|  |  |
| --- | --- |
| Ponctuel 15a | **Les listes chaînées (implantation)**  **partie I** |

* 1. Importez le projet *ListeBaseDepart.jar* . Examinez le code et exécutez le programme principal.  
        
     Modifiez la classe *TestListe* pour que l’application supprime l'élément de la liste contenu à la position correspond à cet indice et qui affiche cet élément.

Notez que si l'indice est en dehors des limites de la liste, la méthode *supprimerParIndice* affiche un message.

De plus, affichez le contenu de la liste une fois la suppression exécutée.

* 1. Dans la classe *Liste.java*, ajouter la méthode *E* **voir***(int indice)* qui retourne l'élément de la liste dont l'indice (ou position) est passé en paramètre (sans le retirer de la liste). Testez votre méthode avec tous les cas possibles (liste vide, liste à un seul élément, cas général). Affichez un message si l'indice est hors limites.
  2. Dans la classe *Liste.java*, ajouter la méthode *E* **avantDerniereDonnee***( )* qui retourne la valeur de l'avant-dernière donnée de la liste, si elle existe. Testez votre méthode avec tous les cas possibles (liste vide, liste à un seul élément, cas général). Affichez un message s'il n'y a pas d'avant-dernière donnée.

1. Dans le même projet, examinez le fichier *Client.java*.
   1. Vous avez l'intention de créer bientôt une liste de clients. Ajoutez les méthodes nécessaires à la classe *Client*. On fait l'hypothèse que deux clients sont considérés identiques si à la fois leur numéro de client et leur numéro de dossier sont identiques.
   2. Dans votre programme principal, ajoutez un client *client1* à votre liste :

Client client1 = new Client ("Bob", 12345, 99);

votreListe.ajouterFin (client1);

De la même manière, ajoutez deux autres clients *client2* et *client3* à votre liste.

Affichez le contenu de la liste.

* 1. Selon-vous, les deux groupes d'instructions suivantes donneront-elles le même résultat?

if (votreListe.contient( client1 )) {

System.out.println("Oui ce client est présent");

}

*et*

Client leClient = new Client("Jean-Paul ", 12345, 99);

if (votreListe.contient( leClient ) {

System.out.println("Oui ce client est présent");

}

* 1. Dans votre programme principal, demandez à l'usager d'entrer un numéro de client et un numéro de dossier, et supprimez ce client (utilisez notre nouvelle méthode *supprimerParValeur* ). Donnez un message si ce client n’était pas présent dans la liste.

Auriez-vous été capable d’accomplir cette tâche si vous aviez omis de redéfinir la méthode *equals* de la classe *Client*?

* 1. On désire conserver dans la liste de clients *lc* uniquement les clients dont le numéro de client est inférieur à 100. Selon vous, laquelle des solutions suivantes fonctionne?

|  |  |  |
| --- | --- | --- |
| i)  Client c;  int nb = lc.longueur();  for (int k=0; k<nb; k++){  c = lc.voir(k);  if (c.getNoClient() >= 100) {  lc.supprimerParIndice(k);  }  } | ii)  int indice = 0;  Client c;  for (int k=0; k< lc.longueur(); k++){  c = lc.voir(indice);  if (c.getNoClient() >= 100) {  lc.supprimerParIndice(indice);  } else {  indice++;  }  } | iii)  int indice = 0;  Client c;  int nb = lc.longueur();  for (int k=0; k<nb; k++){  c = lc.voir(indice);  if (c.getNoClient() >= 100) {  lc.supprimerParIndice(indice);  } else {  indice++;  }  } |

* 1. Donnez votre opinion sur l’efficacité de la boucle (iii) ci-haut. Si une liste comporte 100 éléments, combien de fois chaque nœud est-il accédé lors de l'exécution de cette boucle?
  2. Dans la classe Liste.java, ajoutez la méthode *int* **contientCombien***( E donnee )* qui retourne combien d'éléments de la liste sont "égaux" à la donnée passée en paramètre (inspirez-vous de la méthode *contient*). Testez votre méthode avec votre liste de clients. Votre méthode doit évidemment fonctionner pour tous les types d'éléments possibles!

1. Soient les deux listes suivantes:

Liste<String> maListe1 = new Liste<String>( );

maListe1.ajouterFin("Lundi");

Liste<Integer> maListe2 = new Liste<Integer>( );

maListe2.ajouterFin(99);

* 1. Pourquoi *ajouterFin(99)* fonctionne même si 99 n'est pas un objet?
  2. Selon vous, peut-on appliquer la méthode *contient* sur chacune de ces deux listes, même si l'on n'a pas définit soi-même une méthode *equals* ? Pourquoi?