Linear Structures

The Linear List

١

Abstract Definition

A Linear List is an ordered collection of elements from set S. The order is defined either by position or by other kind of ordering.

In other words, a linear list is either empty or can be written as:

$$(a_1, a_2, a_3, \ldots, a_n)$$

Where a_i are elements of some set S.

١٢

Specifications

- Each element of a list is assumed to have at least two fields, as shown:
- Keys are unique identities of the elements.

- The list contains n elements.

١٣

Structure

- There is a linear relationship between elements.
- Each element in the list has a unique position.
- If the position of the first element is k, then the position of its successor is k+1.

Example:

Element X has position 2; Its successor is A, which has position 3.

Note: Data or Keys?

Position	1	2	3	4	5	6	7
Element	E	X	A	M	P	L	E

١٤

Operations

Some of the operations assume there is one element in the list designated as current.

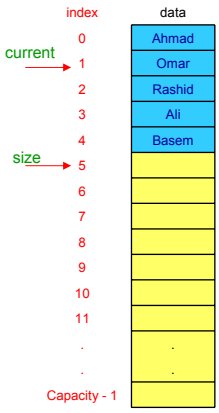
first()
last()
next()
prior()
seek(position)
search(target)
get()
set(entry)

getPosition()
addBefore(entry)
addAfter(entry)
remove()
isElement()
isEmpty() , isFull()
size()
clear()

١٥

List Implementation Using Arrays

- Use a partially filled array of fixed capacity
- Use two integer variables:
  - current -- points to the current element.
  - size -- points to the first empty position in the array.
- An empty list is initialized by setting current = size = 0.
- At any time, if current == size, then there is no current element.

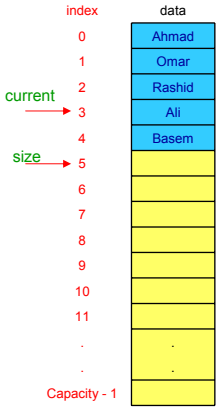


١٦

Mapping Operation: first() ... *O*(1)

- **Postcondition**:
  - First element on the list becomes the current element.
  - If the list is empty, there is no current element.
- **Code**:

```
current = 0;
```

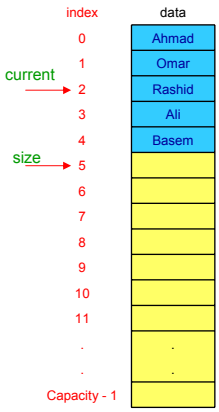


١٧

Mapping Operation: last() ... *O*(1)

- **Postcondition**:
  - Last element on the list becomes the current element.
  - If the list is empty, there is no current element.
- **Code**:

```
if (size > 0)
  current = size - 1;
else
  current = 0;
```

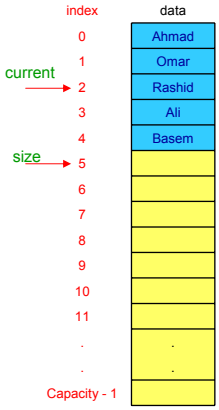


١٨

Mapping Operation: next() ... *O*(1)

- **Precondition:**
  - isElement() returns true.
- **Postcondition**:
  - If current is at the last element, then there is no current element.
  - Otherwise, the new current is the element immediately after the original current element.
- **Code**:

```
assert  isElement();
++current;
```

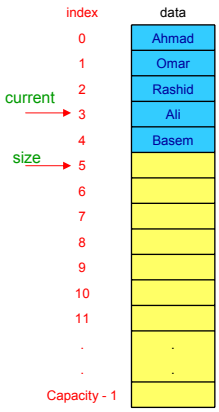


١٩

Mapping Operation: next() ... *O*(1)

- **Precondition:**
  - isElement() returns true.
- **Postcondition**:
  - If current is at the last element, then there is no current element.
  - Otherwise, the new current is the element immediately after the original current element.
- **Code**:

```
assert  isElement();
++current;
```

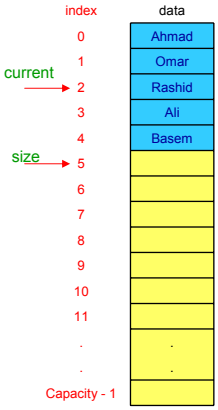


٢٠

Mapping Operation: prior() ... *O*(1)

