Structures Intermédiaires
TP02

Exercice 1.)

Ex 1.1.)

La check-list figure ci-contre. Les tâches sont des booleens, c'est soit ok soit pas ok, on mettra 0 pour pas ok, et 1 pour ok. Le reste de la structure est une structure qu'on empilera. On aura une composition qui ressemblera au graphe suivant :

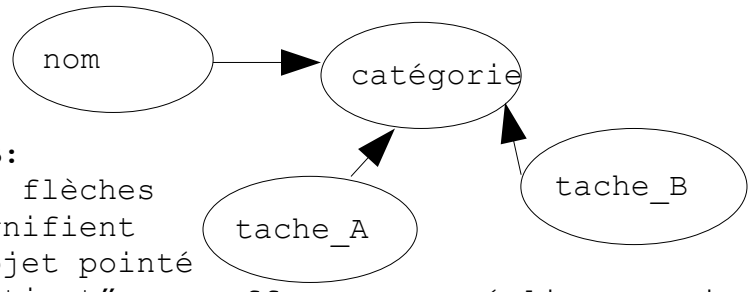
```
categorie,cockpit:

     sous-categorie,pilote
     tache,manette_A
     tache,manette_B

     sous-categorie,co-pilote
     tache,bouton_c
     tache,bouton_d

categorie,fuselage:
     sous-categorie,aile_droite
     tache,reacteur_A
     tache,reacteur_B

     sous-categorie,aile_gauche
     tache,reacteur_C
     tache,reacteur_D
```



N.B:
Les flèches signifient "objet pointé contient". En effet, pour réaliser ceci, on n'a en réalité pas besoin de deux types ! On peux juste en créer un, qui contient statu A et statu B ainsi qu'un nom, pour différencier le pilote du copilote de l'aile droite et gauche.

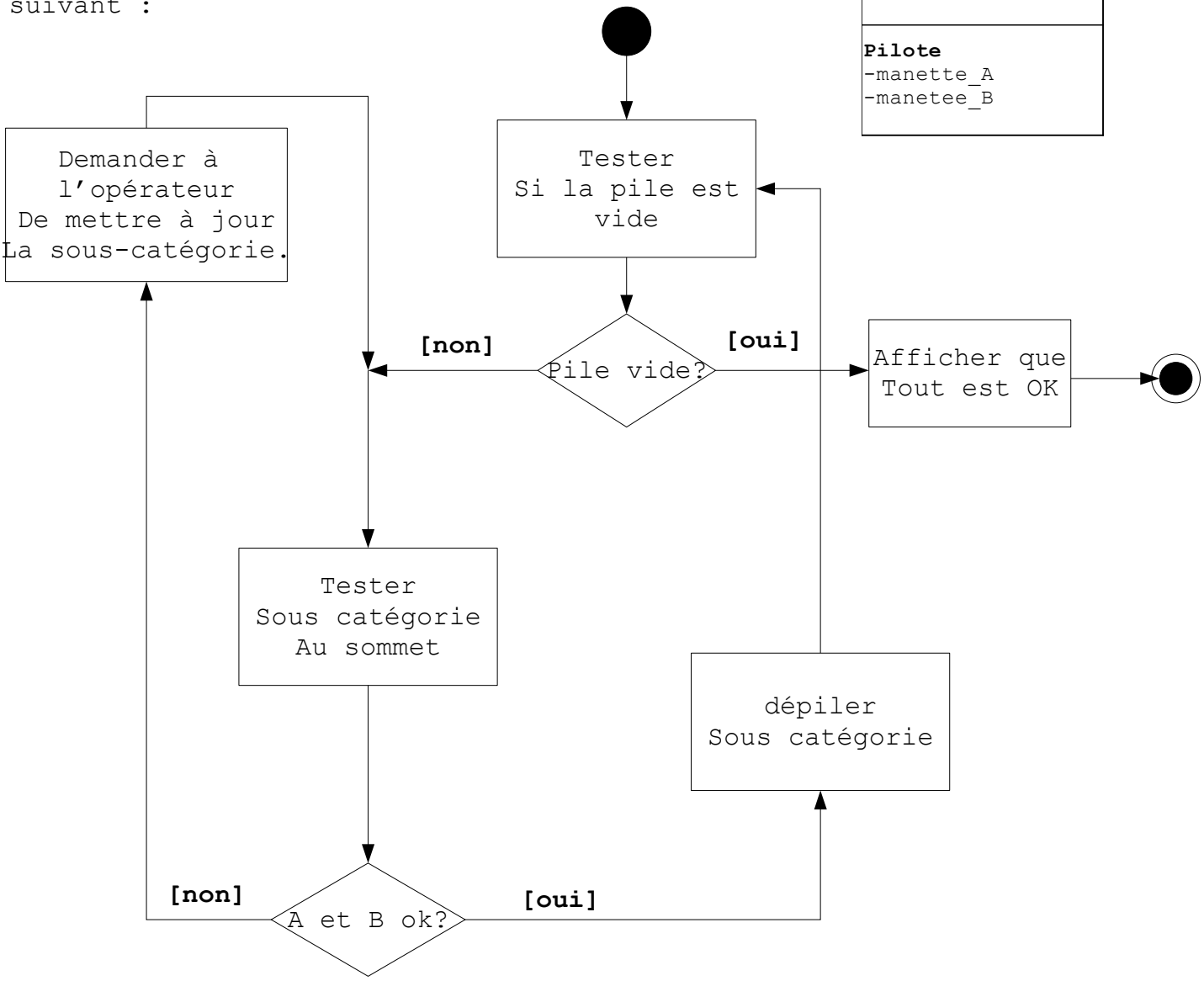
Et on stockera cette structure dans une pile. Grossièrement, on vas empiler tous les statuts dans une pile, puis les dépiler en vérifiant que ce soit bon à chaque fois, en cas de coquille, on recommence l'évaluation **de la catégorie en cours.** Donc si le poste pilote est ok mais qu'il y a une coquille chez le co-pilote, on recommence chez le co-pilote. On vas faire en sorte que l'opérateur puisse entrer les statuts dans la console, indiquant si une tâche est validée ou pas, puis le permettre de vérifier si tout est validé.

