

R1.01

INITIATION AU DÉVELOPPEMENT

Cours 5, partie 1 : – Notion de vecteurs – Classe ArrayList

Hervé Blanchon & Anne Lejeune

Université Grenoble Alpes

IUT 2 – Département Informatique

Sommaire

 Notion de vecteur

 Vecteur en Java

 choix de la classe `ArrayList<E>`

 Approfondissement

 déclaration et la construction d'un `ArrayList<E>`

 Insertion(s) dans un `ArrayList<E>`

NOTION DE VECTEUR

Un vecteur ?


Une **collection d'éléments**

 ... de **même type**

 simple (*ex. un vecteur d'entiers*)

 structuré (*ex. un vecteurs d'étudiants*)

 vecteur (*ex. un vecteur de vecteurs d'entiers*)

 ... **indicés** (numérotés)

Exemple


 Vecteur **v** d'**entiers** **ind****icés** de **0** à **5**

Exemple et notations

 Vecteur **v** d'**entiers** **ind****icés** de **0** à **5**

	0	1	2	3	4	5
V	5	12	7	10	6	8

Notations

 $V[0 \dots 5] = [5, 12, 7, 10, 6, 8]$

 $[0 \dots 5]$ est l'**intervalle des indices** de **V**

 **0** est la **borne inférieure** (*plus petit des indices*)

 **5** est la **borne supérieure** (*plus grand des indices*)

 $V[i]$ est l'élément d'indice **i** du vecteur **V** (*se lit « v de i »*)

 $V[0] = 5$; $V[3] = 10$; $V[5] = 8$

 $V[i]$ n'est défini que si **i** est **dans l'intervalle des indices**

 $V[6]$ **n'est pas définis !!**

Nombre d'éléments d'un vecteur et vecteur vide

 Un vecteur est donc défini sur un intervalle d'indice [$\text{inf}..\text{sup}$]

 $v[\text{inf} .. \text{sup}]$


 Lorsque $\text{inf} \leq \text{sup}$, le vecteur n'est pas vide

 si $\text{inf} = \text{sup}$: le vecteur contient exactement un élément

 si $\text{inf} < \text{sup}$: le vecteur contient exactement $(\text{sup} - \text{inf}) + 1$ éléments

 Exemples

 $v[0 .. 0]$ contient un élément


 $v[0 .. 11]$ contient 12 éléments $\{(\text{11} - 0) + 1\}$

 Lorsque $\text{inf} > \text{sup}$, on dit que le vecteur est vide

 la borne inférieure des indices est strictement supérieure à la borne supérieure

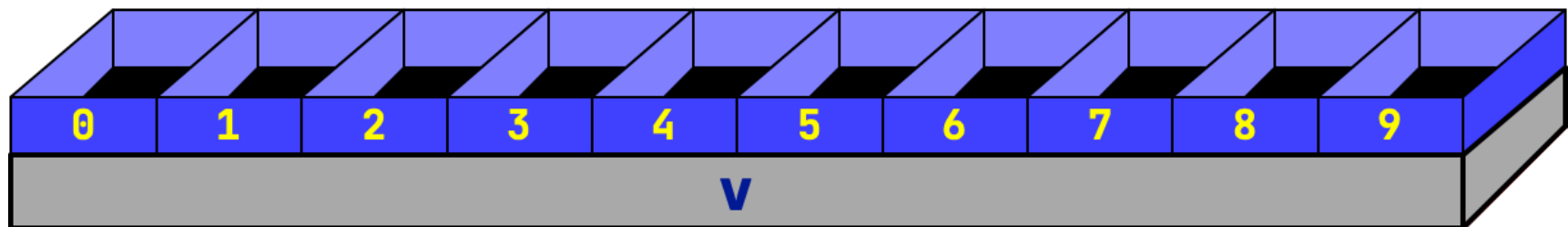
 Exemples

 $v[1 .. 0]$

 $v[17 .. 16]$

Représentation d'un vecteur en mémoire

- Pour l'accès à l'élément à l'indice i ($v[i]$) il faut pouvoir calculer la position de cet élément dans la mémoire
- Dans un vecteur les éléments sont rangés de manière contigüe (soit les uns à la suite des autres)
- Un vecteur peut être vu comme une grande boîte qui contient des petites boîtes (les éléments)
- Sur le schéma suivant v est un vecteur qui contient 10 éléments indicés de **0** à **9**



VECTEUR EN JAVA


Classe ArrayList

 Java propose deux classes `Vector<E>` et `ArrayList<E>` pour implanter des vecteurs

