Tri d'un tableau de chaînes de caractères

Le tableau de chaînes de caractères :

	0	1	2	3	4
mots	Python	Java	Ada	C	Perl

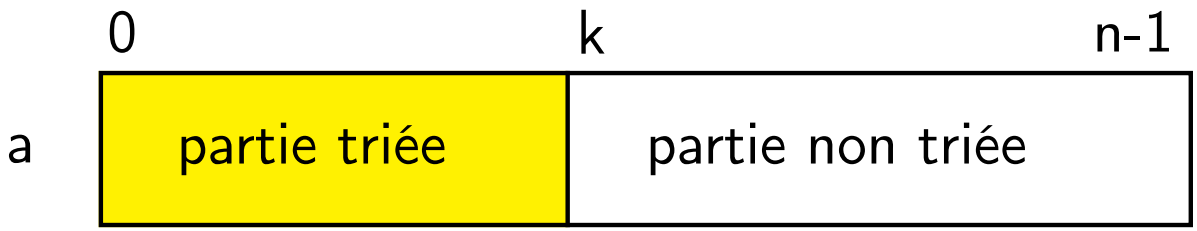
après le tri le tableau précédent devient :

	0	1	2	3	4
mots	Ada	C	Java	Perl	Python

Tri par sélection

Soit **a** un tableau et **n** son nombre d'éléments.

n est égal à **a.length** $-$ **1**



A l'étape **k** on recherche (séléctionne) le plus petit élément de la partie non triéé et on l'échange avec le premier élément (d'indice k) de cette partie non triée.

Tri par sélection (suite)

L'algorithme du tri par séléction est donc :

pour i **de** 0 **à** n - 2 **faire**

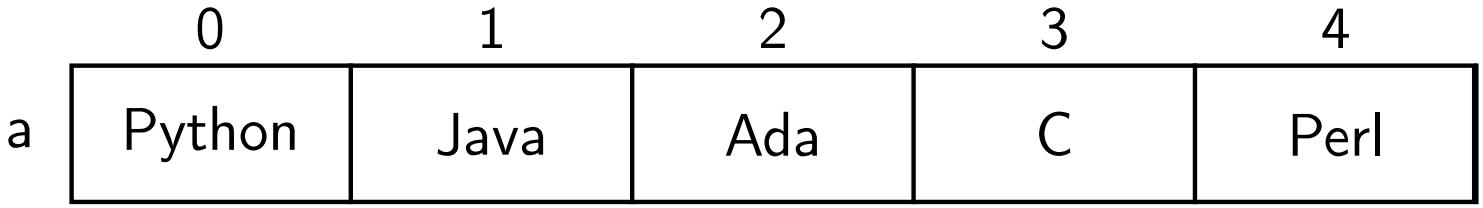
// déterminer l'indice **iMin** du plus petit élément
// de **a [k : a . length - 1]**

// échanger les éléments d'indice **k** et **iMin**

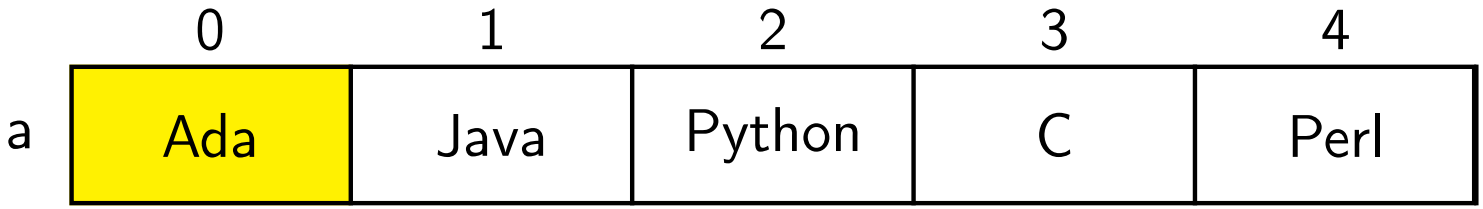
fpour

Tri par sélection (suite)