Ex 1.2.)

Sommet de la pile.
Aile droite -manette_A -manetee_B
Aile gauche -manette_A -manetee_B
Co-pilote -manette_A -manetee_B
Pilote -manette_A -manetee_B

On le représente par le tableau ci-contre. Ou le tableau symbolise une pile, en dépilant, on commence donc par vérifier que les systèmes sont bons c'est à dire, les ailes, puis on vérifie que le quo-pilote et pilote sont bons. On pourrait inverser l'ordre mais cela ne changerait pas grand-chose, mais je trouve cela plus logique de d'abord s'assurer qu'il n'y ait pas de problèmes techniques.

Le diagramme de la procédure, en UML est le suivant :



Ex 1.3.)

L'implémentation est en pièce jointe. J'ai implémenté une pile de taille statique, dont je déplace le pointeur au besoin pour savoir qui est au sommet de la pile ou pas.

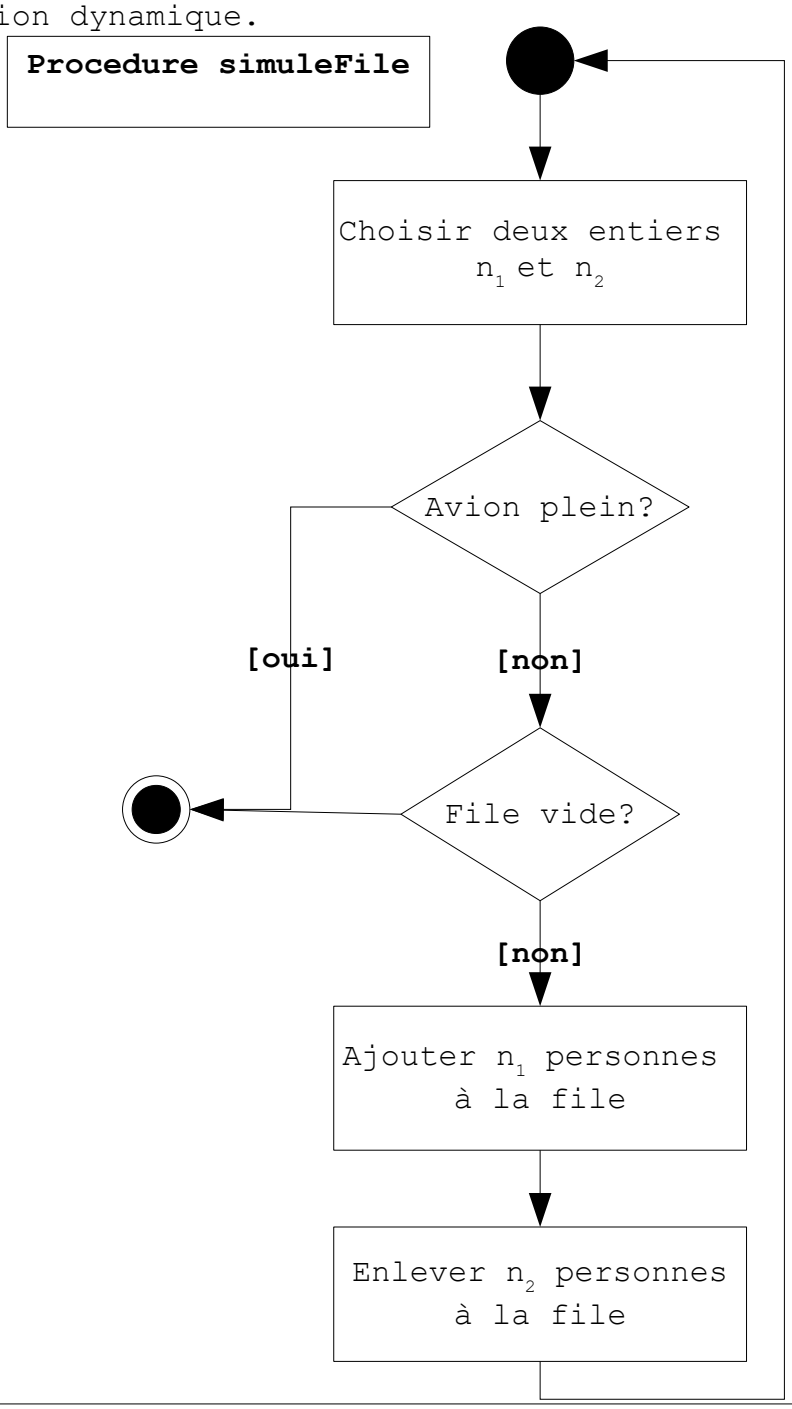
Il aurait fallu que je lise le fichier texte fourni, afin de pouvoir mettre à jour, rajouter ou enlever, les catégories au lieu de les « hard coder » dans le code. Je ne m'en suis que rendu compte qu'on était incité à le faire une fois que j'avais fini mon implémentation. J'ai donc décidé de ne pas l'implémenter cette fois, mais le code permettant de le faire est clair, on lit