TP4 : Listes chaînées simples

Il vous faut respecter la présentation des listes faites en cours. C'est à dire que la modélisation doit se faire en utilisant 3 classes :

- une pour le contenu,
- une pour les cellules, où se trouve l'essentiel de la gestion du chaînage,
- et une principale, qui modélise la liste dans sa globalité, et qui implémente les méthodes classiques de l'interface de liste.

Dans ces exercices nous nommons les classes `Contenu`, `Cellule`, `Liste` d'une autre façon que dans le cours, mais vous les reconnaîtrez sans difficultés.

Conseil : Il est important de s'entraîner, et de s'entraîner encore à écrire du code java de ce type jusqu'à ce que vous soyez capable de le faire de façon très fluide. Il ne serait pas raisonnable de passer plus de 30 minutes sur les deux premiers exercices.

Cadre : On souhaite réaliser une chorale de robots. Pour cela on les regroupe en file indienne, où le premier est désigné chef. L'ensemble est représenté par une structure de liste chaînée, et nous les ferons chanter l'un après l'autre.

Exercice 1 - Les robots seuls

Voici le début de la classe `Robot` :

```
public class Robot{
    private final char nom;//lettre entre 'a' et 'z'
    private int energie;
    private final String texte; //ce qu'il doit dire

    public Robot(char nom, String paroles){
        this.nom = nom;
        // on donne une énergie entre 10 et 20
        energie = 10 + (int)(Math.random() * 11);
        texte = paroles;
    }
}
```

1. Ajoutez une méthode d'objet `description` qui retourne une chaîne de caractères décrivant le robot sur le modèle `"Robot <...> dit <...> quand il parle et a <...> points d'énergie"`
2. Ajoutez une méthode `boolean nomCorrect` qui vérifie qu'un robot a bien comme nom une lettre de l'alphabet minuscule.
3. Ajoutez des accesseurs permettant de récupérer les valeurs des différents attributs de `Robot`.

Exercice 2 - La mise en séquence

On considère les classes `Groupe` et `Cellule` suivantes :

```
public class Cellule{
    private Robot rob;
    private Cellule suivant;
```

```
}
public class Groupe{
    private Cellule chefDeFile;
 }
```

Ajoutez y les éléments suivants :

1. Un constructeur qui crée un groupe vide
2. Un constructeur de `Cellule` qui prend un argument de type `Robot` et un argument de type `Cellule`
3. Un constructeur de `Cellule` qui prend seulement un argument de type `Robot`, et qui fait appel au constructeur précédent avec `null` en second paramètre.
4. Écrivez une méthode `void prendreTete(Robot r)` qui teste si le robot `r` a un nom correct et si oui le place en position d'être le leader du groupe. L'ancien chef de file (avec sa suite), s'il existait, sera situé juste après `r`.

Exercice 3 - Parcours simples

1. Écrivez une méthode `affiche()` qui affiche la description de tout un groupe. Vérifiez votre méthode dans le cas d'un groupe vide.
2. Écrivez une méthode `ajouteNouveau` qui ajoute un robot en fin de groupe, l'ajout n'a lieu que si le nom du robot est correct.
3. Écrivez une méthode `bandName` qui donnera le nom du groupe constitué de la concaténation de tous les noms des robots pris dans l'ordre.
4. (facultatif) On rappelle que le type `char` peut être converti facilement en entiers. Par exemple l'expression `'b'-'a'` s'évalue à 1. Vous pouvez donc associer la première lettre de l'alphabet à la valeur 1 et la dernière à 26. Écrivez une méthode `numerologie` qui donnera pour résultat la somme des valeurs lettres des noms des robots, le tout modulo 9.
5. Les robots vont chanter leur texte : ils vont le répéter 2 fois si leur nom est 'b', 3 fois si c'est 'c' etc... Un robot qui s'appelle 'd' dont le texte est "pomme", chantera donc "pommepommepommepomme"[1]

   Ecrivez une méthode `chante` dans `Robot` qui affiche ce résultat. Après avoir chanté il perdra 10 points d'énergie (sans pouvoir passer en dessous de zéro).
6. Écrivez maintenant une méthode `chantez` qui fait chanter l'un après l'autre tous les robots du groupe considéré.

Exercice 4 : Opérations chirurgicales

1. Écrivez une méthode de la classe `Groupe` dont la signature est :
   `Groupe couperAPartirDe(char nom)`
   Elle élimine de la liste courante tous les robots qui sont placés derrière le premier robot dont le nom est précisé. Et elle retourne la liste de ceux qui ont été enlevés sous la forme d'un nouveau `Groupe`.

   Indications : Comme en cours, on écrira dans la classe `Groupe` simplement ce qu'il est nécessaire pour traiter des cas très particuliers, et l'essentiel du travail sera fait via une méthode auxiliaire de la classe `Cellule`. Probablement de signature :
   `Cellule couperAPartirDe(char nom)`
   Il est possible que cette démarche vous conduise à ajouter d'autres méthodes intermédiaires, ou d'autres constructeurs.

1. comme dans la 5ème symphonie de Beethoven

Exercice 5 : Petites variations (si vous avez le temps)

1. (***) Ecrivez une méthode de signature : `Groupe prendrePause()` qui s'applique à un groupe et en extrait les robots dont l'énergie est passée à zéro. C'est une question difficile à ce moment du cours, et il existe différentes façon de s'y prendre.
2. Définissez une classe qui encapsule un tableau de groupes.
3. Écrivez un constructeur ayant pour paramètre le nombre de groupes attendus.
4. Écrivez une méthode `effectifs` qui retournera un tableau d'entiers correspondant aux effectifs de chaque groupe d'un objet de cette classe.