- **Precondition:**
  - isElement() returns true.
- **Postcondition**:
  - If current is at the first element, then there is no current element.
  - Otherwise, the new current is the element immediately before the original current element
- **Code**:

```
assert  isElement();
 If (current > 0)
   --current;
else
   current = size;
```

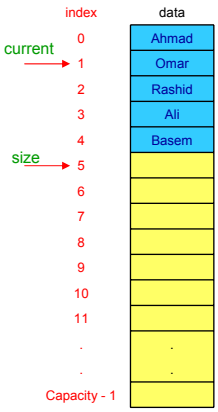


٢١

Mapping Operation: prior() ... *O*(1)

- **Precondition:**
  - isElement() returns true.
- **Postcondition**:
  - If current is at the first element, then there is no current element.
  - Otherwise, the new current is the element immediately before the original current element
- **Code**:

```
assert  isElement();
 If (current > 0)
   --current;
else
   current = size;
```



٢٢

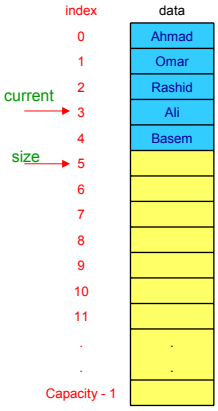
Mapping Operation: seek(position) ... *O*(1)

- **Postcondition**:
  - If position is within the list, the element at that position becomes the current element
  - Otherwise, there is no current element.
- **Code**:

```
If (position < size)
  current = position;
else
  current = size;
```

- **Example**:

```
seek(1);
```



٢٣

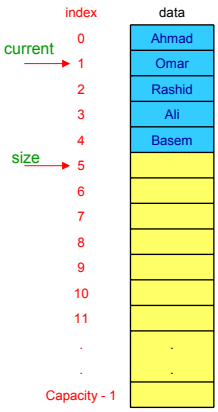
Mapping Operation: seek(position) ... *O*(1)

- **Postcondition**:
  - If position is within the list, the element at that position becomes the current element
  - Otherwise, there is no current element.
- **Code**:

```
If (position < size)
  current = position;
else
  current = size;
```

- **Example**:

```
seek(1);
```



٢٤

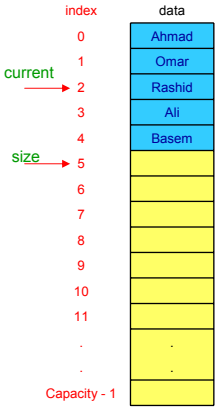
Mapping Operation: get() ... *O*(1)

- **Precondition**:
  - isElement() returns true.
- **Postcondition**:
  - The element returned is the current element in the list.
- **Code**:

```
assert  isElement();
return data[current];
```

- **Example**:

  get(); ➔ returns "Rashid"



٢٥

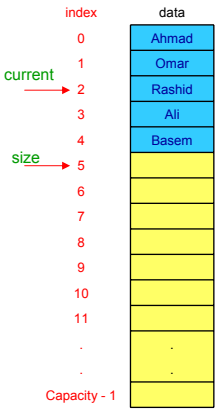
Mapping Operation: set(entry) ... *O*(1)

- **Precondition**:
  - isElement() returns true.
- **Postcondition**:
  - The value of the current element has changed to entry.
- **Code**:

```
assert  isElement();
data[current] = entry;
```

- **Example**:

  set("Rami");



٢٦

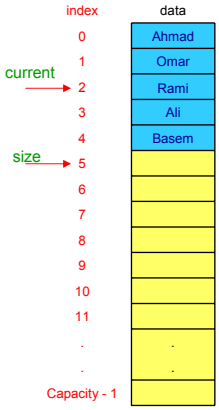
Mapping Operation: set(entry) ... *O*(1)

- **Precondition**:
  - isElement() returns true.
- **Postcondition**:
  - The value of the current element has changed to entry.
- **Code**:

```
assert  isElement();
data[current] = entry;
```

- **Example**:

```
set("Rami");
```



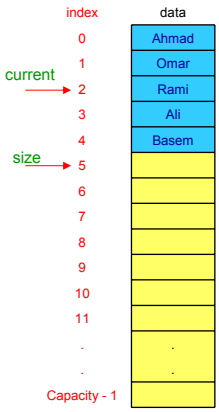
Mapping Operation: getPosition() ... *O*(1)

- **Postcondition**:
  - The returned value is the index of the current element in the list.
- **Code**:

```
return current;
```

- **Example**:

getPosition(); ➔ returns 2.



