**JAVA - Basics**

**Atelier TP N° 3**

Collections

**Exercice 1 :**

Écrivez un programme Java pour extraire une partie d’un ArrayList. Il faut préciser la plage concernée.

**Exemple:**  
[1, 2, 3, 4, 5, 6, 7, 8, 9]

**Output prévu:**  
[2, 3, 4, 5]

Réécrivez le code suivant afin qu’il remplisse les valeurs de **l’ArrayList** avec les éléments du tableau à l’aide d’une [boucle **for-each**](https://waytolearnx.com/2020/05/boucle-for-each-java.html) au lieu d’une [boucle **for**](https://waytolearnx.com/2020/03/la-boucle-for-en-java.html).

**import java.util.\*;**

**public class Principale**

**{**

**public static void main(String[] args)**

**{**

**ArrayList<Integer> l = new ArrayList<Integer>();**

**int[] n = {1, 12, 23, 34, 45, 56, 67, 78, 89};**

**for (int i = 0; i < n.length; i ++)**

**{**

**l.add(n[i]);**

**}**

**System.out.println(l);**

**}**

**}**

**Exercice 2 :**

Réaliser une classe *Repertoire* permettant de gérer un répertoire téléphonique associant un numéro de téléphone (chaîne de caractères) à un nom. Prévoir une classe *Contact* destinée à représenter un contact.

La classe *Repertoire* devra disposer des fonctionnalités suivantes :

• constructeur recevant un argument entier, nombre maximum d’abonnés.

• méthode *addContact* permettant d’ajouter un nouveau contact ; retourne *false* si le répertoire est plein, *true* sinon.

• méthode *getNum, donne* le num associé à un nom de contact fourni en argument.

• méthode *getNContacts* qui fournit le nombre de contacts figurant dans le répertoire.

• méthode *getContact* fournissant le contact dont l’indice est donné en argument.

**Exercice 3 :**

On souhaite modéliser en java le calcul de coûts de transport de marchandises. Les marchandises transportées seront des instances de la classe Marchandise :

**public class Marchandise {**

**private double poids ;**

**private double volume ;**

**public Marchandise (double poids, double volume) {**

**this.poids = poids ;**

**this.volume = volume ;**

**}**

**public double poids () {**

**return poids ;**

**}**

**public double volume () {return volume ;**

**}**

**}**

Les marchandises sont transportées sous la forme de cargaisons. Avec les méthodes suivantes :

**Ajouter :** qui permet d’ajouter une marchandise dans cette cargaison si cela est encore possible.

**Cout :** qui retourne le coût total du transport de cette cargaison.

Une cargaison est caractérisée par la distance sur laquelle elle est transportée.

Une cargaison est limitée par l’encombrement total de ces marchandises à ne pas dépasser.

Cet encombrement est soit le poids total, soit le volume total des marchandises, selon le type de transport utilisé. Ce dernier influe également sur le calcul du coût de transport de la cargaison qui dépend de l’encombrement des marchandises de la cargaison.

On distingue plusieurs types de cargaisons selon le moyen de transport utilisé. On peut cependant trouver un certain nombre de caractéristiques communes à toutes les cargaisons que vous devrez identifier.

Tableau explicatif :



1. Ecrire les classes Java nécessaires pour modéliser ce cahier de charge.