Piles

Introduction aux piles

L'interface `ca.umontreal.IFT2015.adt.Stack` et la classe `Stack` de Java

Implémentation avec un tableau

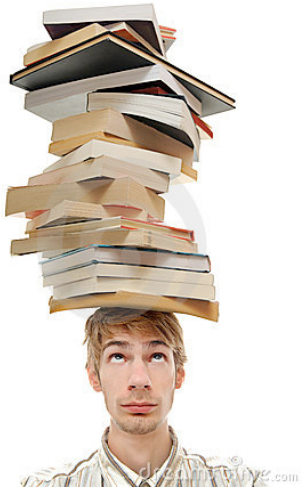
Implémentation avec une liste chaînée

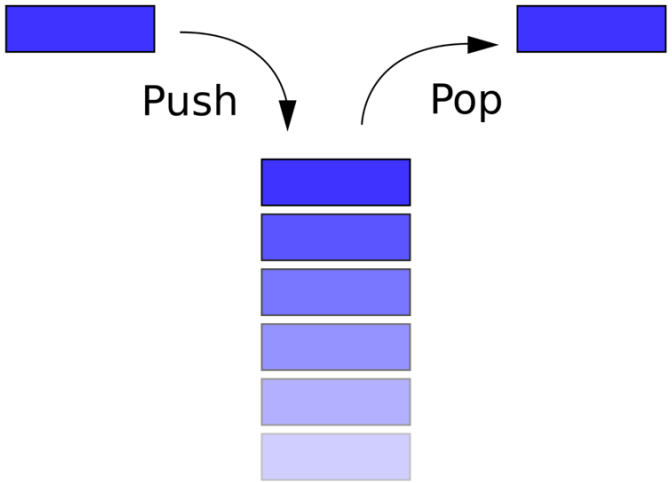
Utilisation

Introduction aux piles

La pile (Stack) est caractérisée par deux opérations :
`push` (empiler) et `pop` (dépiler)

La pile est caractérisée par sa politique de dernier entré premier sorti (last-in-first-out; LIFO).





Quelques applications

Directes :

- Historique (eg pages visitées dans un navigateur Web; commandes d'une App quelconque)
- Annuler une séquence dans un éditeur de texte
- Chaîne de méthodes appellées dans un langage qui prend en charge la récursivité

Indirectes :

- Structure de données auxiliaires pour des algorithmes
- Composant d'autres structures de données

L'interface `ca.umontreal.IFT2015.adt.Stack` et la classe `java.util.Stack`

`push( e )`	ajoute l'élément *e* sur le haut de la pile
`pop( )`	retire et retourne l'élément sur le haut de la pile, *`null`* si vide
`top( )`	retourne l'élément sur le haut de la pile, *`null`* si vide
`size( )`	retourne le nombre d'éléments dans la pile
`isEmpty( )`	retourne un booléen indiquant si la pile est vide

Method	Return Value	Stack Contents
push(5)	–	(5)
push(3)	–	(5, 3)
size( )	2	(5, 3)
pop( )	3	(5)
isEmpty( )	false	(5)
pop( )	5	( )
isEmpty( )	true	( )
pop( )	null	( )
push(7)	–	(7)
push(9)	–	(7, 9)
top( )	9	(7, 9)
push(4)	–	(7, 9, 4)
size( )	3	(7, 9, 4)
pop( )	4	(7, 9)
push(6)	–	(7, 9, 6)
push(8)	–	(7, 9, 6, 8)
pop( )	8	(7, 9, 6)

`java.util.Stack`

Method Summary

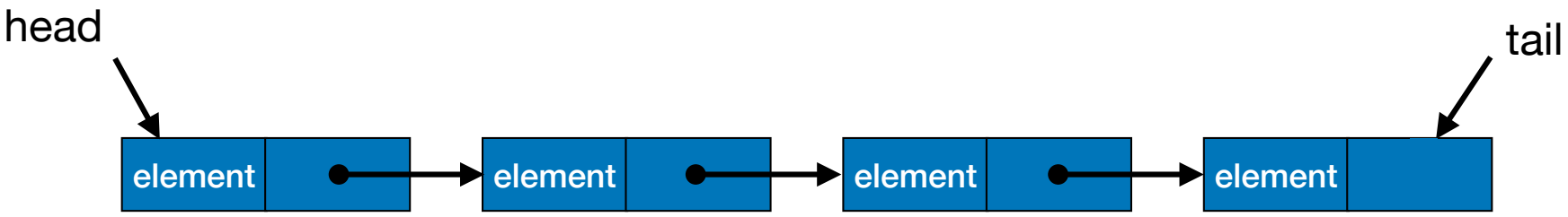
All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method and Description
`boolean`	**`empty()`** Tests if this stack is empty.
`E`	**`peek()`** Looks at the object at the top of this stack without removing it from the stack.
`E`	**`pop()`** Removes the object at the top of this stack and returns that object as the value of this function.
`E`	**`push(E item)`** Pushes an item onto the top of this stack.
`int`	**`search(Object o)`** Returns the 1-based position where an object is on this stack.

Implémentation dans un tableau, voir les codes...