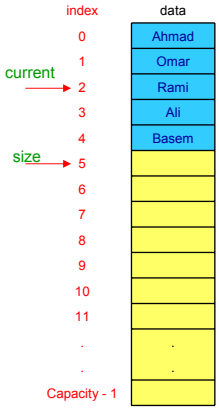
Mapping Operation: isElement() ... *O*(1)

- **Postcondition**:
  - A true return value indicates that there is a valid "current" element.
  - A false return value indicates that there is no valid current element.
- **Code**:

```
return (current < size);
```

- **Example**:

  isElement(); ➔ returns true.



٢٩

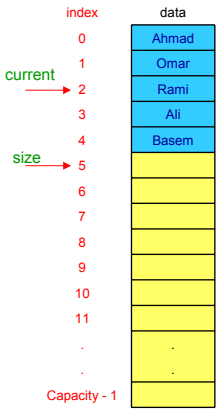
Mapping Operation: isEmpty() ... *O*(1)

- **Postcondition**:
  - The return value is true if the list has no elements, otherwise, it is false.
- **Code**:

```
return (size == 0);
```

- **Example**:

  isEmpty(); ➔ returns false.



٣٠

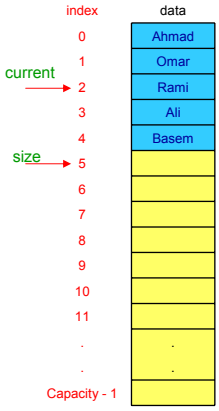
Mapping Operation: isFull() ... *O*(1)

- **Postcondition**:
  - The return value is true if the list has a number of elements equal to its capacity, otherwise it is false.
- **Code**:

```
return (size == CAPCITY);
```

- **Example**:

  isFull(); ➔ returns false.



٣١

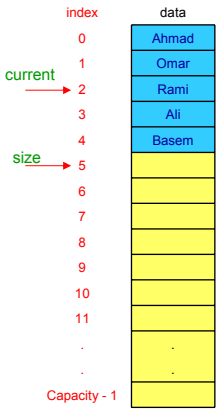
Mapping Operation: size() ... *O*(1)

- **Postcondition**:
  - The return value is the number of elements in the list.
- **Code**:

```
return size;
```

- **Example**:

  size(); ➔ returns 5.



٣٢

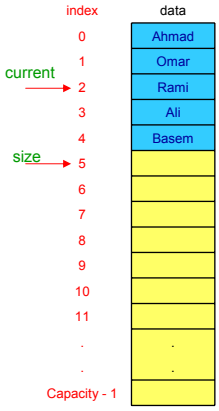
Mapping Operation: clear() ... *O*(1)

- **Postcondition**:
  - The list is now empty.
- **Code**:

```
size = current = 0;
```

- **Example**:

```
clear();
```



٣٣

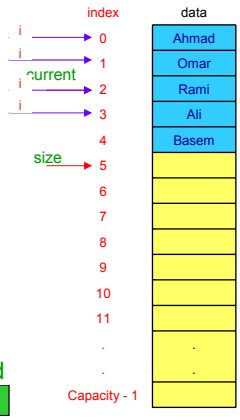
Mapping Operation: search(target) ... *O*(n)

- **Postcondition**:
  - The list has been searched for the target. If the target was present, then the found target is now the current element. Otherwise, there is no current element.
- **Code**:

```
int i = 0;
boolean found = false;
while ((i < size) && !found) {
    if (data[i].compareTo(target) == 0)
        found = true;
    else
        i++;
}
current = i;
```

- **Example**:

```
search("Ali");
```



found
true

٣٤

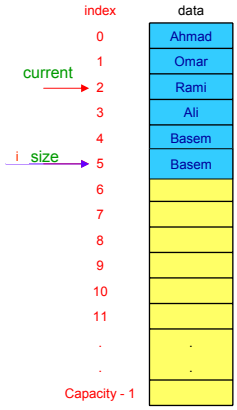
Mapping Operation: addBefore(entry) ... *O*(n)

