TP7 : Listes chaînées circulaires et autres

Ronde d'enfants

On considère une liste chaînée simple, les cellules seront données par la classe `Enfant` ci-dessous

```
public class Enfant{
    private String nom;
    private Enfant suivant;

    public Enfant(String nom, Enfant suivant){
        this.nom = nom;
        this.suivant = suivant;
    }
}
```

et la liste elle-même par une classe `Ronde`

```
public class Ronde{
    private Enfant premier;

    public Ronde(){
        premier = null;
    }
}
```

1. Comment faire en sorte que la liste chaînée fasse une boucle ? On ne vous demande pas de code pour l'instant, faites un dessin pour le cas général et un autre pour une liste à un seul élément.
2. On veut qu'une liste de Ronde, boucle toujours sur le premier élément (liste circulaire). Où est-il le plus facile, dans le cas général, d'ajouter un élément sur une liste non-vide ?
   Écrivez la méthode `ajouter(String nom)` pour que cette propriété soit respectée. On fera attention aux cas particuliers.
3. Écrivez une méthode `Enfant dernierEnfant()` qui retourne le dernier élément.
4. Écrivez une méthode `supprimer(String nom)` qui supprime l'élément de nom `nom`. On suppose qu'il n'y en a qu'un.
5. Écrivez une méthode `Ronde concatener(Ronde r)` qui met bout à bout `this` et `r` pour en faire une nouvelle ronde (en respectant la boucle). Cette méthode ne fait pas de copie de listes, elle modifiera donc `this` et éventuellement `l`.

Ligne d'enfants

On veut maintenant représenter une liste chaînée linéaire, mais avec, en plus du lien vers le début de la liste, un lien vers la fin.
On reprend la classe `Enfant` précédente, mais on utilise maintenant la classe `Ligne` suivante :

```
public class Ligne{

    private Enfant premier;
    private Enfant dernier;

    public Ligne(){
        premier = null;
        dernier = null;
    }
}
```

Écrivez les méthodes suivantes en tirant profit de l'attribut `dernier` quand c'est possible. Il faudra aussi penser à mettre à jour correctement `dernier` en cas de modification.

1. Écrivez la méthode `ajouterEnFin(String nom)` qui ajoute en fin de la liste un enfant.
2. Écrivez la méthode `ajouterDebut(String nom)` qui ajoute en début de la liste un enfant.
3. Écrivez la méthode `dernierNom()` qui retourne le nom du dernier enfant de la liste.
4. Écrivez la méthode `enleveDernier()` qui enlève le dernier enfant de la liste.
5. Écrivez la méthode `concatener(Ligne l)` qui met bout à bout la ligne `this` et `l`. Attention, cette méthode doit fonctionner dans tous les cas, en particulier si `l` vaut `null` et si une ou les deux listes sont vides. Cette méthode ne fait pas de copie de listes, elle modifiera donc `this` et éventuellement `l`.

S'il reste du temps (facultatif)

1. finir le TP 6 si ce n'est pas le cas.
2. reprendre le TP 6 en changeant les règles : On considère que les cellules noires sont vivantes et les blanches sont mourantes.
   — les cellules ont un âge en jours ; à la création de la cellule, il est égal à 0.
   — les cellules vieillissent d'un jour à chaque étape.
   — Entre deux cellules noires d'âge au moins égal à 1 (à l'étape $t-1$) va naître une cellule noire à l'étape t ;
   — Les cellules noires de 3 jours ou plus à l'étape $t-1$ ont 50% de risque de devenir mourantes à l'étape t (elles deviennent blanches). Cet âge est abaissé à 2 jours si elle est entourée de deux cellules noires (pas assez à manger).
   — les cellules blanches ont plus de 70% de chances de mourir, c'est-à-dire de disparaître.