 *Nous allons utiliser la classe `ArrayList<E>`*

 La classe `ArrayList` est une **classe générique** (`<E>`)

 `public class ArrayList<E>`

 `E` est une classe qui détermine de type des objets contenus dans un objet **`ArrayList`**



un vecteur de la classe `ArrayList` ne peut contenir que des objets, instances de la classe `E` (d'où les classes enveloppes ! si on a besoin d'un vecteur d'entiers par exemple)

 Le premier indice pour repérer les éléments est `0` (zéro)

Construction d'un ArrayList

 La classe `ArrayList` propose 3 **constructeurs** dont deux nous intéressent dans ce cours :

`ArrayList()`

Construit un `ArrayList` vide avec une capacité initiale de dix.

`ArrayList(int initialCapacity)`

Construit un `ArrayList` vide avec la capacité initiale spécifiée.

 La capacité de l'`ArrayList` est le nombre d'éléments qu'il peut contenir sans « déborder »

 le débordement est autorisé (voir plus loin)

Déclaration et construction d'un ArrayList

Exemples

 un ArrayList de String

 `ArrayList<String> vecteurDeChaines = new ArrayList<>();`

 un ArrayList de Integer

 `ArrayList<Integer> vecteurDEntiers = new ArrayList<>();`

 un ArrayList de Covoiturage

 `ArrayList<Covoiturage> vecteurDeChaines = new ArrayList<>();`

Contre exemple

 un ArrayList de `int`

 `ArrayList<int> vecteurDeInt = new ArrayList<>();`

int est type primitif donc interdit !

Quelques méthodes de la classe ArrayList

Type du résultat	Méthode documentation <i>condition nécessaire à une exécution normale</i>
int	size() Retourne le nombre d'éléments dans cet ArrayList.
E	get(int index) Retourne l'élément à la position spécifiée dans cet ArrayList. Il faut que index soit dans l'intervalle [0; size()-1]
boolean	add(E element) Ajoute l'élément spécifié à la fin de cet ArrayList. Retourne toujours true.
void	add(int index, E element) Insère l'élément spécifié à la position spécifiée dans cet ArrayList. Il faut que index soit dans l'intervalle [0; size())
E	remove(int index) Supprime l'élément à la position spécifiée dans cet ArrayList. Il faut que index soit dans l'intervalle [0; size()-1]
void	clear() Supprime tous les éléments de cet ArrayList.

NOTE : E est le type classe des éléments de cet ArrayList


À retenir !

 L'intervalle légal des indices d'un ArrayList

 **[0 .. size() - 1]**

 le premier élément est à l'indice **0**

 le dernier élément est à l'indice **size() - 1**

 Avant de pouvoir utiliser une méthode sur une **variable** ou un **attribut** de type ArrayList, il faut que la **variable** ou l'**attribut** ait été construit(e)

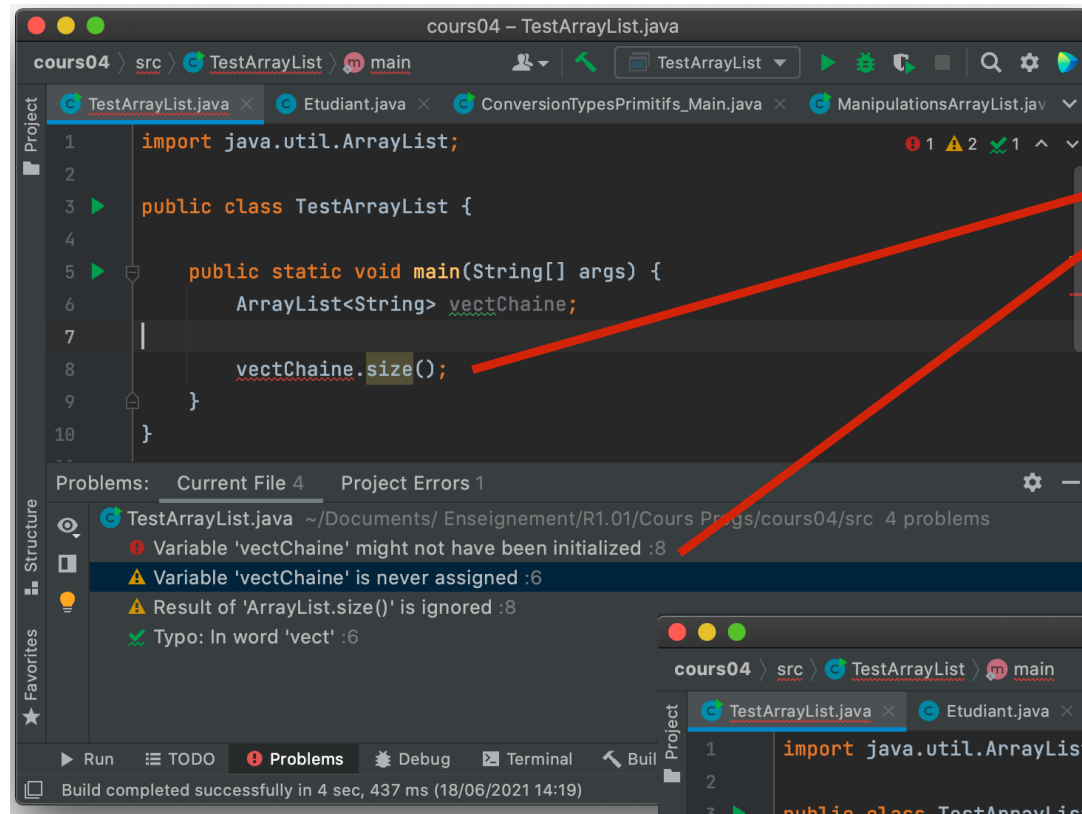
 Si **v** est un **vecteur** implanté sous forme d'ArrayList alors