- **Precondition**:
  - size < CAPACITY.
- **Postcondition**:
  - A copy of entry has been added to the list before the current element. If there was no current element, then entry has been added at the front of the list. In either case, the newly inserted element becomes the current element of the list.
- **Code**:

```
int i;
assert  size < CAPACITY;
if (!isElement())  current= 0;
for (i = size; i > current; i--)
    data[i] = data[i-1];
data[current] = entry;
size++;
```

- **Example**:

```
addBefore("Fouad");
```



٣٧

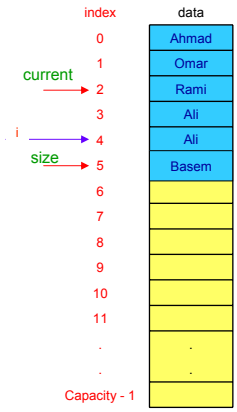
Mapping Operation: addBefore(entry) ... *O*(n)

- **Precondition**:
  - size < CAPACITY.
- **Postcondition**:
  - A copy of entry has been added to the list before the current element. If there was no current element, then entry has been added at the front of the list. In either case, the newly inserted element becomes the current element of the list.
- **Code**:

```
int i;
assert  size < CAPACITY;
if (!isElement())  current= 0;
for (i = size; i > current; i--)
    data[i] = data[i-1];
data[current] = entry;
size++;
```

- **Example**:

```
addBefore("Fouad");
```



٣٨

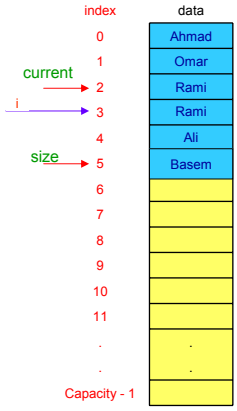
Mapping Operation: addBefore(entry) ... *O*(n)

- **Precondition**:
  - size < CAPACITY.
- **Postcondition**:
  - A copy of entry has been added to the list before the current element. If there was no current element, then entry has been added at the front of the list. In either case, the newly inserted element becomes the current element of the list.
- **Code**:

```
int i;
assert  size < CAPACITY;
if (!isElement())  current= 0;
for (i = size; i > current; i--)
    data[i] = data[i-1];
data[current] = entry;
size++;
```

- **Example**:

```
addBefore("Fouad");
```



٣٩

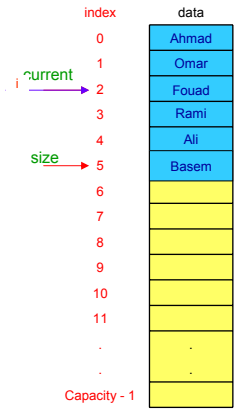
Mapping Operation: addBefore(entry) ... *O*(n)

- **Precondition**:
  - size < CAPACITY.
- **Postcondition**:
  - A copy of entry has been added to the list before the current element. If there was no current element, then entry has been added at the front of the list. In either case, the newly inserted element becomes the current element of the list.
- **Code**:

```
int i;
assert  size < CAPACITY;
if (!isElement())  current= 0;
for (i = size; i > current; i--)
    data[i] = data[i-1];
data[current] = entry;
size++;
```

- **Example**:

```
addBefore("Fouad");
```



٤٠

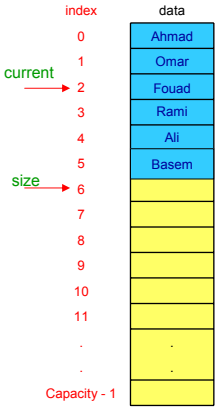
Mapping Operation: addBefore(entry) ... *O*(n)

- **Precondition**:
  - size < CAPACITY.
- **Postcondition**:
  - A copy of entry has been added to the list before the current element. If there was no current element, then entry has been added at the front of the list. In either case, the newly inserted element becomes the current element of the list.
- **Code**:

```
int i;
assert  size < CAPACITY;
if (!isElement())  current= 0;
for (i = size; i > current; i--)
    data[i] = data[i-1];
data[current] = entry;
size++;
```

- **Example**:

```
addBefore("Fouad");
```



٤١

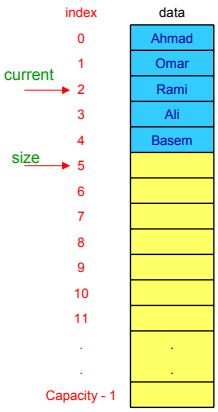
Mapping Operation: addAfter(entry) ... *O*(n)

