TME 1 : Programmation, compilation et exécution en Java

Objectifs pédagogiques :

- **classes**
- **instances**
- **tableaux**
- **itérations**

1.1 La classe `MatriceEntiere`

Le but de cet exercice est d'écrire une classe `MatriceEntiere` offrant les fonctionnalités suivantes :

- la possibilité d'initialiser les éléments de la matrice à partir de données contenues dans un fichier ASCII;
- la possibilité d'exporter dans le même format à l'aide de la méthode standard `public String toString()`;
- les opérations de base : somme de matrices, produit de matrices, produit d'une matrice par un scalaire;
- le calcul de la matrice transposée;

Question 1. Réalisez la classe MatriceEntiere qui est munie d'un tableau à deux dimensions d'entiers. Son constructeur prend en paramètre le nombre de lignes et le nombre de colonnes de la matrice et initialise les éléments à zéro. Ajoutez aussi les accesseurs suivants : `public int getElem(int lig, int col)` et `public void setElem(int lig, int col, int val)` ainsi que les méthodes `public int nbLignes()` et `public int nbColonnes()`.

NB: Les tableaux Java ont un attribut `length` qui donne la taille du tableau.

Question 2. Ajouter l'opération `public static MatriceEntiere parseMatrix(File fichier) throws IOException`. Cette opération doit permettre de lire les fichiers de données fournis et de construire la matrice correspondante. Référez-vous aux tests fournis pour le format précis.

Question 3. Ajoutez à la classe `MatriceEntiere` une méthode `public String toString()` qui produit une répresentation sous la forme de String de la matrice, dans un format compatible avec les fichiers de test fournis.

Question 4. Ajoutez également une comparaison d'égalité logique standard entre deux matrices `public boolean equals(Object o)`.

Question 5. Ajoutez une méthode `public MatriceEntiere ajoute(MatriceEntiere m)throws TaillesNonConcordantesE` qui retourne la somme de la matrice courante et de m, où lève une exception (que vous définirez) si les dimensions ne sont pas concordantes.

Question 6. On rappelle que le produit des matrices $A(l_A, c_A)$ et $B(l_B, c_B)$ peut être calculé si $c_A = l_B$. Le résultat est une matrice $P(l_A, c_B)$ dont l'élément en position (i, j) a pour valeur :

$$p_{i,j} = \sum_{k=0}^{c_A - 1} a_{i,k}.b_{k,j}$$

Ajoutez `public MatriceEntiere produit(MatriceEntiere m)throws TaillesNonConcordantesException` qui calcule ce produit (ou lève une exception si les dimensions ne sont pas concordantes).

Question 7. Ajouter la méthode `public MatriceEntiere transposee()` qui retourne la transposée de la matrice.

1.2 String vs StringBuilder

La String de Java est immuable; pour cette raison il est toujours recommandé quand on construit une grande string petit à petit d'utiliser la classe StringBuilder qui est mutable. Cet exercice vous permettra d'apprécier la différence en pratique.

Question 8. On vous fournit une classe Repeat et un test associé. La classe actuellement ne passe pas le test, corrigez l'implantation pour utiliser un StringBuilder.

Question 9. Avant de finir la session pensez bien à push votre travail sur le dépôt.