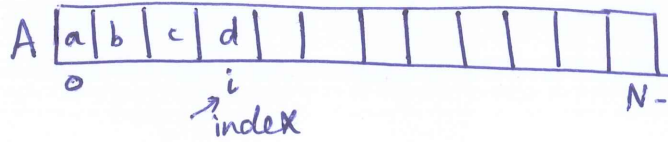


7.1 Array Lists



* $A[i] \rightarrow$ stores (a ref. to) the elem. with index i

* $N \rightarrow$ The capacity of A

* $n \rightarrow$ The # of elems in Array A

$\Rightarrow$ Array lists $\rightarrow$ access of any element by index

$\rightarrow$ index $\rightarrow$ # of elems before it

$\rightarrow$ The first elem in A has index 0 and the last elem. has index $N-1$

$\rightarrow$ if an elem. of A has index i , its previous elem (if it exists) has index $i-1$, and its next elem (if it exists) has index $i+1$.

* The Array List ADT

[1] $get(i)$:- برجع العنصر الموجود في index i

$$0 \leq i \leq size() - 1$$

[2] $set(i, e)$:- بديل العنصر الموجود في i index بالعنصر e
بديل العنصر الموجود في i index بالعنصر e $\rightarrow$ $0 \leq i \leq size() - 1$

$$0 \leq i \leq size() - 1$$

[3] $add(i, e)$:- عنصر e في i index

$$0 \leq i \leq size()$$

[4] $remove(i)$:- حذف العنصر الموجود في i index

$$0 \leq i \leq size() - 1$$

[5] $size()$:- برجع عدد العناصر الموجودة

[6] $isEmpty()$:- true اذا فارغ، false اذا ممتلئ

لازم التزم بالـ range والـ index error وخطـ exception

operation	output	Array List
$add(0, 7)$	-	(7)
$add(0, 4)$	-	(4, 7)
$get(1)$	7	(4, 7)
$add(2, 2)$	-	(4, 7, 2)
$get(3)$	error	(4, 7, 2)
$remove(1)$	7	(4, 2)
$add(1, 5)$	-	(4, 5, 2)
$add(3, 3)$	error	(4, 5, 2) $\rightarrow$

مابق احيف في i index و لا

ان index الـ قبله فارغ

⊛ Simple Array-Based Implementation

Algorithm add (i,e):

if ($n=N$) then
throw exception

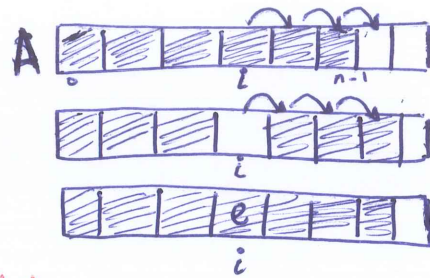
checkIndex($0 \leq i < \text{size}()$) $\rightarrow$ رینج میں آیا اندازہ ختم ہے

for $j = n-1, n-2, \dots, i$ do

بنقل اشارة للمهمة عن نفسه ليقضي هناك
للمهمة الجديد

ضعيف e في الجناح الجيد $\rightarrow e \leftarrow A[j]$

$n \leftarrow n+1$ → زود بعد عناصر اضافی



٤) باختصار / عناصر e عنصر x في index i لنزح أفقياً إلى مكانه فيجاء shift للمجموع
كل عناصر من عند $i-1$ إلى n لأعلى لأنه ببساطة إزاحة من لا خرجت فاندخل
العناصر على بعضها وتصحيح

$$\text{add}(i, e) \Rightarrow O(n) \quad \text{A}$$

Algorithm remove(i):

checkIndex($0 \leq i \leq \text{size}() - 1$)

