Seul document autorisé : une feuille A4 recto-verso manuscrite. Aucune machine. Le barème est indicatif.
Une question peut toujours être traitée en utilisant les précédentes (traitées ou non).
Les morceaux de code Java devront être clairement présentés, indentés et commentés.
L'utilisation de collections de l'API Java est interdite (entre autres `LinkedList` et `ArrayList`).

On veut modéliser un carnet de contacts comme on peut en trouver sur un téléphone.

On pourra utiliser les méthodes suivante de la classe **`java.lang.String`** :

- **`char charAt(int index)`** qui renvoie le caractère situé à l'indice transmis en argument (indices numérotés à partir de 0) ;
- **`int length()`** qui renvoie la longueur de la chaîne.

1 Contact

Un contact correspond aux coordonnées d'une personne (nom, prénom, numéro de téléphone).

On modélise un contact comme une instance de la classe **`Contact`** qui contient donc au moins trois attributs privés de type chaîne de caractères : **`name`** (nom), **`firstName`** (prénom) et **`phone`** (numéro de téléphone). L'interface de cette classe (on ne spécifie pas ici quelle(s) méthode(s) est (sont) statique(s), ce sera votre travail) est donnée par :

```
/** renvoie le nombre de contacts crees depuis la premiere utilisation du carnet d'adresses
*/
int nbCreatedContacts();

/** renvoie le nom, le prenom et le telephone du contact */
String toString();

/** enregistre le numero de telephone du contact si son format est correct: 10 chiffres dont
le premier est un 0; sinon ne fait rien */
void setPhone(String phone);
```

Cette classe possède en outre deux constructeurs. L'un d'eux prend en argument deux chaînes de caractères représentant respectivement le nom et le prénom du contact ; l'autre prend un argument supplémentaire représentant le numéro de téléphone du contact qui n'est enregistré que si son format est correct.

Exercice 1. (1 points) Parmi les méthodes de l'interface de la classe **`Contact`**, lesquelles doivent être statiques et lesquelles pas ? (Justifiez la réponse.)

Exercice 2. (4 points) Écrire la classe **`Contact`**.

2 Carnet d'adresses

Dans cette partie, on modélise un carnet d'adresses sous forme d'une liste chaînée : la tête de liste est représentée par la classe **`AddressBook`**, les cellules sont représentées par la classe **`ContactCell`**.

La classe **`AddressBook`** possède l'interface suivante :

```
/** renvoie le nombre de contacts du carnet d'adresse */
int nbContacts();

/** renvoie le nombre de contacts dont le numero commence par "01" */
int nbFrancilienContacts();

/** affiche la liste des contacts (nom, prenom et telephone pour chaque contact) */
void ls();

/** affiche tous les contacts dont le nom est donne en parametre
 * @param name nom des contacts a afficher */
void lsByName(String name);

/** ajoute un nouveau contact dans le carnet
 * @param c contact a ajouter */
void addNewContact(Contact c);
```

```java
/** supprime de la liste le contact dont le telephone est donne en argument
 * @param phone telephone du contact a supprimer
 * @return true si le contact existait, false sinon */
boolean rmContact(String phone);

/** supprime de la liste tous les contacts dont le nom est donne en parametre
 * @param name nom des contacts a supprimer
 * @return true s'il existait de tels contact, false sinon */
boolean rmByName(String nom);
```

Dans chacun des exercices suivants vous pouvez écrire des méthodes non spécifiquement demandées si c'est plus pratique. Attention à bien spécifier pour chaque méthode dans quelle classe elle se trouve.

Exercice 3. (3 points) Donner les attributs et les constructeurs des classes **`ContactCell`** et **`AddressBook`**. Les attributs doivent être privés. À vous de décider quels constructeurs écrire pour que ces deux classes soient utilisables et que vous puissiez écrire toutes les méthodes demandées pour **`AddressBook`**.

Exercice 4. (2 points) Écrire les méthodes **`nbContacts`** et **`nbFrancilienContacts`**.

Exercice 5. (2 points) Écrire les méthodes **`ls`** et **`lsByName`**.

Exercice 6. (1,5 points) Écrire la méthode **`addNewContact`**.

Exercice 7. (2,5 points) Écrire les méthodes **`rmContact`** et **`rmByName`**. On suppose (sans avoir à le vérifier) que deux contacts différents ne peuvent partager le même numéro de téléphone, alors qu'ils peuvent partager le même nom.

3 Hachage

Pour accélérer l'accès aux données, on décide de représenter l'ensemble des contacts non plus sous forme d'une liste chaînée, mais sous forme d'un tableau de longueur 26 de listes chaînées, chaque case de ce tableau étant une tête de liste correspondant à tous les contacts de même initiale de nom (pour simplifier, on suppose qu'aucun nom ne comporte de lettre accentuée). Par exemple la case 0 permet de référencer sous forme de liste chaînée tous les contacts dont le nom commence par A, et la case 4 ceux dont le nom commence par E. Voir un exemple en figure 1.

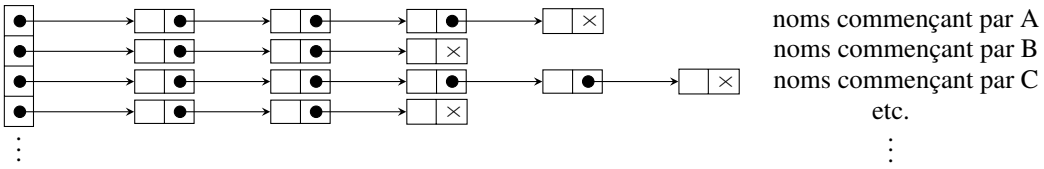

FIGURE 1 – Exemple de carnet d'adresses sous forme de tableau de listes.

Exercice 8. (2 points)

1. Quel est le type d'un tel tableau ?
2. Écrire une méthode statique qui prend en argument un tel tableau et affiche la liste de tous les contacts du carnet.

Exercice 9. (2 points) Écrire une méthode statique qui prend en argument un tableau tel que décrit ci-dessus et un nom et affiche tous les contacts possédant ce nom.