Les files et piles implémentées via pointeurs

Exercices obligatoires

A Implémentation de l'interface Pile via pointeurs

A1 Complétez à la main le document *A1*.
La solution de cet exercice se trouve sur moodle dans le dossier *solutions*. Elle est là pour vérifier vos réponses après avoir terminé l'exercice ! Le document s'appelle *A1Sol*.

A2 Sur moodle, répondez au questionnaire à choix multiples *PileImplChainee*.

A3 Implémentez l'interface *Pile* :
Complétez la classe *PileImplChainee*.
Testez-la avec la classe *TestPileImplChainee*.
Cette classe de tests propose un menu.
Ce menu permet de tester chaque méthode séparément.
Ce menu propose également de tester le scénario repris dans l'exercice A1.
Testez d'abord chaque méthode séparément avant de vérifier les implications des unes sur les autres via le scénario.

B Implémentation de l'interface File via pointeurs

B1 Complétez à la main le document *B1*.
La solution de cet exercice se trouve sur moodle dans le dossier *solutions*. Le document s'appelle *B1Sol*.

B2 Sur moodle, répondez au questionnaire à choix multiples *FileImplChainee*.

B3 Implémentez l'interface *File*.
Complétez la classe *FileImplChainee* et testez-là avec la classe *TestFileImplChainee*
La classe *TestFileImplChainee* reprend des tests pour chaque méthode ainsi que les tests repris dans le scénario repris dans l'exercice B1.

C Les drapeaux : le retour !

C1 La classe *DrapeauBelge* que vous allez compléter contient une liste chaînée de nœuds de la classe interne *NoeudCouleur*.
Cette liste va contenir les couleurs du drapeau belge.

Une couleur est représentée par un caractère :

Noir → n
Jaune → j
Rouge → r

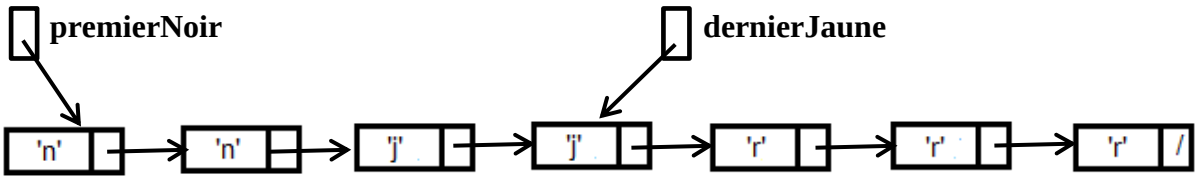
Dans la liste, les couleurs doivent apparaître triées selon les couleurs du drapeau belge : noir-jaune-rouge.

On vous demande d'écrire :

- Le constructeur
- La méthode `ajouter(char couleur)`

Pour éviter de nombreux cas particuliers à la méthode `ajouter(char couleur)`, le constructeur va initialiser la liste avec 3 nœuds : un noir, un jaune et un rouge.

Afin de rendre la méthode `ajouter(char couleur)` la plus efficace possible, 2 nœuds sont retenus : le premier nœud qui est de couleur noire (`premierNoir`) et le dernier nœud de couleur jaune (`dernierJaune`).

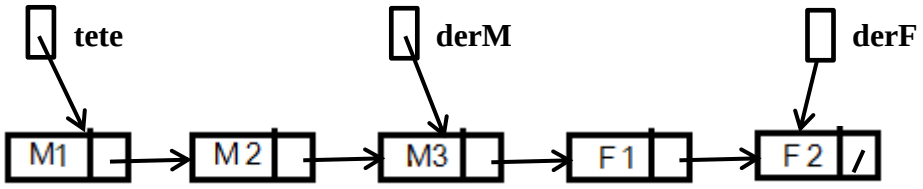


Avant d'écrire la méthode `ajouter()`, complétez les différents schémas qui se trouvent dans le document *Drapeau*.

Testez votre classe grâce à la classe *TestDrapeauBelge.*

D BAL

D1 La classe *Bal1* contient la liste des étudiants inscrits au bal de fin d'année.
La liste est triée par sexe. On y trouve d'abord les hommes et ensuite les femmes.
Pour ces deux sous-listes, l'ordre dans lequel les étudiants vont apparaître doit respecter l'ordre d'encodage (du plus ancien au plus récent).
Pour permettre des ajouts sans parcours de liste, 3 nœuds sont retenus :
Le nœud de tête (`tete`), le nœud contenant le dernier homme (`derM`) et le nœud contenant la dernière femme (`derF`).