 l'élément à la position **i** dans **v** se note **v[i]**

 **v[i]** contient un pointeur sur un objet

 on y accède en Java avec : **v.get(i)**

Utilisation d'un ArrayList non initialisé



The screenshot shows an IDE window titled 'cours04 - TestArrayList.java'. The code is as follows:

```
1 import java.util.ArrayList;
2
3 public class TestArrayList {
4
5     public static void main(String[] args) {
6         ArrayList<String> vectChaine;
7
8         vectChaine.size();
9     }
10 }
```

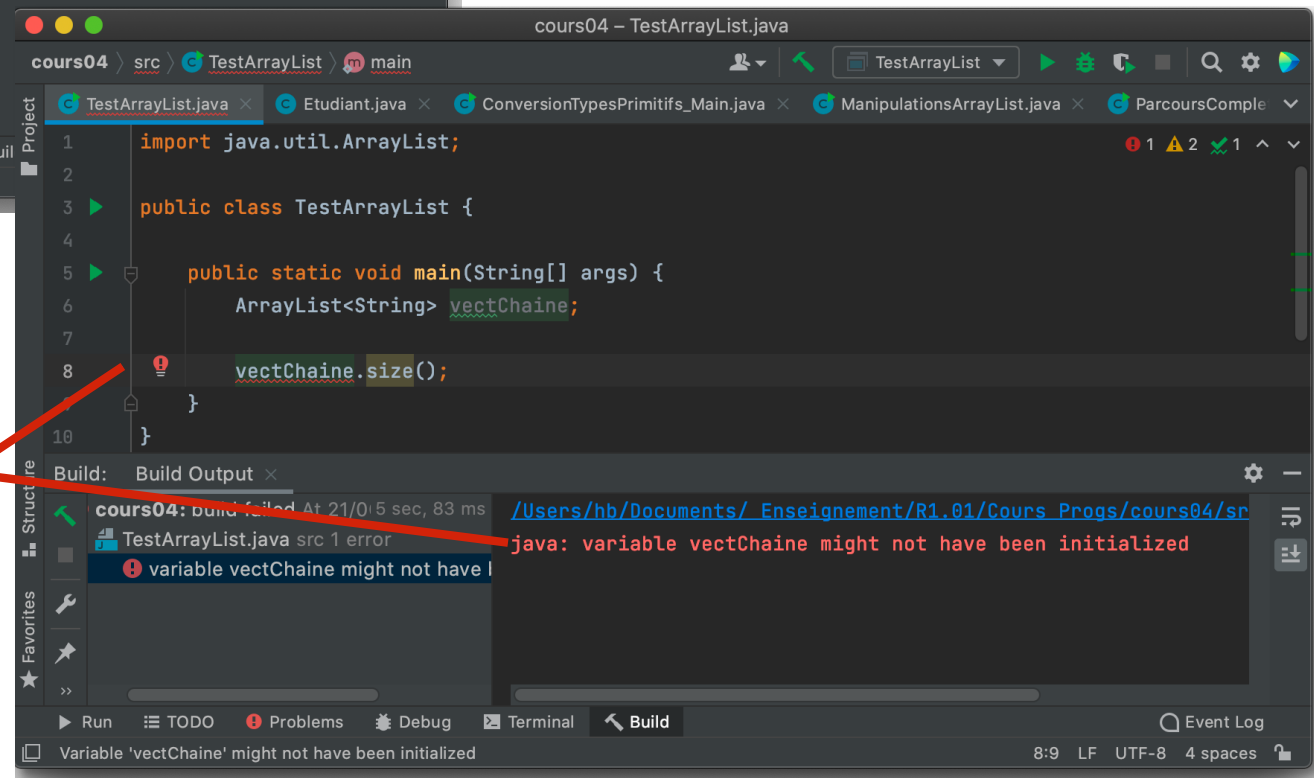
The 'Problems' panel at the bottom shows four warnings:

- Variable 'vectChaine' might not have been initialized :8
- Variable 'vectChaine' is never assigned :6
- Result of 'ArrayList.size()' is ignored :8
- Typo: In word 'vect' :6

Red arrows point from the first two warnings to the corresponding lines in the code.

Problème repéré dans l'examen du code

Problème repéré lors d'une tentative d'exécution



The screenshot shows the same IDE window after a build attempt. The 'Build' panel at the bottom shows the following error:

```
cours04: build failed At 21/0:5 sec, 83 ms
TestArrayList.java src 1 error
java: variable vectChaine might not have been initialized
```

Red arrows point from the error message to the 'vectChaine.size();' line in the code.

Insertion dans un ArrayList et capacité

- On peut toujours insérer dans un ArrayList
 - sous-réserve de ne pas excéder la capacité mémoire disponible

■ Gestion de la capacité d'un ArrayList

- De manière générale, lorsque la capacité courante est atteinte, une nouvelle insertion déclenche un déplacement dans une zone pouvant accueillir 1,5 fois la capacité courante, puis l'insertion est effectuée
- Avec le constructeur par défaut (`new ArrayList<>()`) la capacité initiale est de 10 éléments
 - donc après l'insertion de 10 éléments la capacité est atteinte
 - lorsque l'on veut insérer un 11^{ième} élément, l'objet ArrayList est déplacé dans un espace mémoire de capacité 15 ($1,5 \times 10$), puis l'insertion est effectivement réalisée

Exemple d'utilisation des méthodes

Code	<pre>1 import java.util.ArrayList; 2 public class ManipulationsArrayList { 3 public static void main(String[] args) { 4 // un vecteur de capacité initiale 10 5 ArrayList<String> vecteurDeChaines = new ArrayList<>(); 6 vecteurDeChaines.add("zéro"); // à l'indice 0 7 vecteurDeChaines.add("un"); // à l'indice 1 8 vecteurDeChaines.add("deux"); // à l'indice 2 9 vecteurDeChaines.add("trois"); // à l'indice 3 10 vecteurDeChaines.add("quatre"); // à l'indice 4 11 vecteurDeChaines.add("cinq"); // à l'indice 5 12 vecteurDeChaines.add("six"); // à l'indice 6 13 vecteurDeChaines.add("sept"); // à l'indice 7 14 vecteurDeChaines.add("huit"); // à l'indice 8 15 vecteurDeChaines.add("neuf"); // à l'indice 9 16 // capacité atteinte (déplacement et capacité de 15) 17 vecteurDeChaines.add("dix"); // à l'indice 10 18 vecteurDeChaines.add("onze"); // à l'indice 11 19 vecteurDeChaines.add("douze"); // à l'indice 12 20 System.out.println("nombre d'éléments de vecteurDeChaines : " + vecteurDeChaines.size()); 21 System.out.println(vecteurDeChaines); 22 System.out.println("insertion entre \"un\" et \"deux\" à l'indice 2"); 23 vecteurDeChaines.add(2, "entre un et deux"); // à l'indice 2 24 System.out.println("nombre d'éléments de vecteurDeChaines : " + vecteurDeChaines.size()); 25 System.out.println(vecteurDeChaines); 26 } 27 }</pre>
Trace	<pre>20 nombre d'éléments de vecteurDeChaines : 13 21 [zéro, un, deux, trois, quatre, cinq, six, sept, huit, neuf, dix, onze, douze] 22 insertion entre "un" et "deux" à l'indice 2 24 nombre d'éléments de vecteurDeChaines : 14 25 [zéro, un, entre un et deux, deux, trois, quatre, cinq, six, sept, huit, neuf, dix, onze, douze]</pre>