Stack.java
ArrayStack.java

Implémentation dans une liste chaînée



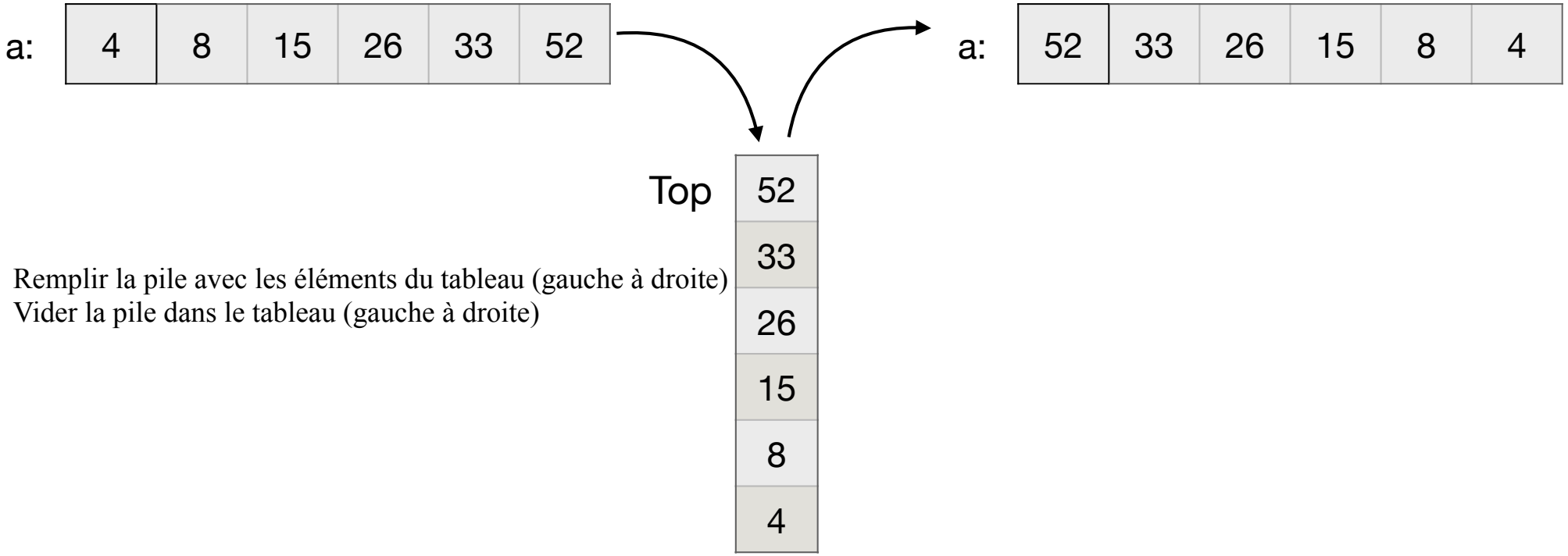
Adapter design pattern:

Stack Method	*Singly Linked List Method*
size( )	list.size( )
isEmpty( )	list.isEmpty( )
push(e)	list.addFirst(e)
pop( )	list.removeFirst( )
top( )	list.first( )

Implémentation dans un tableau, voir le code...

