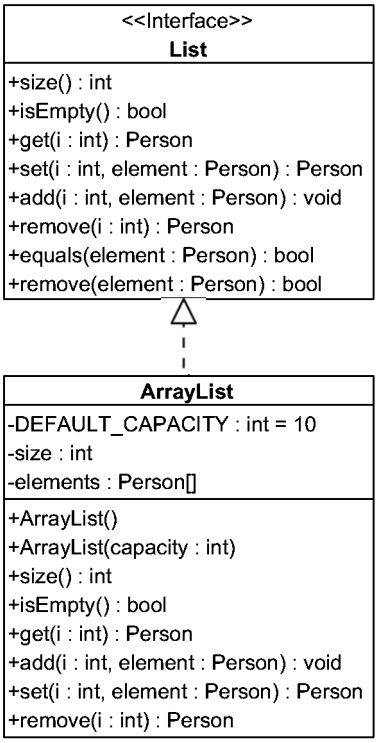
Série 3

Cette série met en application la structure de données « List » et met en pratique les interfaces et classes abstraites. Cette série est construite sur la série précédente. Pour commencer, vous devez utiliser le projet SDA_Serie3_BASE.

1. Veuillez analyser le code de l'interface « List » et de son implémentation « ArrayList »:



Méthode	Description
size( )	Retourne le nombre d'éléments stockés dans la liste.
isEmpty( )	Retourne un booléen qui indique si la liste est vide.
get(i)	Retourne l'élément à l'indice i.
set(i, person)	Remplace l'élément à l'indice i par une instance de Person et renvoie l'élément remplacé.
add(i, person)	Insère un élément de type Person à l'indice i, en décalant les éléments suivants d'un indice vers la droite.
remove(i)	Supprime et renvoie l'élément à l'indice i, en décalant les éléments suivants d'un indice vers la gauche.

2. Quels sont les principes de POO appliqués à cette structure de données ? Veuillez développer en vous référant au code source si besoin.

3. Veuillez effectuer la trace de la méthode *add(0, Bill)* lorsque l'état de l'ArrayList est le suivant : {Elon; Larry; Mark}. Bill, Elon, Larry et Mark sont des personnes (ou d'un type qui en hérite de Person).

Méthodes supplémentaires

Notre *ArrayList* permet notamment d'ajouter et de supprimer des éléments avec les méthodes ci-dessous. Toutefois, on doit toujours gérer les indices.

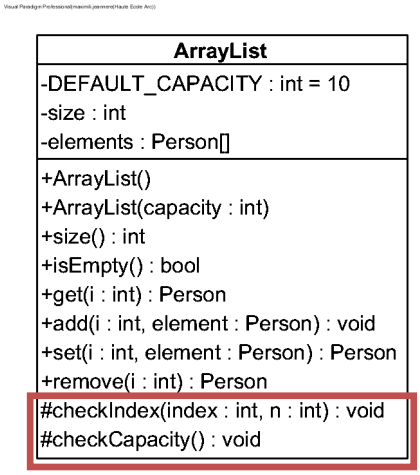
1. Que proposez-vous pour ne plus avoir à gérer cet indice lors de l'insertion ?

2. Veuillez alors implémenter les méthodes suivantes :

Modifier et type	**Méthode et description**
void	*add(E element)* Ajoute un élément à la fin de la liste.
E	*removeLast()* Supprime et renvoie l'élément situé à la dernière case (dernier indice).

3. Veuillez schématiser l'énoncé suivant : *une ArrayList offre une abstraction au tableau.*

Des méthodes de vérification (encadrées en rouge) ont été ajoutées pour garantir le bon fonctionnement de l'ArrayList.



Méthode	**Description**
get(*i*)	… Une erreur survient si i n'est pas dans la plage [0, size () -1].
set(*i, person*)	… Une erreur survient si i n'est pas dans la plage [0, size () -1].
add(*i, person*)	… Une erreur survient si i n'est pas dans la plage [0, size ()].
remove(*i*)	… Une erreur survient si i n'est pas dans la plage [0, size () -1].
checkIndex(i)	Vérifie si l'indice i donné est dans la plage [0, n].
checkCapacity()	Vérifie si le tableau à de la place disponible.

4. Depuis la classe contenant la méthode main(…), veuillez attraper une exception de type *IllegalStateException* ou *IndexOutOfBoundsException.*

5. Pour le moment, notre « ArrayList » peut stocker uniquement des éléments de type Person.
   a. Est-ce qu'il s'agit d'une limite ?
   b. Que proposez-vous pour stocker n'importe quel type de données ?

Exercice facultatif : ArrayList dynamique

Pour le moment, notre ArrayList peut contenir un nombre fini d'éléments. Veuillez implémenter une ArrayList dynamique. C'est-à-dire, lorsque la capacité est pleine il faut pouvoir agrandir la liste afin d'y insérer un nouvel élément.