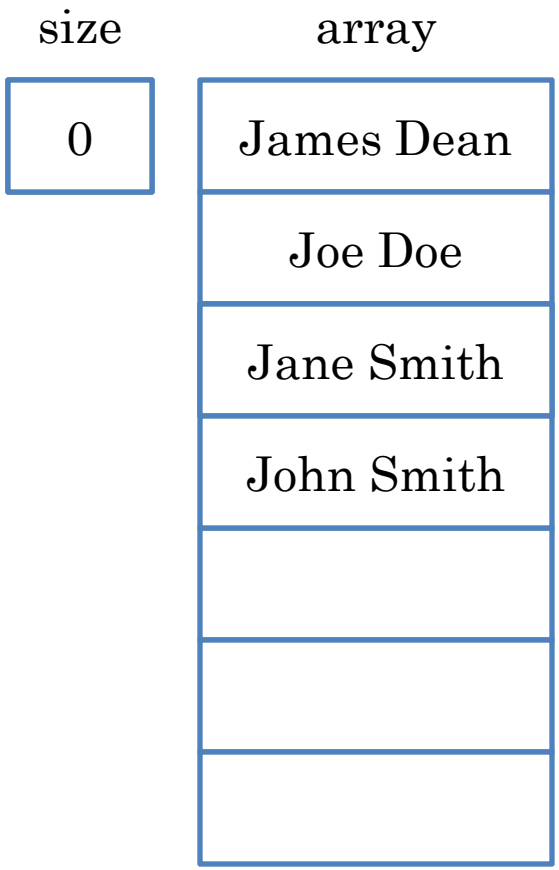



```
public void clear() {
    while (size > 0)
        removeElement(size -1);
}
```

Correct but inefficient

PUBLIC VOID CLEAR()

clear();

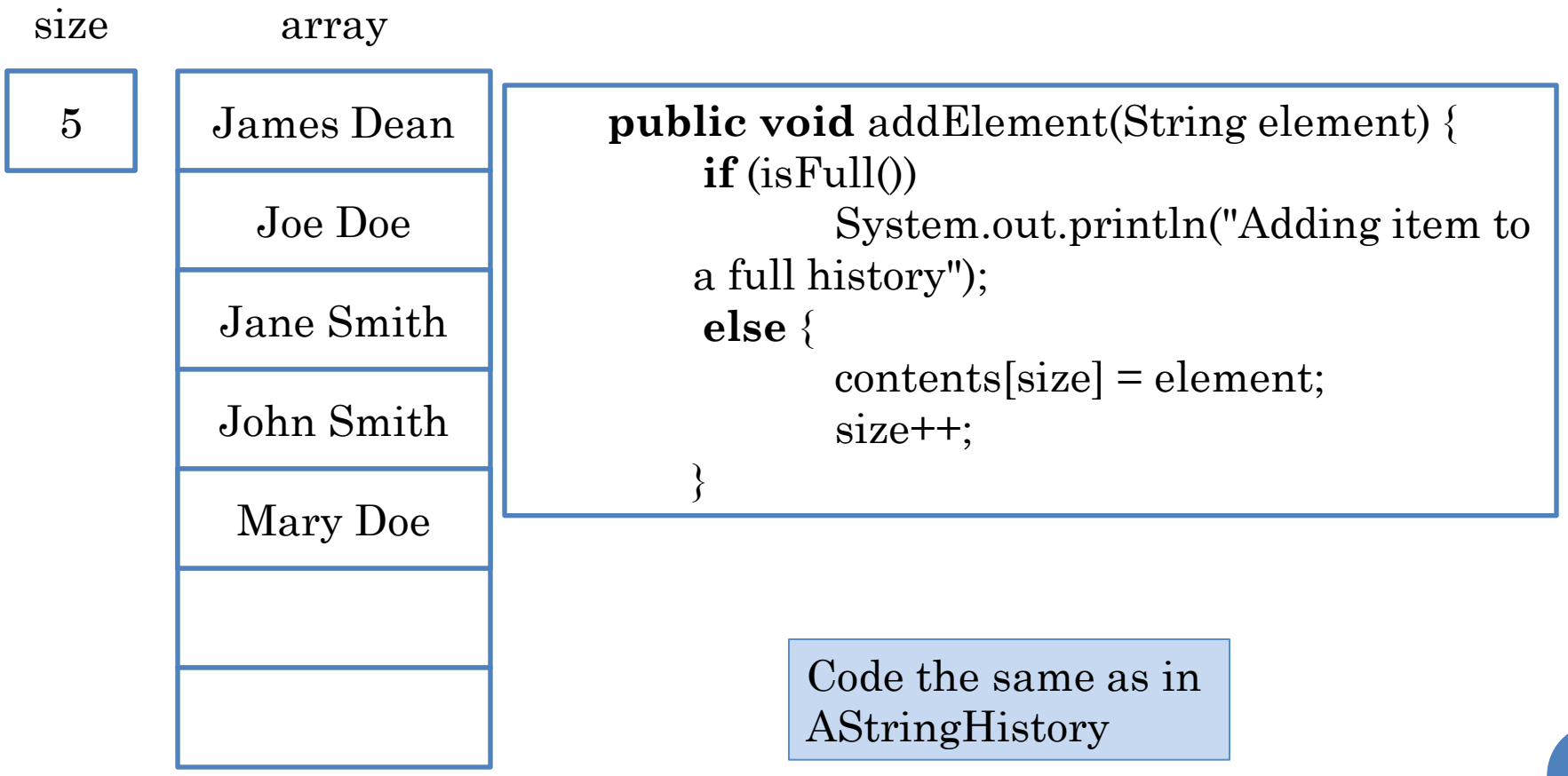



```
public void clear() {
    size = 0;
}
```

