

Exercice K

Le but de cet exercice est de créer une simple classe qui permet de gérer des noms d'élèves.

| StudentPicker | |
|---------------|---|
| – | alNames : ArrayList<String> |
| + | addName(pName : String) : void |
| + | getName(pIndex : int) : String |
| + | setName(pIndex : int, pNewName : String) : void |
| + | getNumberOfNames() : int |
| + | removeNameAtIndex(pIndex : int) : void |
| + | removeAllNames() : void |
| + | getRandomName() : String |
| + | displayAllNames() : void |
| + | toString() : String |

Cet exercice sera réalisé entièrement dans Unimozer. Procédez comme suit :

1. Ouvrez le projet **ExerciceK_M**
2. Compilez le projet et créez un nouvel objet de la classe `StudentPicker`.
3. Appelez la méthode `addName` pour ajouter le nom **Olivier** à la liste.
4. Appelez la méthode `addName` pour ajouter un deuxième nom **Antoine** à la liste.
5. Appelez maintenant la méthode `getName` et essayez d'obtenir le premier nom de la liste. Quel index (=position) faut-il passer en paramètre ?
Effectuez un clic droit sur l'objet, choisissez **Inspect**, puis sélectionnez la liste pour voir les index des noms.

The top screenshot shows the **studentPicker0 : StudentPicker** object in the workspace. The **alNames** field is highlighted, showing its value as `[Olivier]`. An **Inspect** window is open, displaying the object's methods: `void addName(String)` and `String getName(int)`, along with an **Remove** option.

The bottom screenshot shows the **studentPicker0 : StudentPicker** object with the **alNames : java.util.ArrayList** field selected. A red arrow points to the **Index** column in the **Monitor** window, which displays the following data:

| Index | Value |
|-------|---------|
| 0 | Olivier |
| 1 | Antoine |

Below the table is a **Monitor** button. A red text box on the left asks: "Quel est le premier index?" (What is the first index?).

6. Ajoutez une méthode `setName` qui accepte deux paramètres, un index `pIndex` à valeur entière et une chaîne de caractères `pNewName`. La méthode remplace le nom à l'index fourni par le nouveau nom `pNewName`.
7. Ajoutez une méthode `removeNameAtIndex` qui accepte un index en paramètre et supprime le nom correspondant.
8. Ajoutez une méthode `getNumberOfNames` qui retourne le nombre de noms dans la liste.
9. Ajoutez une méthode `getRandomName` qui retourne un des noms de la liste. Ce nom est choisi aléatoirement. Que se passe-t-il si la liste est vide ? Comment peut-on résoudre ce problème ?
10. Ajoutez une méthode `removeAllNames` qui supprime tous les noms de la liste.
11. Ajoutez une méthode `displayAllNames` qui affiche tous les noms dans la fenêtre des messages.
12. Ajoutez une méthode `toString` qui tous les noms de la liste sous forme d'une chaîne de caractères. Les noms sont séparés par des espaces. Utilisez une boucle `for`.

Voici un aperçu des méthodes publiques les plus importantes de la classe `ArrayList` :

| Méthodes publiques | Description |
|--|--|
| <code>boolean add(Object pObj)</code> | Ajouter un nouvel objet à la liste. La valeur retournée est toujours <code>true</code> , mais on ne l'utilisera jamais. |
| <code>Object get (int pIndex)</code> | Retourner l'objet à l'index spécifié en paramètre. |
| <code>Object set(int pIndex, Object pObj)</code> | Remplacer l'objet à l'index spécifié <code>pIndex</code> en paramètre par <code>pObj</code> . La méthode retourne l'objet qui a été remplacé. |
| <code>Object remove(int pIndex)</code> | Enlever l'objet à l'index <code>pIndex</code> La méthode retourne l'objet qui vient d'être enlevé. |
| <code>void clear()</code> | Enlever tous les objets de la liste |
| <code>int size()</code> | Retourner le nombre d'objets qui se trouvent dans la liste |
| <code>boolean isEmpty()</code> | Vérifier si la liste est vide. La méthode retourne <code>true</code> si la liste est vide, <code>false</code> sinon. |