le texte ligne par ligne, on extrait le nom de chaque catégorie en utilisant des identifiants (catégorie, sous-catégorie, tache), et on prends le mot à droite de l'identifiant. Si c'est une sous-catégorie, j'initialise une nouvelle structure, j'y rajoute les taches en dessous, et j'empile et ré-belote lors de la lecture d'une nouvelle sous-catégorie.

Exercice 2.)

Ex 2.1.)

Nous nous servons d'une file, qui n'est rien d'autre qu'une chaîne à laquelle on rajoute des éléments au début (fin de la queue) et on enlève des éléments à la fin (début de la queue). Je fais une implémentation dynamique.

Ex 2.2.)

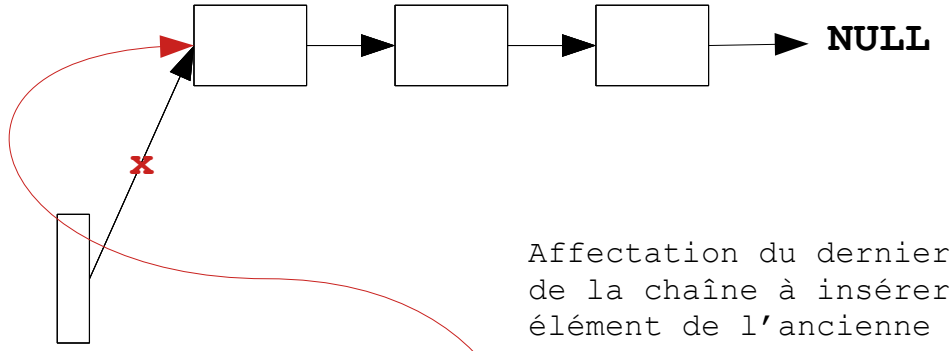


Ex 2.3.)

Pour que le lecteur puisse s'y retrouver, je rajoute des schémas pour illustrer comment fonctionnent les fonctions **add** et **removeLast.**

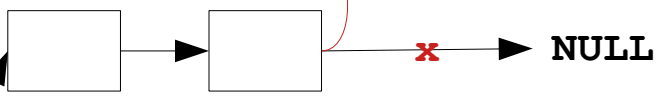
Add ajoute les nouveaux arrivants qui entrent dans la salle d'attente à la fil d'attente, elle est implémentée selon le principe suivant :

procedure add(debut:^TFile, int n):^TFile

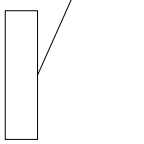


Affectation du dernier pointeur de la chaîne à insérer au premier élément de l'ancienne chaîne. (en rouge)

Debut: ^TFile



(n = 2)