معينة على الـ element، والـ حافة

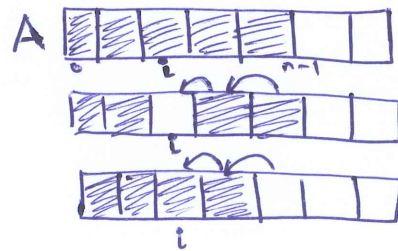
$e \leftarrow AC[i]$

for $j=i, i+1, \dots, n-2$ do

بنوعی انرا به نسیب می نامند
 مثل آنکه از صاف خالص
 $A[j] \leftarrow A[j+1]$
 $-n-1$

$$n \leftarrow n - 1$$

return e



④ heaph. / لا نجد elem. بان في index نزيد محل shift نبدأ ، من عند $i+1$ لـ $n-1$
عنه أعين مكانه الفاضل لـ في الـ حركت .

remove(i) $\Rightarrow O(n)$ $\Leftarrow$

Algorithm get(i);

checkIndex($0 \leq i \leq \text{size}() - 1$)

```
return A[i]
```

$get(i) \Rightarrow O(1)$

Algorithm set(i,e):

checkIndex (0 ≤ size() - 1)

تخزن القيم على زيف $tmp \leftarrow AC[i]$

$AC[i] \leftarrow e$ نضيف العنصر

return tmp رسمی

$$\text{set}(i, e) \rightarrow O(1)$$

* Implementing Deque using Array List

deque

Array List

size(), isEmpty $\rightarrow$ size(), isEmpty()

getFirst() $\rightarrow$ get(0)

getLast() $\rightarrow$ get(size()-1)

addFirst(e) $\rightarrow$ add(0, e)

addLast(e) $\rightarrow$ add(size(), e)

removeFirst() $\rightarrow$ remove(0)

removeLast() $\rightarrow$ remove(size()-1)

* Implementing an Array list using Extendable (Dynamic) Array

* بسبب حجم array المحدود، عند إضافة عنصر جديد، نحتاج إلى زيادة حجم array. في هذه الحالة، نستخدم exception. أي عند محاولة إضافة عنصر جديد، نستخدم exception.

* The steps of add(i, e) when $n=N$

① Allocate a new array B of capacity $2N$

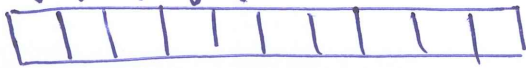
② let $B[i] \leftarrow A[i]$, for $i=0, \dots, N-1$

③ let $A \leftarrow B$

④ insert new element in A

B  $\leftarrow$ ①

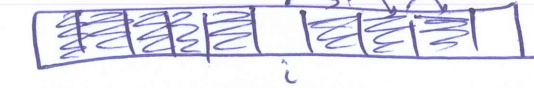
A 

B 

B 

A 

A 

A 

A 

3

Q Java Implementation of the add method in Array List ADT using an extendable (Dynamic) Array

```
public void add (int r, E e) throws IndexOutOfBoundsException {
```

```
    CheckIndex(r, size() + 1)  $\rightarrow 0 \leq r \leq \text{size}()$ 
```

```
    if (size == capacity) {
```

```
        capacity *= 2;  $\rightarrow 2N$  capacity
```

```
        E[] B = (E[]) new Object [capacity];  $\rightarrow$  new array B
```

```
        for (int i = 0; i < size; i++)
```

```
            B[i] = A[i];
```

```
        A = B;  $\rightarrow$  step ③
```

```
    }
```

```
    for (int i = size() - 1; i >= r; i++)
```

```
        A[i+1] = A[i];
```

```
    A[r] = e;
```

```
    size++;
```

```
}
```

7.2 Node Lists

يتعامل مع لنفد بشكل مباشر لأنه كل node ال reference عليه

* **position** :— place within a data structure where a single object is stored.
مكان فيه ال element ال بيدي اياه

Has one method : element() returns the element stored at the position

* **Node List** $\Rightarrow$ مجموعة من ال positions مخزنه فيه elements

* **Node List ADT (for node list S)**

[1] **first()** :— يرجع ال position ال موجود فيه أول عنصر ، بصير error لو S خاضعة بيني
position = null

[2] **Last()** :— يرجع ال position ال موجود فيه آخر عنصر ، بصير error لو S خاضعة بيني
position = null