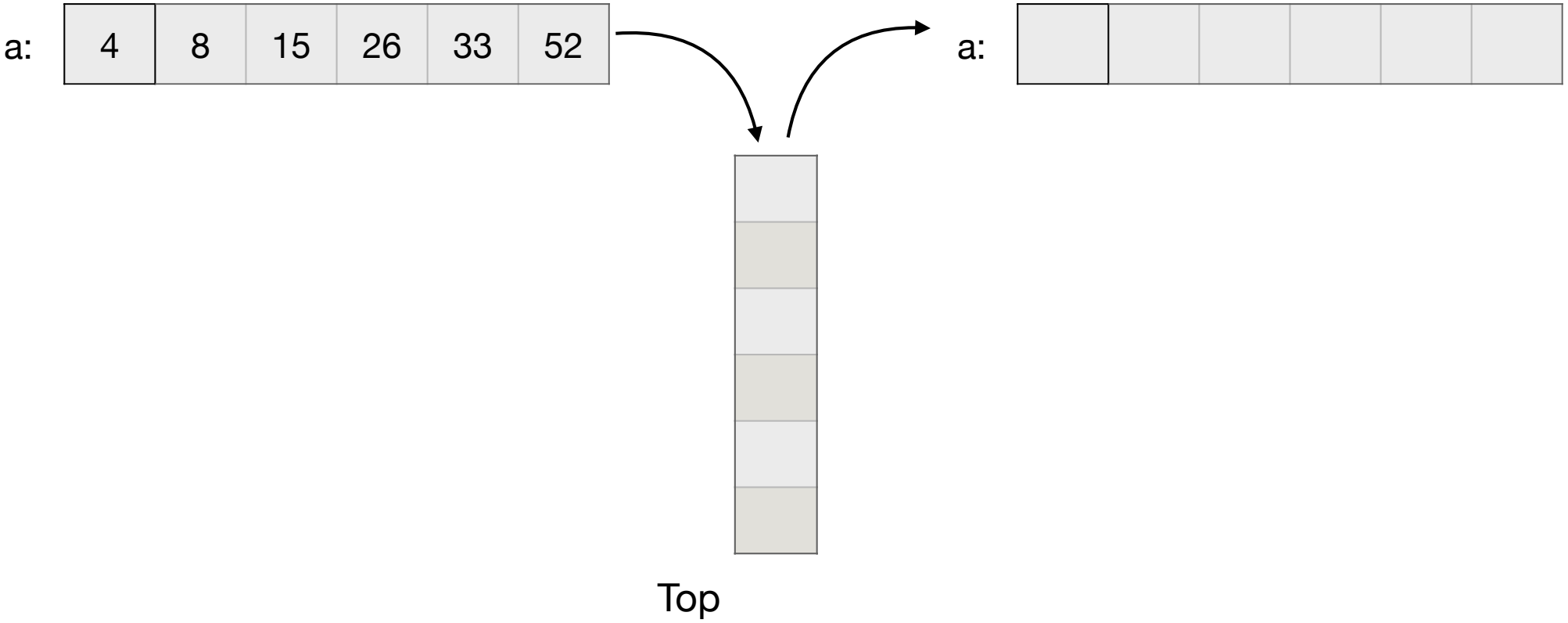
`LinkedListStack.java`

Application: Renverser les éléments d'un `Array`

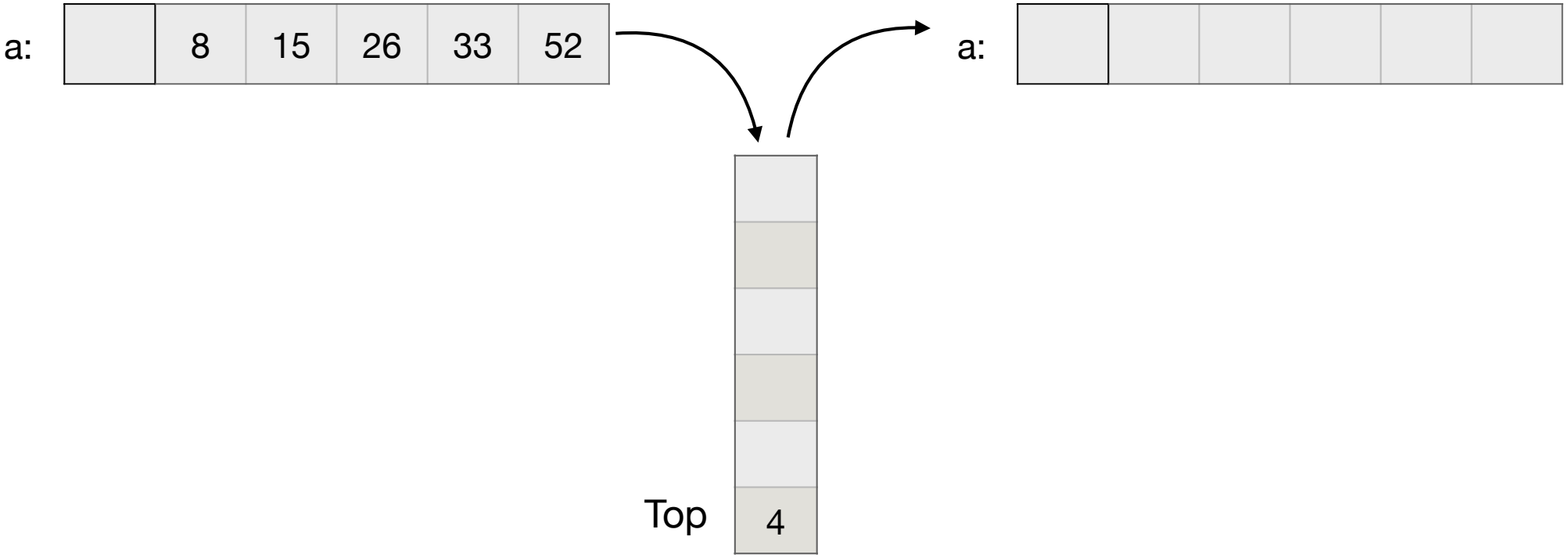


1. Remplir la pile avec les éléments du tableau (gauche à droite)
2. Vider la pile dans le tableau (gauche à droite)

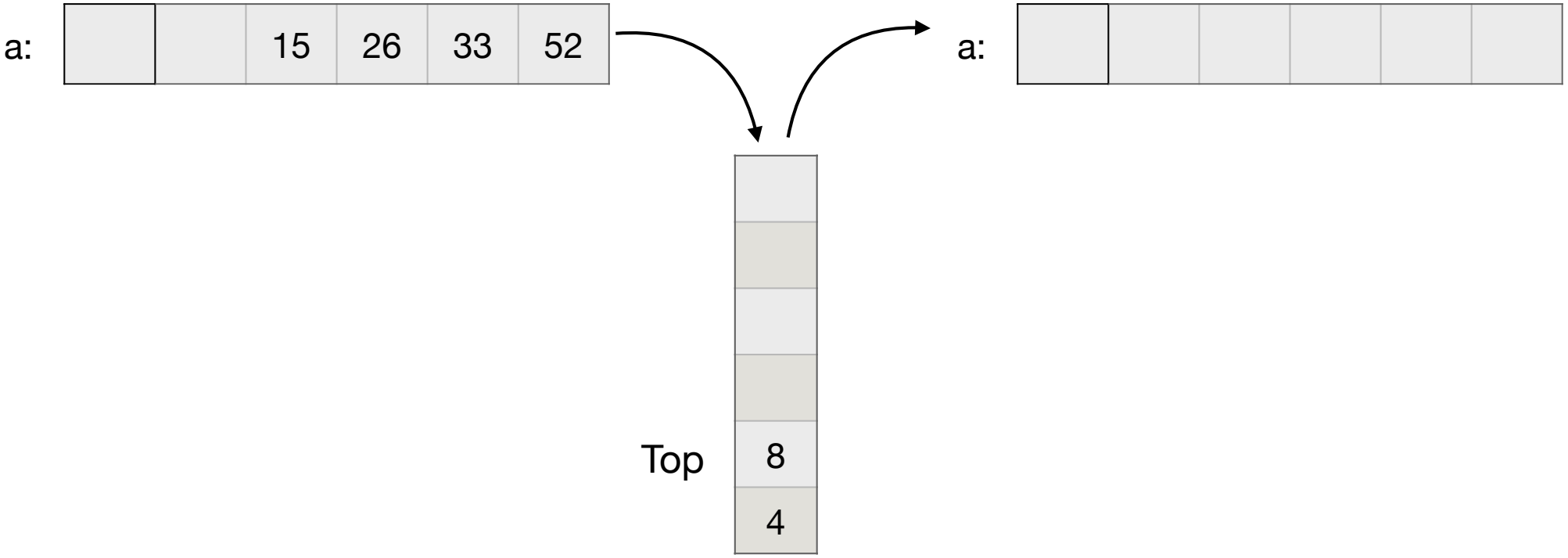
Renverser les éléments d'un `Array`



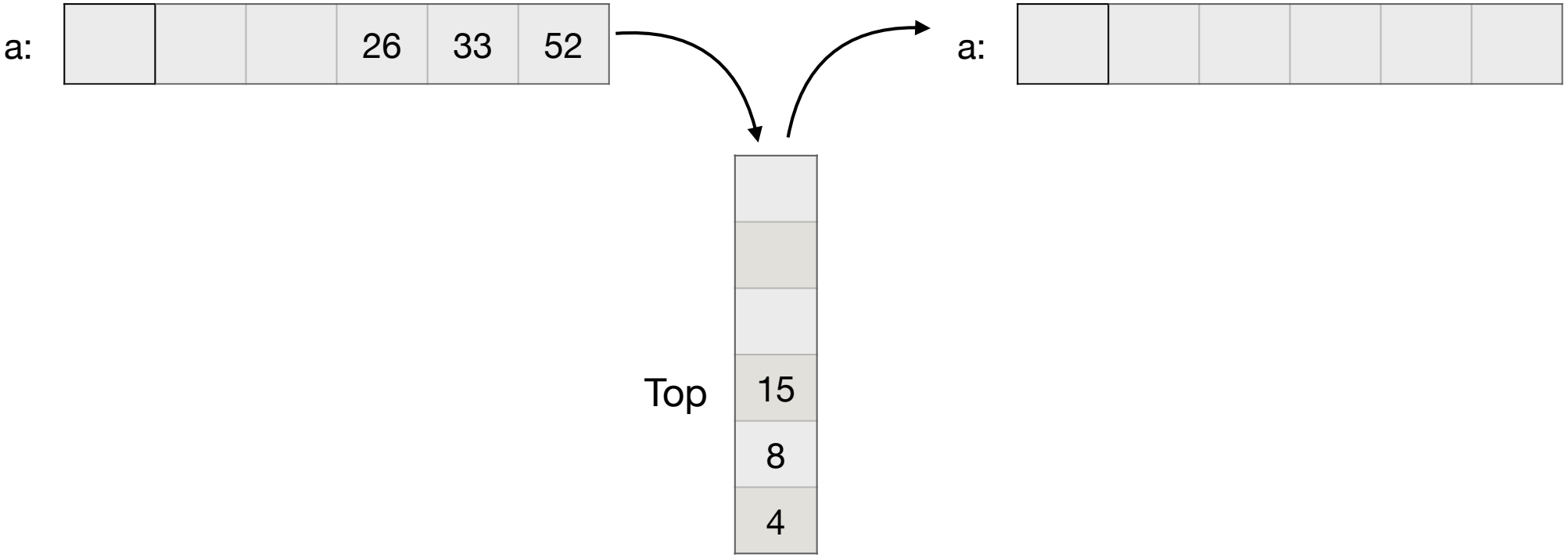
Renverser les éléments d'un `Array`



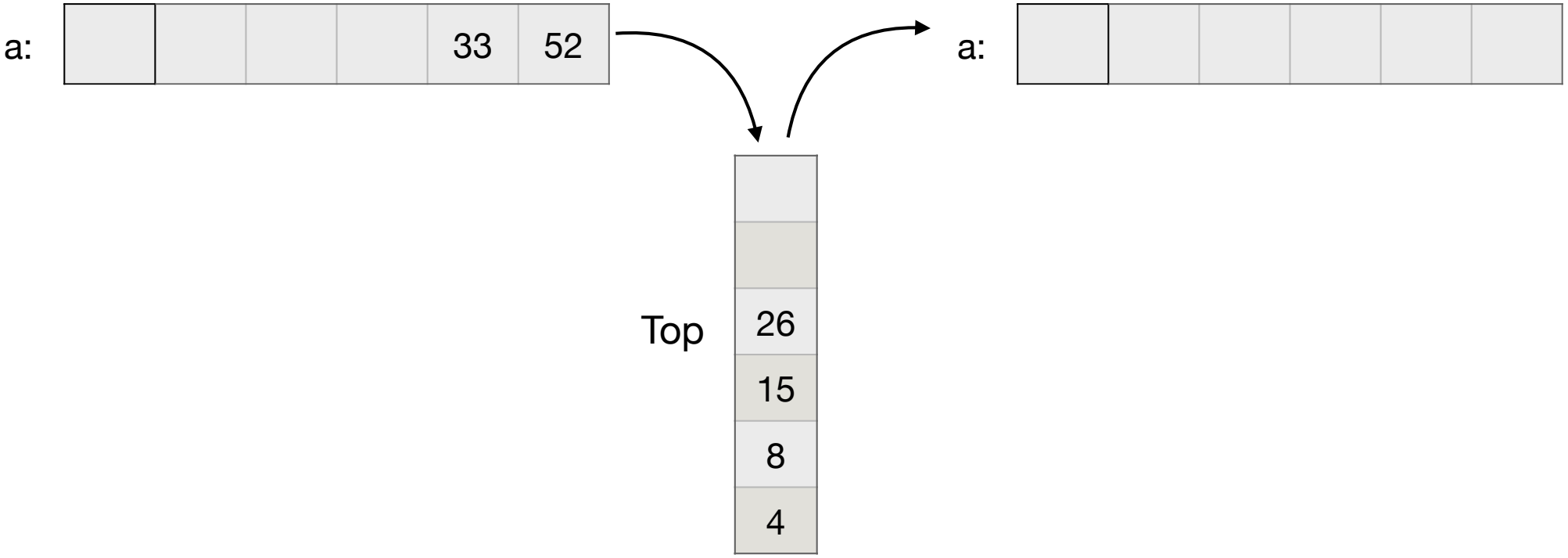