Pour le tableau a



à l'étape **k=0** on trouve **iMin=2**, le tableau a devient :



Tri par sélection (suite)

	0	1	2	3	4
a	Ada	Java	Python	C	Perl

à l'étape **k=1** on trouve **iMin=3**, le tableau a devient :

	0	1	2	3	4
a	Ada	C	Python	Java	Perl

à l'étape **k=2** on trouve **iMin=3**, le tableau a devient :

	0	1	2	3	4
a	Ada	C	Java	Python	Perl

Tri par sélection (suite)

	0	1	2	3	4
a	Ada	C	Java	Python	Perl

à l'étape **k=3** on trouve **iMin=4**, le tableau a devient :

	0	1	2	3	4
a	Ada	C	Java	Perl	Python

Il ne reste plus qu'un élément supérieur à tous ceux qui le précèdent, le tableau **a** est donc complètement trié.

	0	1	2	3	4
a	Ada	C	Java	Perl	Python

Execice 1

Ecrire la fonction **tri** qui permet de trier par ordre croissant le tableau **a** de chaînes de caractères passé en paramètre.

Cette fonction doit utiliser les deux fonctions **indMin** et **echange**

Excercice 1 (suite)

```
package tp12 ;

public class Tri {
    public static void echange ( String [ ] a, int i1 , int i2 ) {

    }

    public static int indMin ( String [ ] a , int iDeb ) {

    }

    public static void tri ( String [ ] a ) {

    }

    public static void main ( String [ ] args ) {
        String [ ] mots ;
        mots = new String [ ]
                    { "Python" ,"Java" ,"Ada" , "C" , "Perl" } ;

        Tableaux.afficher( mots );
        tri ( mots ) ;
        Tableaux.afficher( mots );
    }
}
```

Egalité de deux chaînes de caractères

Il n'est pas possible d'utiliser les opérateurs d'égalité ou d'inégalité avec les chaînes de caractères
L'expression

```
s1 . equals ( s2 )
```

retourne vrai si les deux chaînes sont égales et faux dans le cas contraire.

Egalité de deux chaînes de caractères (suite)

```
    ...
s1 = "ABC" ;
s2 = "abc" ;
System.out.println( s1.equals ( s2 ) ) ;
  // -> false
System.out.println( s1.equalsIgnoreCase ( s2 ) ) ;
  // -> true
```

Ordre lexicographique

La chaîne **s1** précède la chaîne **s2** (**s1** $\prec$ **s2**) **ssi**

- **s1** est un préfixe de **s2**

  bal $\prec$ ballon

- Le premier caractère de **s1** qui ne coïncide pas avec le caractère de même range de **s2** est inférieur à ce dernier

  ballon $\prec$ bas

Comparaison de deux chaînes de caractères

L'expression

```
s1 . compareTo ( s2 )
```

$$\text{retourne une valeur} \begin{cases} < 0 & \text{si} \quad s1 \prec s2 \\ 0 & \text{si} \quad s1 \text{ est égal à } s2 \\ > 0 & \text{si} \quad s1 \succ s2 \end{cases}$$

L'expression

```
s1 . compareToIgnoreCase ( s2 )
```

effectue la même comparaison sans tenir compte des majuscules et minuscules.

Recherche séquentielle d'une valeur dans un tableau non rempli

Ecrire la fonction **appartient** qui admet trois paramètres

- **a** un tableau de chaînes de caractèes dans lequel s'effectue la recherche
- **nbElts** entier égal au nombre d'éléments du tableau précédent
- **valRech** chaîne de caractères valeur recherchée dans le tableau

et qui retourne **vrai** si la valeur recherchée **valRech** appartient au tableau **a** et **faux** dans le cas contaire.

La fonction appartient

```
public static boolean appartient( String [ ] a, int nbElts, String valRech) {


}
```

Elimination des doublons

```
public static int elimDoublons ( String [ ] a, int nbElts ) {


}
```