[3] **before(p)** :— يرجع ال position ال موجود فيه العنصر ال قبل العنصر الموجود في position (p)
بصير error اذا كانت p هو أول position (ما في قبله شيء)

[4] **after(p)** :— يرجع ال position ال موجود فيه العنصر ال بعد العنصر الموجود في position (p)
بصير error اذا كانت p هو آخر position (ما في بعده شيء)

[5] **set(p, e)** :— بتسبيل ال elem ال موجود في position p بال elem ال بتراجع
ال elem ال كان موجود ب p انا ال position

[6] **addFirst(e)** :—

بتضيف e elem في الاول

[7] **addLast(e)** :—

بتضيف e elem في الآخر

[8] **addBefore(p, e)** :—

بتضيف e elem قبل position p

[9] **addAfter(p, e)** :—

بتضيف e elem بعد position p

[10] **remove(p)** :—

بتحذف و بتراجع ال elem ال بال position p
مبتلاغي هادا ال position

operation	Output	Node List S
addFirst(8)	—	(8)
first()	P ₁ (8)	(^{P₁} 8)
addAfter(p ₁ , 5)	—	(^{P₁} 8, ^{P₂} 5)
next(p ₁)	P ₂ (5)	(^{P₁} 8, ^{P₂} 5)
addBefore(p ₂ , 3)	—	(^{P₁} 8, ^{P₃} 3, ^{P₂} 5) ←
prev(P ₂)	P ₃ (3)	(^{P₁} 8, ^{P₃} 3, ^{P₂} 5)
addFirst(9)	—	(^{P₄} 9, ^{P₁} 8, ^{P₃} 3, ^{P₂} 5)
Last()	P ₂ (5)	(^{P₄} 9, ^{P₁} 8, ^{P₃} 3, ^{P₂} 5)
remove(first())	9	(^{P₁} 8, ^{P₃} 3, ^{P₂} 5)
set(P ₃ , 7)	3	(^{P₁} 8, ^{P₃} 7, ^{P₂} 5)
addAfter(first(), 2)	—	(^{P₁} 8, ^{P₅} 2, ^{P₃} 7, ^{P₂} 5) ←

گزی اندہ elem

بصیفہ بیا خہ position جو یہ

بوقت نظر عنہ مکانہ

دی لہ سیرط ترتیب

P₁, P₂, P₃

الترتیب پائی جب میںہ بیضاف

بعد میںہ

دہ نہ مبل خلیا remove

ایہ بتلہی ان position ← یعنی

P₄ البتہ فلہ اصیفہ عہ تلوہ P₅

(*) Reasons for a position p to be invalid include:

- p = null
- p was previously deleted from the list
- p is a position of a different list
- p is the first position of the list and we call prev(p)
- p is the Last " " " " " " next(p)

(*) position interface:

public interface Position<E> {

/** return the element stored at this position */

E element();

}

* Doubly Linked List Implementation

```

public class DNode<E> implements Position<E> {
    private DNode<E> prev, next;
    private E element;

    public DNode(DNode<E> newPrev, DNode<E> newNext, E elem) {
        prev = newPrev;
        next = newNext;
        element = elem;
    }

    public E element() throws InvalidPositionException {
        if ((prev == null) || (next == null))
            throw new InvalidPositionException("Position is not in a list");
        return element;
    }

    public DNode<E> getNext() { return next; }
    public DNode<E> getPrev() { return prev; }
    public void setNext(DNode<E> newNext) { next = newNext; }
    public void setPrev(DNode<E> newPrev) { prev = newPrev; }
    public void setElement(E newElement) { element = newElement; }
}

