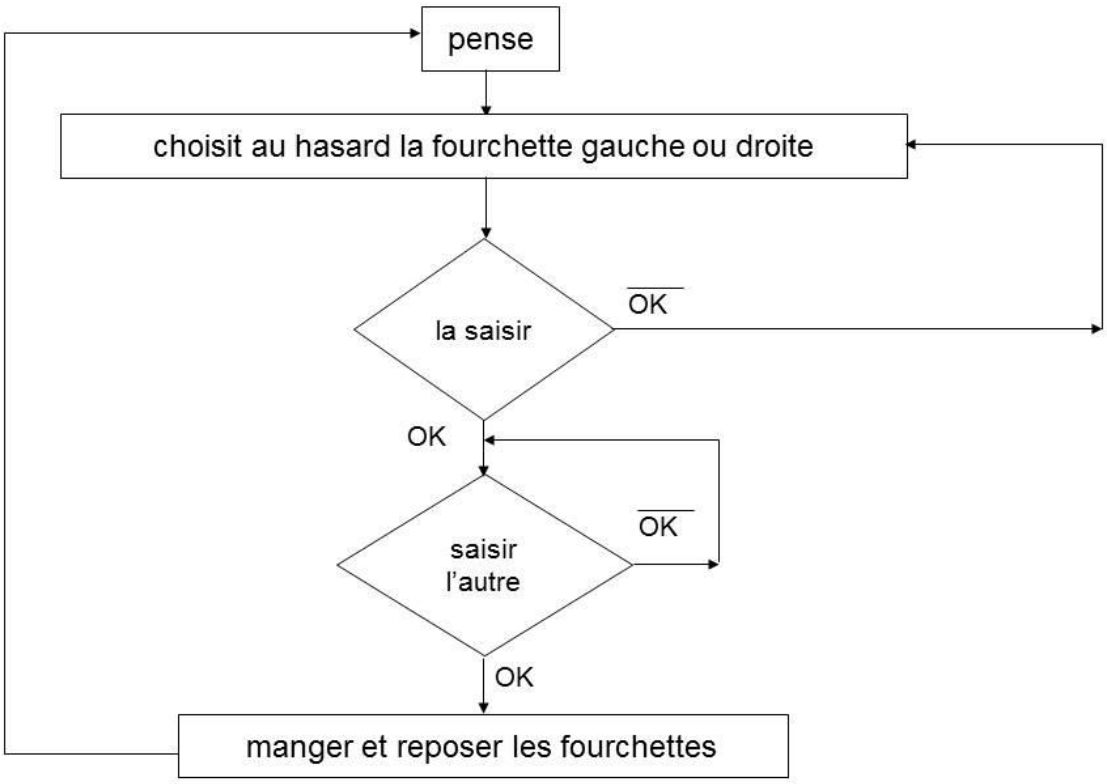
TP 9 Systèmes concurrents et distribués

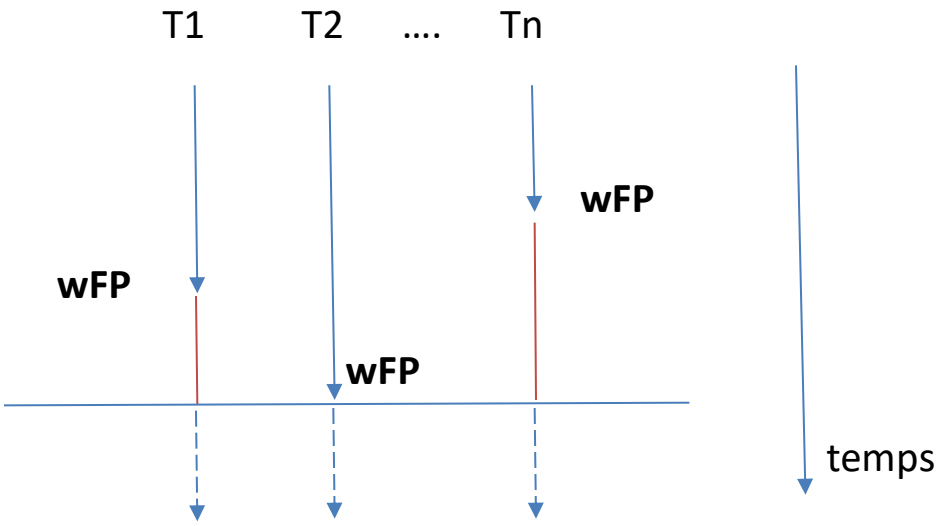
Exercice 1 : L'algorigramme (ISO 5807) ci-dessous représente un algorithme sensé résoudre le problème des philosophes.

1. Donner une exécution qui montre que l'algorithme n'est pas correct.
2. Quelle est la propriété nécessaire pour qu'un inter-blocage se produise que vous pouvez supprimer en modifiant l'algorithme ?
3. En général, combien au plus de philosophes peuvent manger simultanément ?



Exercice 2 : Implémentez une barrière de synchronisation en Java en utilisant des *sémaphores*.

On suppose qu'il y a *n* threads dans le système et que *n* est connu. La barrière de synchronisation permet aux n threads d'attendre les uns sur les autres. Les threads appellent waitForPool() et sont bloqués tant qu'un des *n* threads n'a pas appelé la méthode.



Attention, on peut utiliser la barrière à plusieurs endroits dans le programme, elle est réutilisable. Il faut s'assurer que les différents appels à wFP n'interfèrent pas.

Ajouter au code de la barrière un *printf()* **après** que le dernier thread ait accéder à wFP et **avant** que le premier thread soit libéré.

- Finalement, regardez la classe *java.util.concurrent.CyclicBarrier*.