Au départ, la liste est vide. Les 3 nœuds sont à `null`.
De nombreux cas vont devoir être envisagés pour la méthode `ajouterEtudiant()` :

M
- → la liste est vide
- → la liste ne contient que des hommes
- → la liste ne contient que des femmes
- → la liste contient des hommes et des femmes

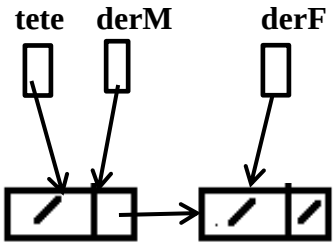
F
- → la liste est vide
- → la liste ne contient que des hommes
- → la liste ne contient que des femmes
- → la liste contient des hommes et des femmes

Avant d'écrire la méthode `ajouterEtudiant()`, complétez les différents schémas qui se trouvent dans le document *Bal1*.

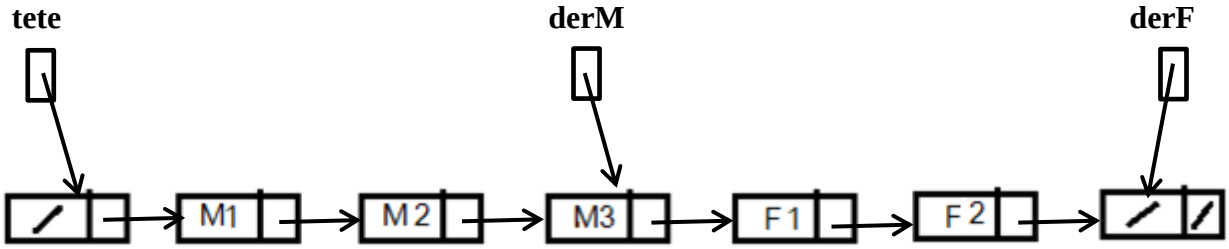
La classe *TestBal1* permet de tester votre classe.

D2 ❤️
Afin d'éviter de nombreux cas particuliers, souvent on ajoute dans une liste des nœuds « bidon ».
Complétez la classe *Bal2* qui est une variante de la classe *Bal1*.
Cette classe possède 3 attributs : le nœud de tête, le dernier nœud contenant un homme et le dernier nœud contenant une femme.
Le constructeur crée deux nœuds « bidon ». (Mettez les éléments de ces nœuds à *null*). Il crée une liste de départ en enchaînant ces 2 nœuds.

Au départ :

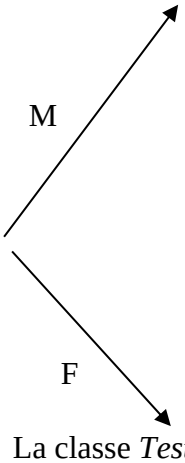


Après quelques ajouts :



La méthode `toString()` a été revue pour qu'elle ne renvoie pas d'étudiants « *null* » !

Grâce à la présence des 2 nœuds bidons, il ne reste plus que deux cas à envisager pour la méthode `ajouterEtudiant()` :
Pensez à faire des schémas.



La classe *TestBal2* permet de tester votre classe.

E *Deque*

Un *deque* (*double ended queue*) est une structure de données dans laquelle les ajouts et les retraits peuvent se faire aux 2 extrémités.

E1 Comme la file et la pile, le *deque* peut être implémenté via une structure chaînée.
Comment aller vous chaîner les différents nœuds ? Quels nœuds allez-vous retenir ?
Faites une représentation schématique.
💣Mise en garde :
Le chaînage qui va garantir un maximum d'efficacité n'est pas évident.
Faites **valider** votre représentation schématique par un professeur !

E2 Implémentez l'interface *Deque.*
Complétez la classe *DequeImplChainee* et testez-là avec la classe *TestDequeImplChainee*

Exercices supplémentaires

D3 ❤️
Pourquoi ne pas faire simple et utiliser l'existant.
La classe *Bal3* possédera 2 files : 2 objets de type *ArrayDeque.*
Les ajouts des étudiants se feront dans l'une ou dans l'autre file selon le sexe.
La méthode `toString()` renverra bien une seule liste, mais fera la concaténation des 2 files.

C2 La classe *DrapeauBelgeBis* est similaire à la classe *DrapeauBelge* mais elle retient 3 nœuds : le premier nœud de couleur noire, le premier nœud de couleur jaune et le premier nœud de couleur rouge.

E3 L'ajout de 2 nœuds « bidon » permet d'éviter de nombreux tests. Ces 2 nœuds sont appelés sentinelles. L'un prend place au début de la liste et l'autre en fin de liste.
Ecrivez une nouvelle implémentation de l'interface *Deque*.