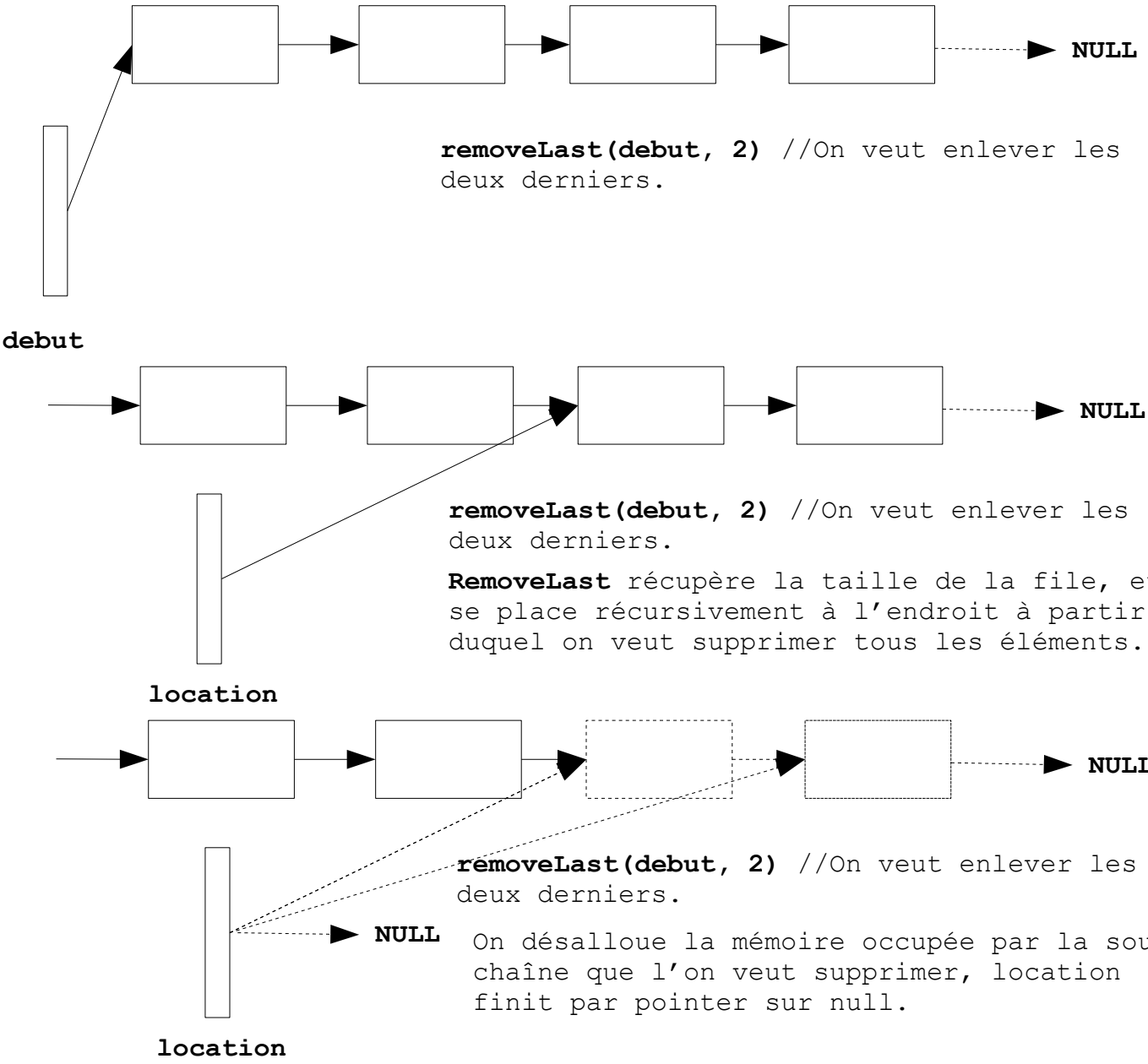
Nouvelle chaîne représentant les nouveaux arrivants dans la queue.

Nouvelle chaîne représentant les nouveaux arrivants dans la queue.

NewStart: ^TFile

return newStart, car il faut mettre à jour la localisation de la chaîne, le début s'est décalé vers l'arrière!

procedure removeLast(^TFile:debut,int n)



Globalement, l'implémentation marche bien, mais j'ai quelques soucis d'indexage de la file. C'est quelque part dans la fonction removeLast à mon avis, ou lors de mon décalage de pointeurs que se produit un pépin. Il y a également le fait qu'après la suppressions de personnes en début de file, la première personne aura toujours 0 ans, ce qui veut dire que createPersonne n'est pas appelée pour ce contenu.