Illustration du contenu d'un ArrayList



Soient



E une classe



v déclaré et initialisé comme suit

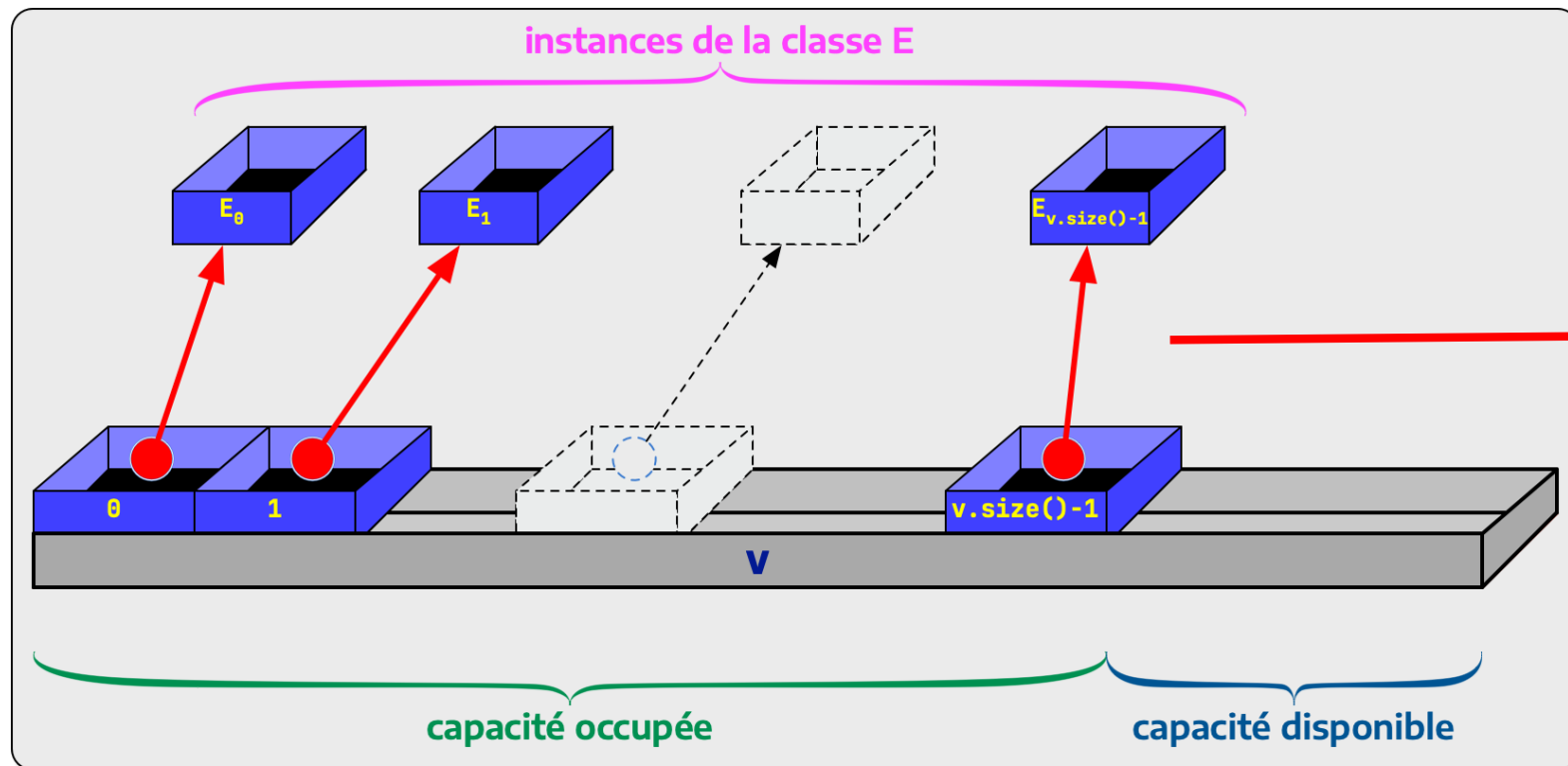


```
ArrayList<E> v = new ArrayList<>();
```



Après **v.size()** insertions, on peut schématiser **v** comme suit :

un ArrayList contient des
pointeurs sur des objets



RETOUR SUR LA DÉCLARATION ET LA CONSTRUCTION D'UN `ArrayList<E>`

Déclaration d'un ArrayList<E>

 **Déclaration** d'une variable de type ArrayList<E>

 ArrayList<E> monVecteurDeE;


 Effet dans la mémoire :



 le pointeur (la référence à l'objet) vaut **null**

 aucun objet n'a été construit !



 on ne peut pas utiliser les méthodes de la classe ArrayList<E> sur la variable monVecteurDeE !