- **Precondition**:
  - size < CAPACITY.
- **Postcondition**:
  - A copy of entry has been added to the list after the current element. If there was no current element, then entry has been added at the end of the list. In either case, the newly inserted element becomes the current element of the list.
- **Code**:

```
int i;
assert  size < CAPACITY;
if (isElement()) {
    ++current;
    for (i = size; i > current; i--)  data[i] = data[i-1]; }
data[current] = entry;
size++;
```

- **Example**:

```
addAfter("Fouad");
```



٤٢

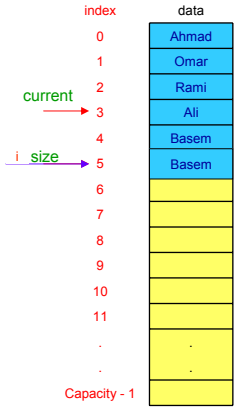
Mapping Operation: addAfter(entry) ... *O*(n)

- **Precondition**:
  - size < CAPACITY.
- **Postcondition**:
  - A copy of entry has been added to the list after the current element. If there was no current element, then entry has been added at the end of the list. In either case, the newly inserted element becomes the current element of the list.
- **Code**:

```
int i;
assert  size < CAPACITY;
if (isElement()) {
   ++current;
   for (i = size; i > current; i--)  data[i] = data[i-1]; }
data[current] = entry;
size++;
```

- **Example**:

```
addAfter("Fouad");
```



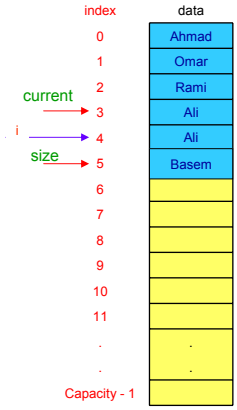
Mapping Operation: addAfter(entry) ... *O*(n)

- **Precondition**:
  - size < CAPACITY.
- **Postcondition**:
  - A copy of entry has been added to the list after the current element. If there was no current element, then entry has been added at the end of the list. In either case, the newly inserted element becomes the current element of the list.
- **Code**:

```
int i;
assert  size < CAPACITY;
if (isElement()) {
   ++current;
   for (i = size; i > current; i--)  data[i] = data[i-1]; }
data[current] = entry;
size++;
```

- **Example**:

```
addAfter("Fouad");
```



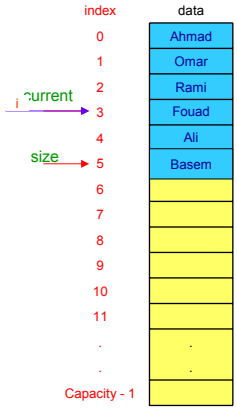
Mapping Operation: addAfter(entry) ... *O*(n)

- **Precondition**:
  - size < CAPACITY.
- **Postcondition**:
  - A copy of entry has been added to the list after the current element. If there was no current element, then entry has been added at the end of the list. In either case, the newly inserted element becomes the current element of the list.
- **Code**:

```
int i;
assert  size < CAPACITY;
if (isElement()) {
   ++current;
   for (i = size; i > current; i--)  data[i] = data[i-1]; }
data[current] = entry;
size++;
```

- **Example**:

```
addAfter("Fouad");
```



٤٥

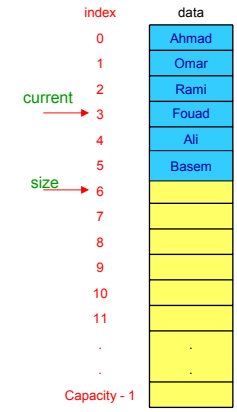
Mapping Operation: addAfter(entry) ... *O*(n)

- **Precondition**:
  - size < CAPACITY.
- **Postcondition**:
  - A copy of entry has been added to the list after the current element. If there was no current element, then entry has been added at the end of the list. In either case, the newly inserted element becomes the current element of the list.
- **Code**:

```
int i;
assert  size < CAPACITY;
if (isElement()) {
   ++current;
   for (i = size; i > current; i--)  data[i] = data[i-1]; }
data[current] = entry;
size++;
```