```

Constructor

Accessor methods

Update methods

Node List Implementation

Node list implementing DLL

ADT Node list implementing DLL

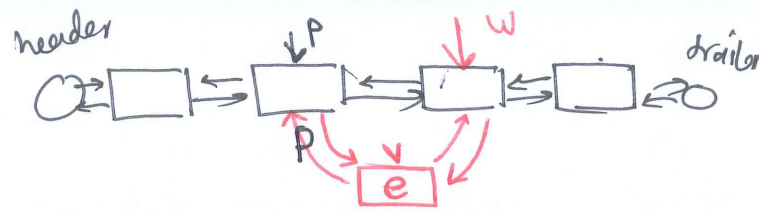
prev(p) ← node before position p

addAfter(p) ← add new node after position p

remove(p) ← remove node at position p

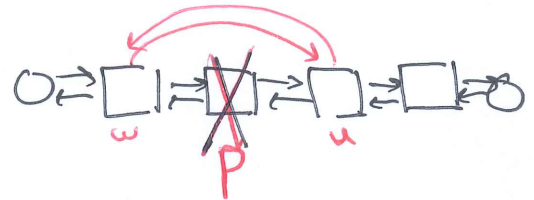
Algorithm addAfter(p, e):

create a new node v
 v.setElement(e)
 v.setPrev(p)
 v.setNext(p.getNext())
 (p.getNext()).setPrev(v)
 p.setNext(v)



Algorithm remove(p):

t ← p.element()
 (p.getPrev()).setNext(p.getNext())
 (p.getNext()).setPrev(p.getPrev())
 p.setPrev(null)
 p.setNext(null)
 return t



* Node List Implementation in Java

عند استخدام صندوق linked list لازم أصل ال position ال node فالبكي
 مع اعمل casting، بس قبل ال casting لازم أتأكد انه ال position ← valid

(الطالان ال عينا قبل)
 ← انه مت null
 ← ليس header
 ← ليس trailer
 ← ليس next و prev ال node بعد انك مت null
 after casting →
 CheckPosition(p) →

المبني الي بيقوم براد ال قبل
 ← لو بي استخدم node ب استخدام position
 ← position وبي، لكن
 ← خيري اعمل Casting لانه node

- * Node is a position
- * position is not a node

```
public class NodePositionList<E> implements PositionList<E> {
```

```
    protected int numElts;
```

```
    protected DNode<E> header, trailer;
```

```
    public NodePositionList() {
```

```
        numElts = 0;
```

```
        header = new DNode<E>(null, null, null);
```

```
        trailer = new DNode<E>(header, null, null);
```

```
        header.setNext(trailer);
```

```
    }
```

```
    protected DNode<E> checkPosition(Position<E> p) throws InvalidPositionException {
```

```
        if (p == null) position is null
```

```
            throw new InvalidPositionException("Null position passed to NodeList");
```

```
        if (p == header) dummy node, i.e. node is header
```

```
            throw new InvalidPositionException("The header node is not a valid position");
```

```
        if (p == trailer) dummy node, i.e. node is trailer
```

```
            throw new InvalidPositionException("The trailer node is not a valid position");
```

```
        try {
```

```
            DNode<E> tmp = (DNode<E>) p; ← casting
```

```
            if ((tmp.getPrev() == null) || (tmp.getNext() == null)) ← next or prev is null, casting failed, i.e. node is null
```

```
                throw new InvalidPositionException("Position doesn't belong to a valid node in list");
```

```
            return tmp; ← node is valid
```

```
        } catch (ClassCastException e) {
```

```
            throw new InvalidPositionException("Position is of wrong type for this list");
```

```
        }
```

```
    }
```

```
    public int size() { return numElts; }
```

```
    public boolean isEmpty() {
```

```
        return (numElts == 0);
```

```

public Position <E> first() throws EmptyListException {
    if (isEmpty()) throw new EmptyListException("List is Empty");
    return header.getNext(); ← ارجع اول node من القائمة
}

```

```

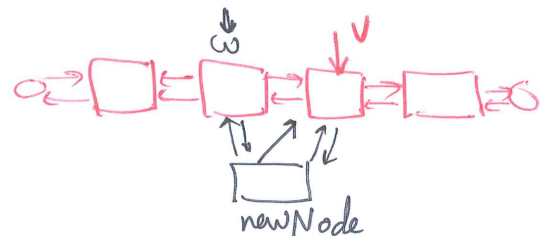
public Position <E> prev(Position <E> p) throws InvalidPositionException {
    DNode <E> v = checkPosition(p); ← node و position في القائمة
    DNode <E> prev = v.getPrev();
    if (prev == header) throws new BoundaryValidationException (
        "Can't advance past the beginning of the list");
    ← invalid position ← node و prev ان node null
    return prev;
}