Déclaration et construction d'un ArrayList<E>

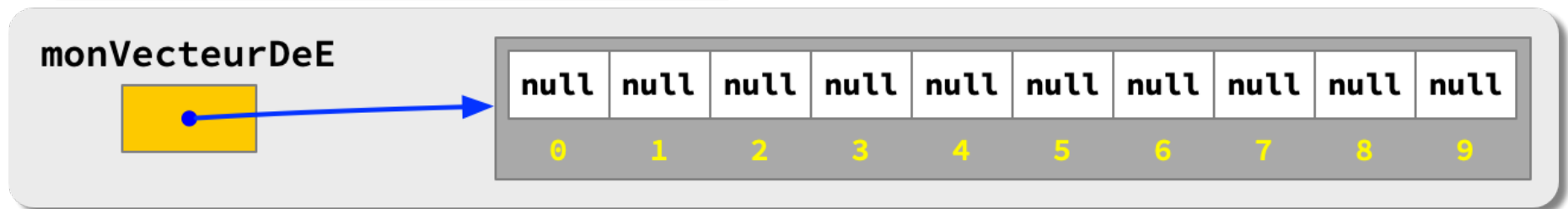
 **Déclaration** d'une variable de type ArrayList<E>

 ArrayList<E> monVecteurDeE;

 **Construction** de la variable

 monVecteurDeE = new ArrayList<E>();

 Effet dans la mémoire :



 un objet est construit avec new

 la variable monVecteurDeE **pointe** sur l'objet construit

 on peut utiliser les méthodes de la classe ArrayList<E> sur la variable monVecteurDeE !

DIFFÉRENTES FORMES D'INSERTION

Insertion unique (rappel)



 Insertion à la fin : `add(E element)`

 Insertion à un indice : `add(int index, E element)`

Code	Commentaire et Trace
<pre>1 public class TestArrayList { 2 3 public static void main(String[] args) { 4 ArrayList<Integer> vectInt = new ArrayList<>(); 5 6 vectInt.add(10); 7 System.out.println("vectInt : " + vectInt); 8 vectInt.add(12); 9 System.out.println("vectInt : " + vectInt); 10 vectInt.add(0, 9); 11 System.out.println("vectInt : " + vectInt); 12 vectInt.add(2, 11); 13 System.out.println("vectInt : " + vectInt); 14 vectInt.add(13); 15 System.out.println("vectInt : " + vectInt); 16 vectInt.add(15); 17 System.out.println("vectInt : " + vectInt); 18 vectInt.add(5, 14); 19 System.out.println("vectInt : " + vectInt); 20 } 21 }</pre>	<pre>6 // insertion à la fin -> indice 0 7 vectInt : [10] 8 // insertion à la fin -> indice 1 9 vectInt : [10, 12] 10 // insertion à l'indice 0 (première position) 11 vectInt : [9, 10, 12] 12 // insertion à l'indice 2 (troisième position) 13 vectInt : [9, 10, 11, 12] 14 // insertion à la fin -> indice 4 15 vectInt : [9, 10, 11, 12, 13] 16 // insertion à la fin -> indice 5 17 vectInt : [9, 10, 11, 12, 13, 15] 18 // insertion à l'indice 5 (sixième position) 19 vectInt : [9, 10, 11, 12, 13, 14, 15]</pre>

Insertion par lot

Situations

-  on veut créer rapidement un `ArrayList` de classe enveloppe ou `String`
-  on dispose d'un certain nombre de données du type `E` à insérer dans un `ArrayList<E>` au moment de sa création

Solution

-  utiliser la méthode `Arrays.asList()`

Exemples

```
ArrayList<Integer> unVectEnt = new ArrayList<>(Arrays.asList(12, 14, 4, 45,  
                                                             17, 23, 65, 18, 36, 8, 67));  
  
ArrayList<String> unVectString = new ArrayList<>(Arrays.asList("chaîne 1",  
                                                                "chaîne 2", "chaîne 3", "chaîne 4", "chaîne 5");  
  
ArrayList<Etudiant> unVectPersonne = new ArrayList<>(Arrays.asList(etu1,  
                                                                    etu2, etu3, etu4, etu5, etu6));
```