- **Example**:

```
addAfter("Fouad");
```



٤٦

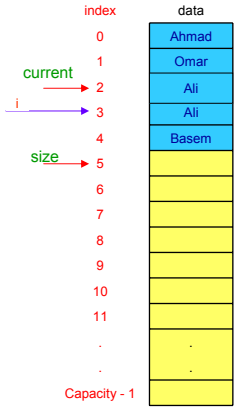
Mapping Operation: remove() ... *O*(n)

- **Precondition**:
  - isElement() returns true.
- **Postcondition**:
  - The current element has been removed from the list, and the element after it (if there is one) becomes the current element.
- **Code**:

```
int i;
assert  isElement();
for (i = current+1; i < size; i++)
   data[i-1] = data[i];
size--;
```

- **Example**:

```
remove();
```



٤٧

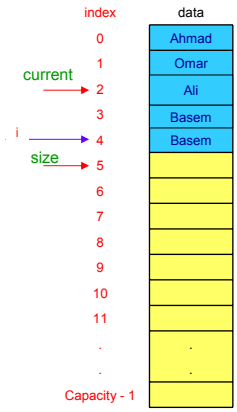
Mapping Operation: remove() ... *O*(n)

- **Precondition**:
  - isElement() returns true.
- **Postcondition**:
  - The current element has been removed from the list, and the element after it (if there is one) becomes the current element.
- **Code**:

```
int i;
assert  isElement();
for (i = current+1; i < size; i++)
   data[i-1] = data[i];
size--;
```

- **Example**:

```
remove();
```



٤٨

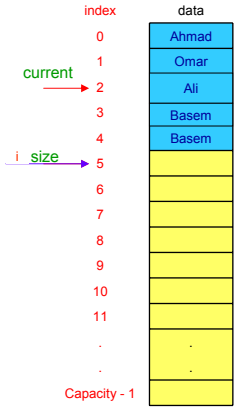
Mapping Operation: remove() ... *O*(n)

- **Precondition**:
  - isElement() returns true.
- **Postcondition**:
  - The current element has been removed from the list, and the element after it (if there is one) becomes the current element.
- **Code**:

```
int i;
assert  isElement();
for (i = current+1; i < size; i++)
   data[i-1] = data[i];
size--;
```

- **Example**:

```
remove();
```



٤٩

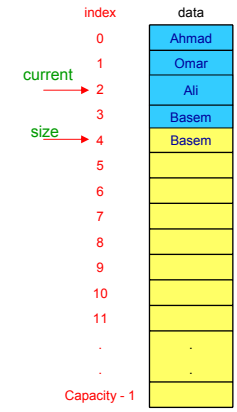
Mapping Operation: remove() ... *O*(n)

- **Precondition**:
  - isElement() returns true.
- **Postcondition**:
  - The current element has been removed from the list, and the element after it (if there is one) becomes the current element.
- **Code**:

```
int i;
assert  isElement();
for (i = current+1; i < size; i++)
   data[i-1] = data[i];
size--;
```

- **Example**:

```
remove();
```



٥٠

List Implementation in Java Using Arrays and Generic Classes

```
public class ArrayList<E extends Comparable> {
   private static final int CAPACITY = 30;
   private E[ ] data;
   private int size;
   private int current;


   // CONSTRUCTOR
   public ArrayList( ) {
      data = (E[ ]) new Comparable[CAPACITY];
      size = 0;
      current = 0;
   }
```

٥١

List Implementation in Java Using Arrays and Generic Classes

```
// MUTATOR METHODS
 public void first()              { current = 0; }
 public void last()               { ……. }
 public void next()               { assert isElement();  ++current }
 public void prior()              { ……. }
 public void seek(int position)          { ……. }
 public void search(E target)            { ……. }
 public void sort()               { ……. }
 public void set(E entry)         { ……. }
 public void addBefore(E entry)          { ……. }
 public void addAfter(E entry)           { ……. }
 public void remove()             { ……. }
 public void clear()              { size = 0;  current = 0; }
```

٥٢

List Implementation in Java Using Arrays and Generic Classes

```
    // OBSERVER METHODS
    public E get()          { assert isElement(); return data[current]; }
    public int getPosition()        { return current; }
    public int size()               { return size; }
    public boolean isElement() { return (current < size); }
    public boolean isEmpty()    { return size == 0; }
    public boolean isFull()         { return size == CAPACITY; }
}
```

٥٣