So simple only because we don't do anything with the deleted elements (such as clear them or delete related elements)

addElement("Mary Doe");

addElement("Mary Doe");



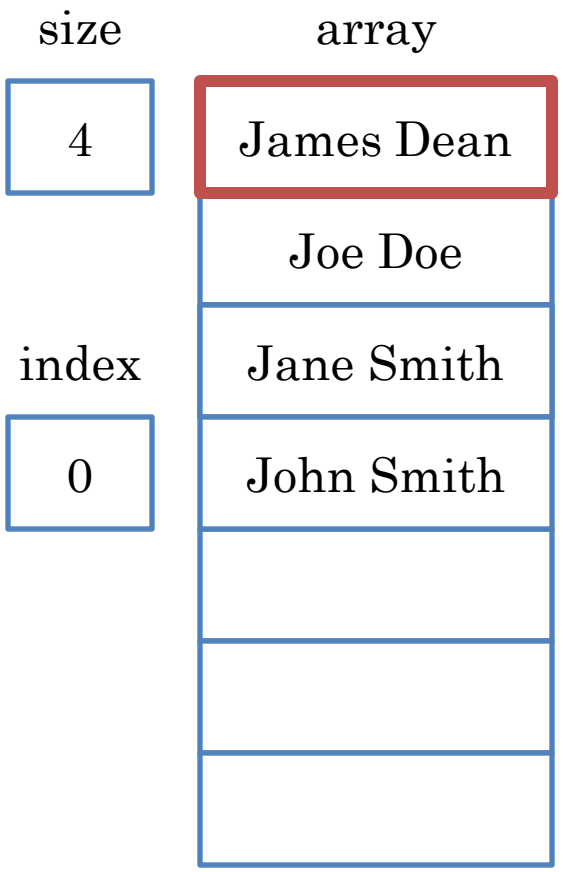

```
public void addElement(String element) {
        if (isFull())
                System.out.println("Adding item to
        a full history");
        else {
                contents[size] = element;
                size++;
        }
```

Code the same as in AStringHistory

Extra Slides

indexOf(String element)

indexOf("James Dean");



Visualization of Variable-Sized Collections

```
public interface  StringHistory  {
        public void addElement(String element);
        public int size();
        public String elementAt(int index);
}
```

```
public interface  StringDatabase  {
        //from history
        public int size();
        public void addElement(String element);
        public String elementAt(int index);

        //additional methods
         public void removeElement(String element);
         public boolean member(String element);
         public void clear();
}
```