```

```

public void addBefore(Position <E> p, E element) throws InvalidPositionException {
    DNode <E> v = checkPosition(p); ← نفس node الذي في p
    numElts++; ← size بزيادة
    DNode <E> newNode = new DNode <E> (v.getPrev(), v, element);
    v.getPrev().setNext(newNode); ← v.getPrev() و newNode
    v.setPrev(newNode); ← v و newNode
}

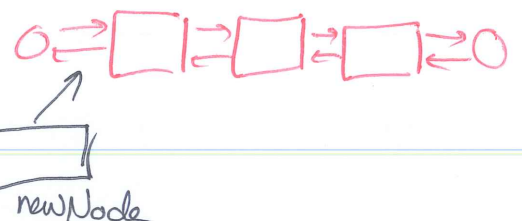
```



```

public void addFirst(E element) {
    numElts++;
    DNode <E> newNode = new DNode <E> (header, header.getNext(), element);
    header.getNext().setPrev(newNode); ← header و newNode
    header.setNext(newNode); ← header و newNode
}

```

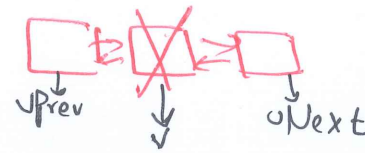


Public E remove(Position<E> p) throws InvalidPositionException {

DNode<E> v = checkPosition(p);

numElt --;

DNode<E> vPrev = v.getPrev();
DNode<E> vNext = v.getNext();



vPrev.setNext(vNext);

vNext.setPrev(vPrev);

E vElem = v.element();

v.setNext(null);

v.setPrev(null);

return vElem;

Public E set(Position<E> p, E element) throws InvalidPositionException {

DNode<E> v = checkPosition(p);

E oldElt = v.element();

v.setElement(element);

return oldElt;

}

7.3 Iterators

An iterator is a software design pattern that abstracts the process of scanning through a collection of elements one element at a time.

اللايتريتر نمثل على كل العناصر الموجودة بالـ data structure

* Iterators ADT

1 hasNext(): - Test whether there are elements left in the iterator
تفحص اذا ضايل عناصر باللايتريتر true وانما خالي يرجع false

2 next(): - returns the next element
يرجع العنصر التالي باللايتريتر يعني يرجع اول عنصر ضايل باللايتريتر

For Example :-

ArrayList A

Iterator I = A.iterator();

while (I.hasNext())

e = I.next();

بدی اهری علی لیست ArrayList

فهرست iterator

عبر علی عناصر لا یرتبه
عبر غیر

*** Positions():** this method returns an Iterable object (Like an array list or node list) containing the positions in the collection as elements
 ← یعنی بقیه method positions علی این که بقیه الی بقیه الی positions
 بقیه الی positions بقیه الی الی بقیه الی بقیه الی

*** The java.util.Iterator interface contain third method (next(), hasNext())**
 which is remove()

[3] remove(): remove from the collection the element returned by the most recent call to next()

یعنی آخ element و حذف آن با next()
 (مثلاً) مثلاً بقیه الی بقیه الی بقیه الی بقیه الی

*** لا** استیم iterator بقیه الی بقیه الی بقیه الی بقیه الی reset و بقیه الی

الباقی لذلک Iterable interface بقیه الی بقیه الی multiple traversal با بقیه الی

الشیء iterators() returns an iterator of the elements in the collection

کل حده بنادی های الی بقیه الی iterator بقیه الی بقیه الی بقیه الی بقیه الی

ArrayList <Double> data;

Iterator <Double> walk = data.iterator();

while (walk.hasNext())

if (walk.next() < 0.0)

walk.remove();

کود ای الی الی remove الی الی

الباقی بقیه الی double

arrayList بقیه الی set of elements

عبر علی عناصر و لا الی الی
 بقیه الی بقیه الی
 الی الی