ArrayList en JAVA

ArrayList en JAVA

- En Java, et d'une façon plus général dans les langages de programmation, pour utiliser un tableau, il faut le déclarer et fixer sa taille lors de la déclaration. Par exemple.

- int[] tab = new int[5];

- Une fois que le tableau est déclaré, il est difficile de le redimensionne

ArrayList en JAVA

- En Java, et d'une façon plus général dans les langages de programmation, pour utiliser un tableau, il faut le déclarer et fixer sa taille lors de la déclaration. Par exemple.

- int[] tab = new int[5];

- Une fois que le tableau est déclaré, il est difficile de le redimensionne

ArrayList en JAVA

Avantage:

- L'avantage des tableaux est qu'ils permettent de gérer plusieurs valeur de même type dans une même variable.

Inconvénient:

- Une fois qu'on a définit un tableau sur un nombre de dimension, il est assez pénible de le modifier tout au long de vos programmes.

ArrayList en JAVA

- C'est pourquoi pour résoudre ce problème, la classe **ArrayList** de Java a été mise sur point. Elle permet de créer des tableaux redimensionnables.

- Contrairement aux tableaux, les ArrayList peuvent **ajuster** automatiquement leur capacité lorsque vous y ajoutez ou supprimez des éléments. C'est pourquoi on les appelle aussi les **tableaux dynamiques**.

ArrayList en JAVA

Création d'une ArrayList:

```
ArrayList<Type> maListe= new ArrayList<>();
```

ArrayList en JAVA

Création d'une ArrayList:

```
ArrayList<Int> nombres= new ArrayList<>();
 ArrayList<String> mots= new ArrayList<>();
```

ArrayList en JAVA

Création d'une ArrayList:

- Voici un exemple complet qui illustre l'utilisation de l'ArrayList :

ArrayList en JAVA

Création d'une ArrayList:

```
public class Programme { public static void main(String[] args)
{
// création de l'ArrayList
ArrayList<String> jours = new ArrayList<>();
// Ajout d'éléments dans l'ArrayList
jours.add("Lundi"); jours.add("Mardi"); jours.add("Mercredi");
jours.add("Jeudi"); jours.add("Vendredi"); jours.add("Samedi");
jours.add("Dimanche");
 System.out.println("Les jours de la semaine: " + jours);
} }
```

ArrayList en JAVA

Opérations de base sur les ArrayList:

- Voici quelques méthodes efficaces pour manipuler la collection ArrayList :
- **add(E e)** : permet d'ajouter un élément à un arraylist.
- **get(int index)** : retourne l'élément à la position index de la liste.
- **set(int index, E e)** : remplace l'élément à la position index par l'élément e .
- **remove(int index)** : supprime l'élément à la position index.

Exceptions en JAVA

```
// création de l'ArrayList
ArrayList<String> fruits = new ArrayList<>();
// Ajout d'éléments dans l'ArrayList
fruits.add("apple");
fruits.add("banana");
fruits.add("orange");
System.out.println(fruits);
fruits.add("ananas");
System.out.println(fruits);
fruits.set(3, "avocat");
 System.out.println(fruits);
 fruits.remove(3);
```

ArrayList en JAVA

Parcourir une ArrayList:

```
for(String fruit: fruits)
{
        System.out.print(fruit + " , ");
}
```

ArrayList en JAVA

- Vous pouvez aussi effectuer facilement des tris de tableau. Pour ce faire, utilisez la méthode **sort()** de Collections:

ArrayList en JAVA

```
public static void main(String[] args){
// création de l'ArrayList
 ArrayList<String> outils = new ArrayList<>();
// Ajout d'éléments dans l'ArrayList
outils.add("hadoop"); outils.add("Scala"); outils.add("Spark");
System.out.println("ArrayList avant trie :"+ outils);
//affiche
//ArrayList avant trie :[hadoop, Scala, Spark]
//Trie
Collections.sort(outils);
System.out.println("ArrayList après trie :"+ outils);
//affiche //ArrayList après trie:[Scala, Spark, hadoop]
```