Renverser les éléments d'un `Array`



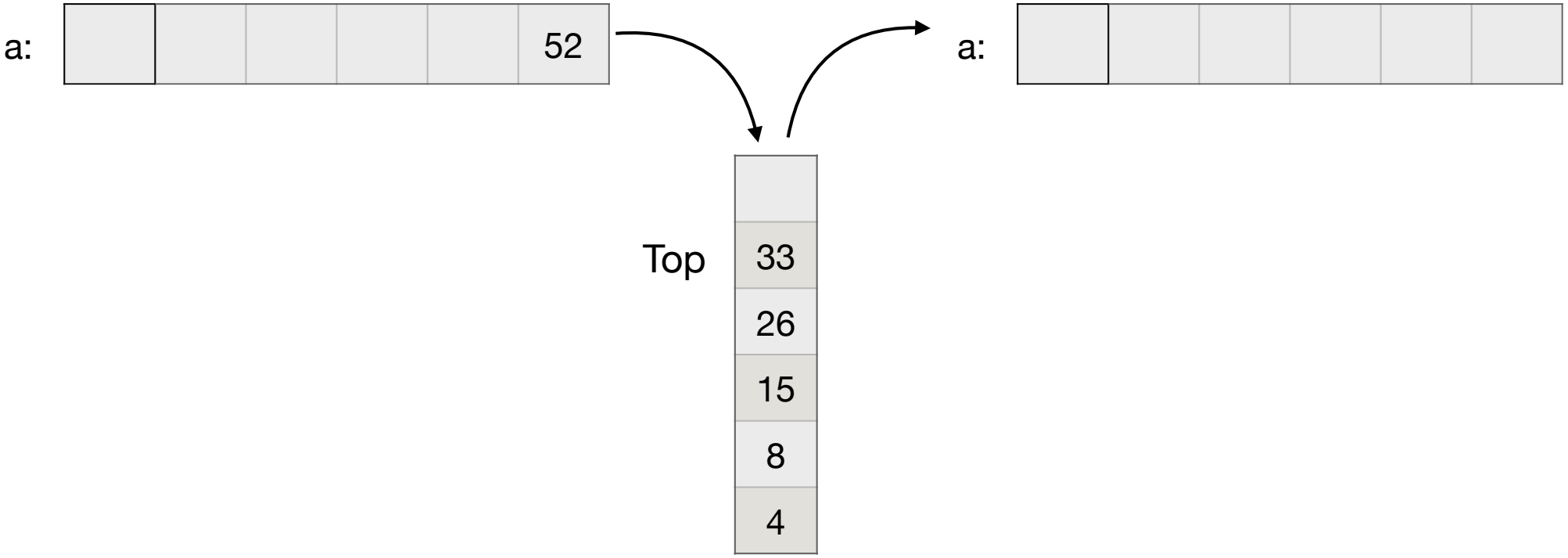
Renverser les éléments d'un `Array`



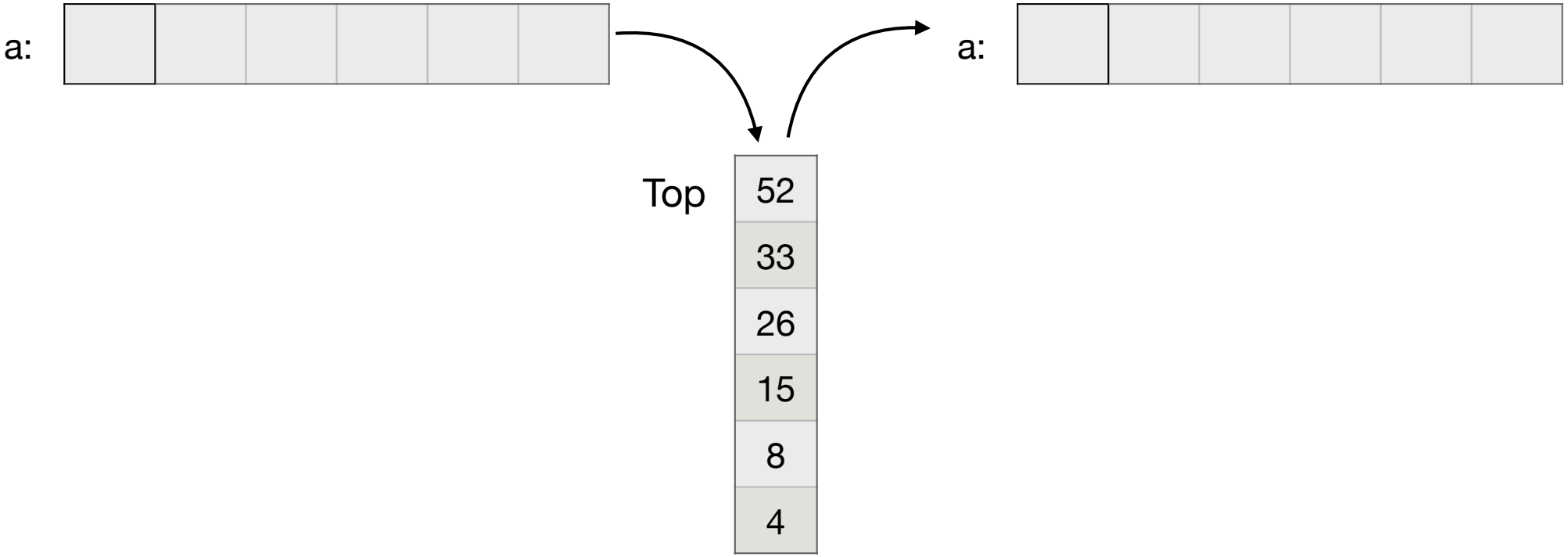
Renverser les éléments d'un `Array`



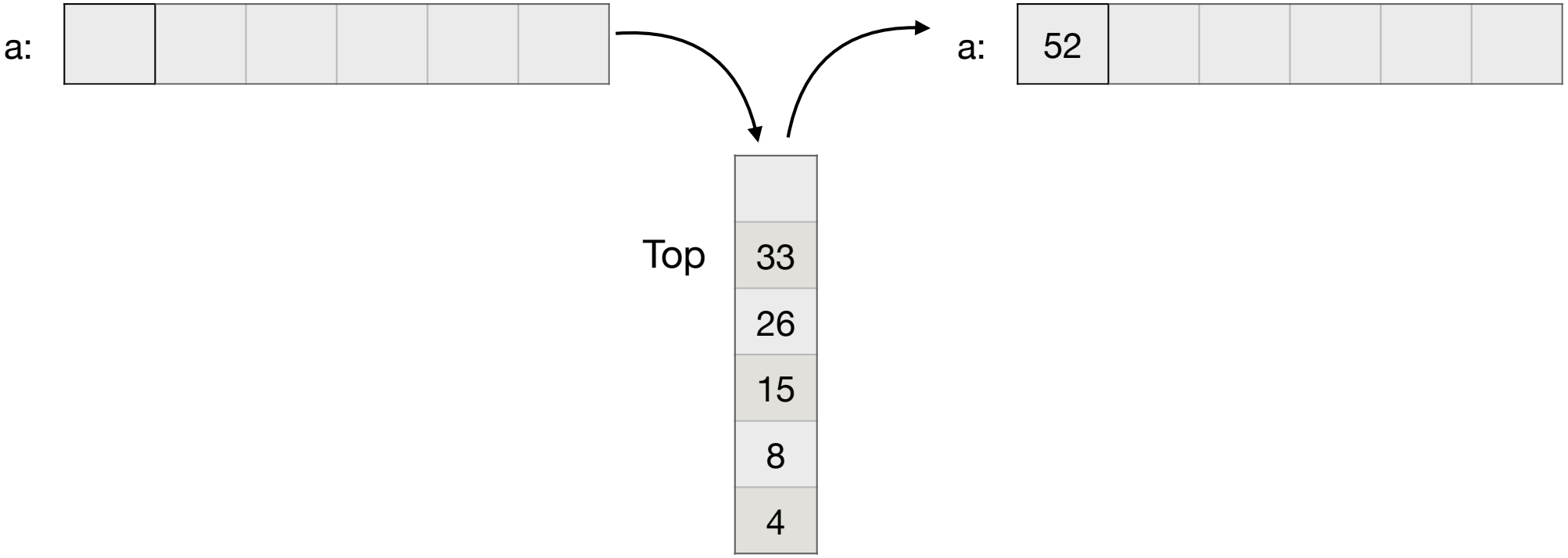
Renverser les éléments d'un `Array`



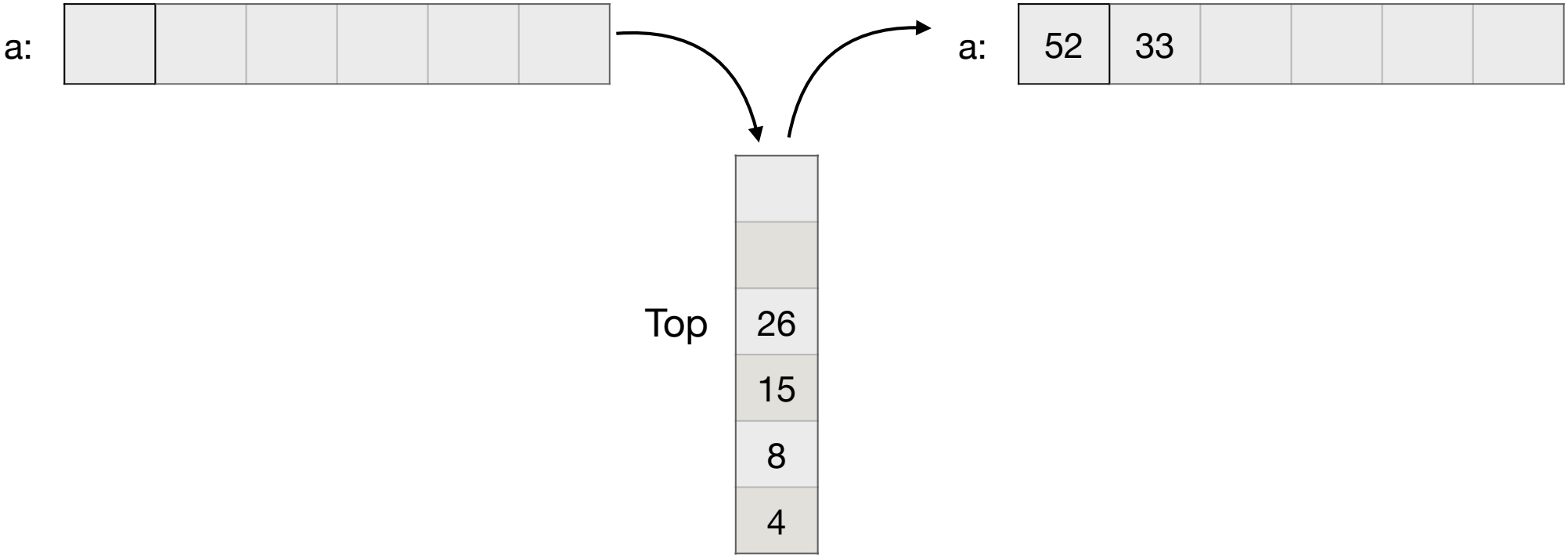
Renverser les éléments d'un `Array`



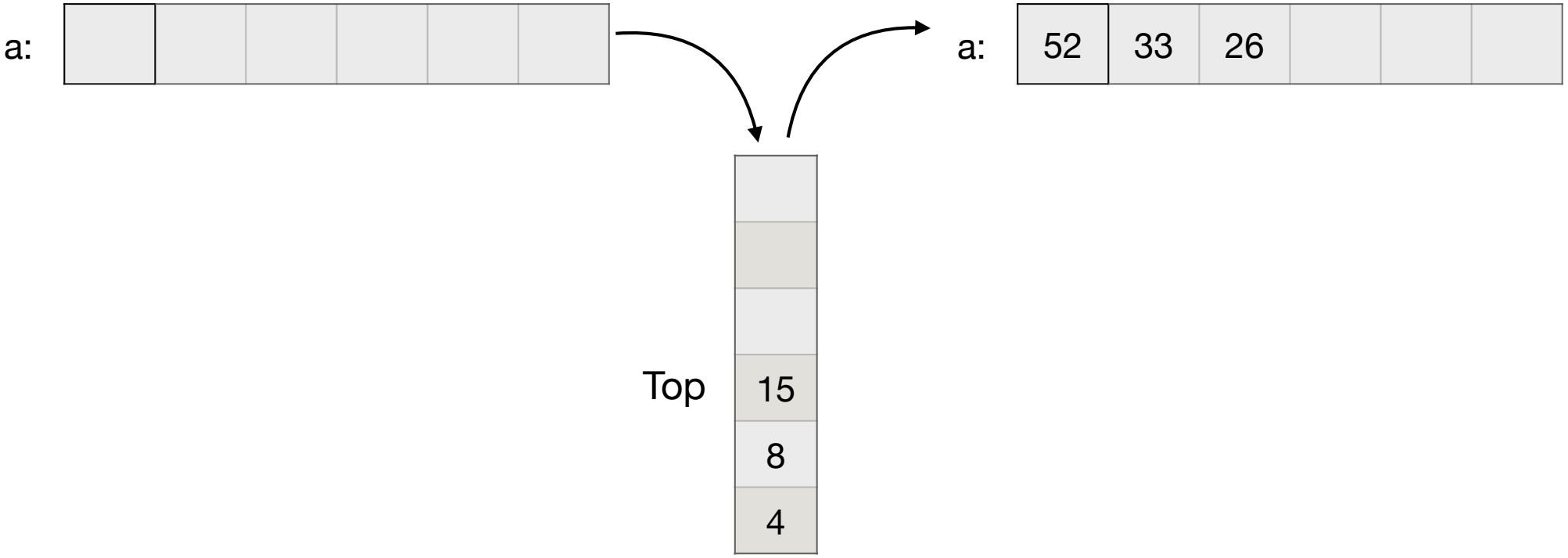
Renverser les éléments d'un `Array`



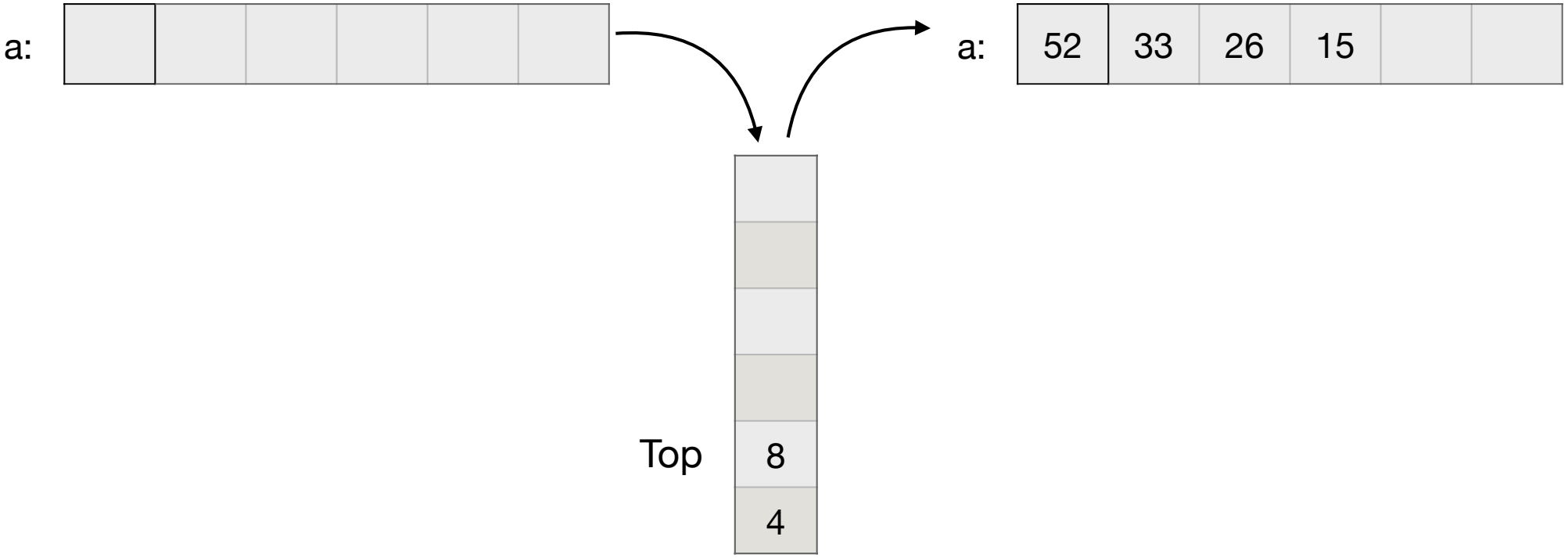
