UNIVERSITÉ TOULOUSE CAPITOLE

FACULTÉ D'INFORMATIQUE

Travaux pratiques n°5

Programmation Objet
Découverte de collections Java

Consignes

- La lecture de l'API Java sera indispensable pour ces exercices.

Objectifs

- Utiliser des types génériques.
- Découvrir les structures de données `ArrayList`, `ArrayDeque`, `LinkedList`, `HashSet`, `HashMap`.
- Analyser le contexte et savoir choisir la bonne structure de données en fonction de certains critères :
  - l'accès aux données : index, suivant, précédent, …
  - l'ordre des données,
  - l'autorisation de stockage de null, l'autorisation des doublons,
  - les opérations nécessaires : ajout, recherche, insertion, suppression, …
  - la position de l'opération : en tête, en fin, au milieu, …
  - l'exigence de performance en fonction des opérations les plus utilisées

Travail à faire

Toutes les classes doivent être réalisées, codées proprement, commentées et correctement testées dans le programme principal.

Exercice sur ArrayList

1) Reprendre l'exercice sur la liste FIFO du TP3 en remplaçant le tableau par une `ArrayList`.

Exercice sur ArrayDeque ou LinkedList

2) Ces deux structures sont de bonnes solutions à l'exercice de réflexion du TP3 lorsque l'auditeur en tête de liste est mise en ligne. Mais, attention, si les auditeurs raccrochent fréquemment, l'une de ces 2 structures doit être privilégiée.
   Choisir votre contexte, est-ce-que les auditeurs raccrochent fréquemment ou pas ?, et implémenter la structure la plus adaptée.

Exercice sur HashSet : Random

Supposons que nous utilisons un générateur aléatoire de nombres entiers entre 0 et 100. A chaque tirage, nous enregistrons le nombre généré dans une `HashSet` à condition qu'il n'ait pas déjà été généré dans un tirage précédent.

```
Random rd = new Random();
rd.nextInt(100+1);
```

3) Réaliser un sous-programme qui calcule le nombre de tirages aléatoires nécessaires afin de générer chaque entier entre 0 et 100 au moins 1 fois.

4) Réaliser le même sous-programme mais en stockant les nombres dans une `ArrayList`.

5) Calculer le rapport entre le temps de calcul nécessaire au sous_programme utilisant une `ArrayList` (question 4) et celui utilisant une `HashSet` (question 3) pour des nombres entre 0 et 10_000 puis entre 0 et 100_000.
Analyser le résultat.

`System.nanoTime()`
`System.currentTimeMillis()`

Exercice sur HashMap : Le scrabble

6) Réaliser un sous-programme qui calcule la valeur, en nombre de points, d'un mot au scrabble. Pour cela, vous vous servirez de la valeur de chaque lettre du mot pour la langue de votre choix, par exemple pour le français. Vous utiliserez une `HashMap` pour stocker la valeur de ces lettres.