Exercices dirigés
séance n°9

Exercice 1 : piles et files

Un système muti-tâches peut exécuter *n* tâches en quasi parallélisme. Chaque tâche est munie d'une priorité et d'un numéro. Elles sont rangées, dans l'ordre de leur arrivée, sur une *pile*. A chaque niveau de priorité est associée une *file*. Les tâches sont ensuite dispatchées sur l'une des files selon leur priorité de manière à ce que, pour une même priorité, la plus ancienne soit la première traitée.

Pour tester ce système, on considérera 3 niveaux de priorité. Le résultat du test sera un affichage de chaque tâche (sous la forme (heure, numéro)) rangés selon un ordre de priorité .

L'analyse de ce problème montre la nécessité de deux types de données à construire: les classes `Pile` et `File`.

Question 1

Construire la classe `Tache`.

Question 2

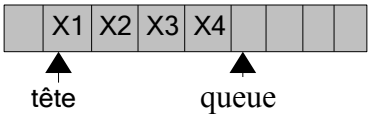
Construire la classe `Pile`. Les opérations possibles sont :

- `empiler` une tâche (ajouter une tâche au sommet de la pile)
- `depiler` une tâche (ce qui provoque son élimination de la pile)
- `sommet` (consulter le sommet de la pile sans le supprimer de la pile)

On s'inspirera de la classe `Liste` vue en cours pour construire la classe `Pile`. Il s'agira donc d'une représentation chaînée.

Question 3

Une file est accessible par une tête et une queue. Les éléments sont ajoutés en queue et retirés en tête (si elle n'est pas vide).



Construire la classe `File`. On choisira une représentation de la classe `File` sous la forme d'un tableau de taille suffisante. Les opérations possibles sont :

- enfiler une tâche (ajouter un élément dans la file selon l'ordre de priorité). On suppose que 2 tâches ne peuvent pas arriver en même temps (dates différentes). On n'envisagera pas de gestion circulaire du tableau.
- défiler, retourner l'élément en tête de file (c'est-à-dire le plus prioritaire).
- tester si elle est pleine
- tester si elle est vide

Question 4

Comment transformer ce système en exploitant les possibilités de l'Orienté Objet ?

Question 5

Ecrire un programme de test qui générera aléatoirement un certain nombre de tâches, les rangera selon leur priorité puis les affichera sous la forme (numéro, priorité) selon l' ordre de priorité et d'arrivée décroissante.

Rappel :

la méthode `int nextInt(int i)` de la classe `java.util.Random` génère un entier aléatoirement dans l'intervalle[0,i[.