Entrepôt de caisses de marchandises

Dans un entrepôt, les marchandises sont stockées dans des caisses. Chaque caisse est identifiée par un identifiant (une chaîne de caractères) et a un poids (exprimé en kilogrammes).

Dans l'entrepôt, il a des rangées de piles de caisses. Les caisses peuvent être manipulées mais il n'est pas possible d'en changer le contenu (donc le poids) ni l'identifiant.

On suppose que les caisses ont toutes le même format. L'entrepôt a un certain nombres de rangées, chaque rangée contient un certain nombre de piles de caisses, et chaque pile contient un certain nombre de caisses. L'entrepôt a une hauteur maximale exprimé en termes du nombre de caisses maximum que peut contenir une pile.

Question 1

Modélisez la classe `Caisse` en Java.

Question 2

On décide de modéliser les données de la classe `Entrepot` en utilisant un tableau de tableaux de `Caisse` et un un entier positif `hauteurMaximale` qui représente la hauteur maximale des piles de caisses. Par convention, on note `caisses[i][j]` la casse à la position `j` dans la pile `i`. Par convention, `j` représente la position de la caisse dans la pile en partant du bas (donc `caisses[i][0]` est la caisse en bas de la pile `i`).

```
class Entrepot {
    Caisse[][] caisses;
    int hauteurPileMax;
}
```

Implémenter le constructeur de la classe `Entrepot` qui initialise un entrepôt avec nbPiles piles vides.

```
Entrepot(int nbPiles, int hauteurPileMax) {...}
```

Question 3

On vous donne la classe suivante qui modélise la position d'une caisse dans l'entrepôt:

```
public class PositionCaisse {
    final int indiceRangée;
    final int indicePile;
    final int indiceCaisse;

    public PositionCaisse(int indiceRangée, int indicePile, int
indiceCaisse) {
```

```
        this.indiceRangée = indiceRangée;
        this.indicePile = indicePile;
        this.indiceCaisse = indiceCaisse;
    }
}
```

Implémentez les méthodes suivantes dans la classe `Entrepot`:

- getPositionCaisse:

```
/**
 * Renvoie la position de la caisse dont la référence est donnée en
paramètre
 * @param ref
 * @return la position de la caisse ou null si la caisse n'est pas dans
l'entrepôt
 */
PositionCaisse getPositionCaisse(String ref);
```

- ajouterCaisse :

```
/**
 * Ajoute une caisse dans une pile donnée
 * @param caisse la caisse à ajouter
 * @param indicePile l'indice de la pile dans laquelle ajouter la caisse
 * @return la position de la caisse ajoutée
 * @throws Exception "Indice de pile invalide" si l'indice de pile est
invalide
 * @throws Exception "Pile pleine" si la pile est pleine
 * @throws Exception "Caisse déjà présente" si la caisse est déjà dans
l'entrepôt
 */
PositionCaisse ajouterCaisse(Caisse caisse, int indicePile) throws
Exception;
```

- insérerCaisses :

```
/**
 * Insère des caisses dans une pile à une position donnée
 * @param caisses le tableau de caisses à insérer
 * @param indicePile l'indice de la pile dans laquelle insérer les caisses
 * @param positionDansPile la position dans la pile à laquelle insérer les
caisses
 * @throws Exception "Indice de pile invalide" si l'indice de pile est
invalide
 * @throws Exception "Position dans la pile invalide" si la position dans
la pile est invalide
```

```
 * @throws Exception "Pas assez de place dans la pile" si la pile n'a pas
assez de place pour accueillir les caisses
 * @throws Exception "Caisse déjà présente" si une des caisses est déjà
dans l'entrepôt
 */
void insererCaisses(Caisse[] caisses, int indicePile, int positionDansPile)
throws Exception;
```

- poidsTotal :

```
/**
 * Calcule le poids total des caisses dans l'entrepôt
 * @return le poids total des caisses
 */
double poidsTotal();
```

- poidsDesPiles :

```
/**
 * Calcule le poids des caisses dans chaque pile
 * @return un tableau contenant le poids de chaque pile
 */
double[] poidsDesPiles();
```