Renverser les éléments d'un `Array`



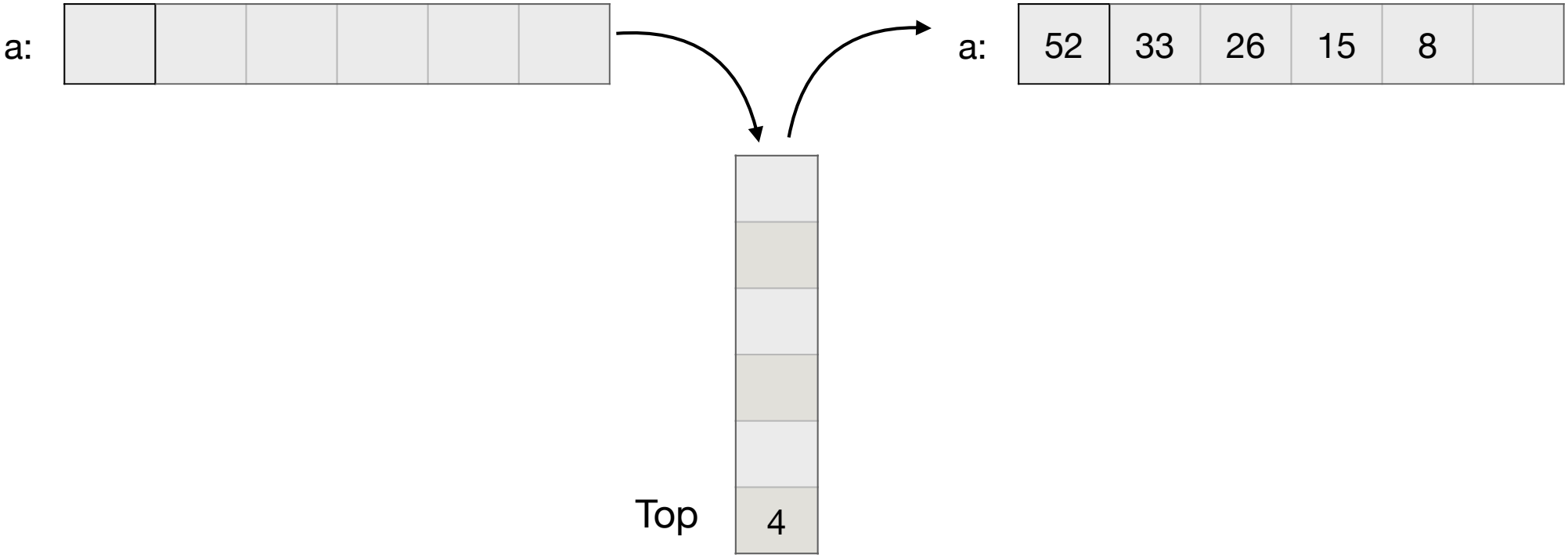
Renverser les éléments d'un `Array`



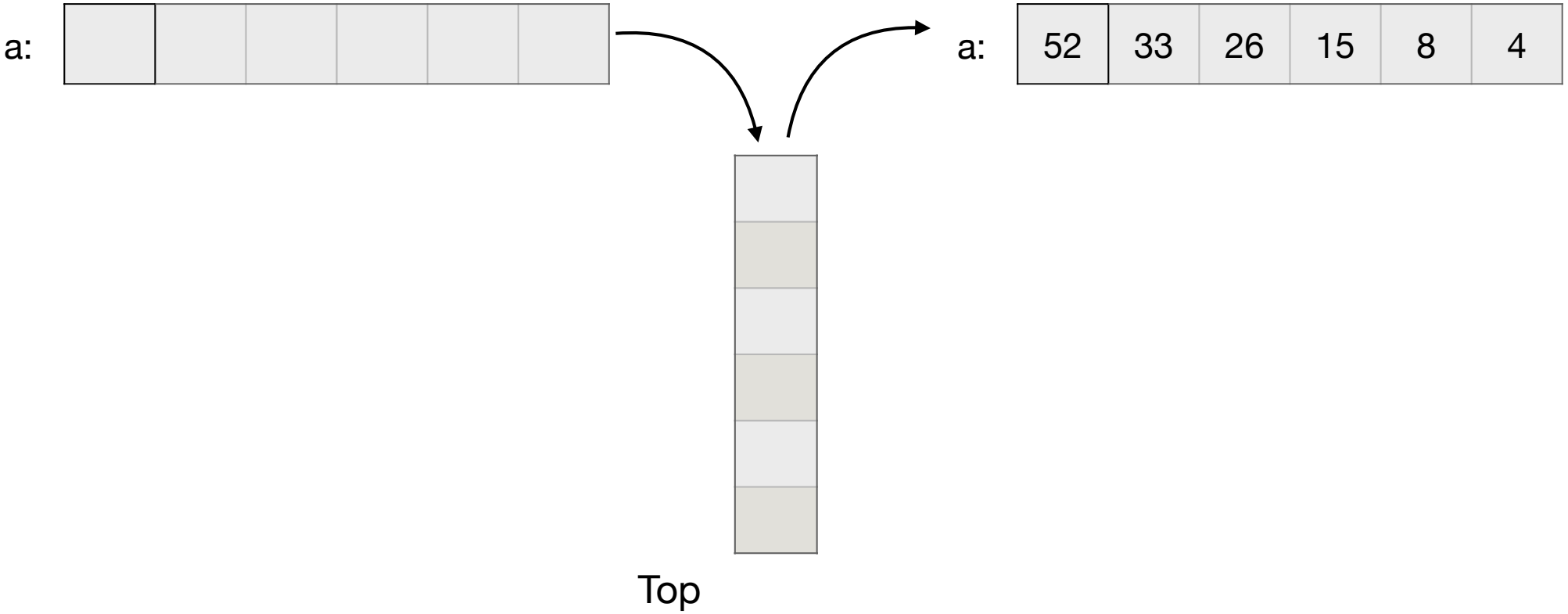
Renverser les éléments d'un `Array`



Renverser les éléments d'un `Array`



Renverser les éléments d'un `Array`



Voir le code...

`StackApp.java`

Une application: le balancement de parenthèses/délémiteurs

Chaque "(", "{", ou "[" doit être associé à ")", "}" ou “]” :

[(7 - x) + (y + z)]

correct : ( ) (( )) { ([( )]) }
correct : ((( ) (( )) { ([( )]) }))

Incorrect :) (( )) { ([( )]) }
Incorrect : ({ [ ]) }
Incorrect : (

Voir le code...

`StackApp.java`

Une application: le balancement de “tags” HTML

```
<body>
<center>
<h1> The Little Boat </h1> </center>
<p> The storm tossed the little boat like a cheap
sneaker in an old washing machine. The three drunken
fishermen were used to such treatment, of course, but
not the tree salesman, who even as a stowaway now
felt that he had overpaid for the voyage. </p>
<ol>
<li> Will the salesman die? </li>
<li> What color is the boat? </li>
<li> And what about Naomi? </li>
</ol>
</body>
```

The Little Boat

The storm tossed the little boat like a cheap sneaker in an old washing machine. The three drunken fishermen were used to such treatment, of course, but not the tree salesman, who even as a stowaway now felt that he had overpaid for the voyage.

1. Will the salesman die?
2. What color is the boat?
3. And what about Naomi?

Voir le code...

`StackApp.java`

Remarques conclusives

- On a regardé ce qu'est une pile et quelques applications.
- On a définit l'ADT avec une interface `Stack`
- On a regardé les différences avec `java.util.Stack`
- On a implémenté l'interface `Stack` avec un `Array` et avec une `SinglyLinkedList`
- Toutes les opérations s'exécutent en $O(1)$
- On a regardé des applications : Renverser un `Array` et balancer des parenthèses et "tags" HTML.