TD3 – Fichiers

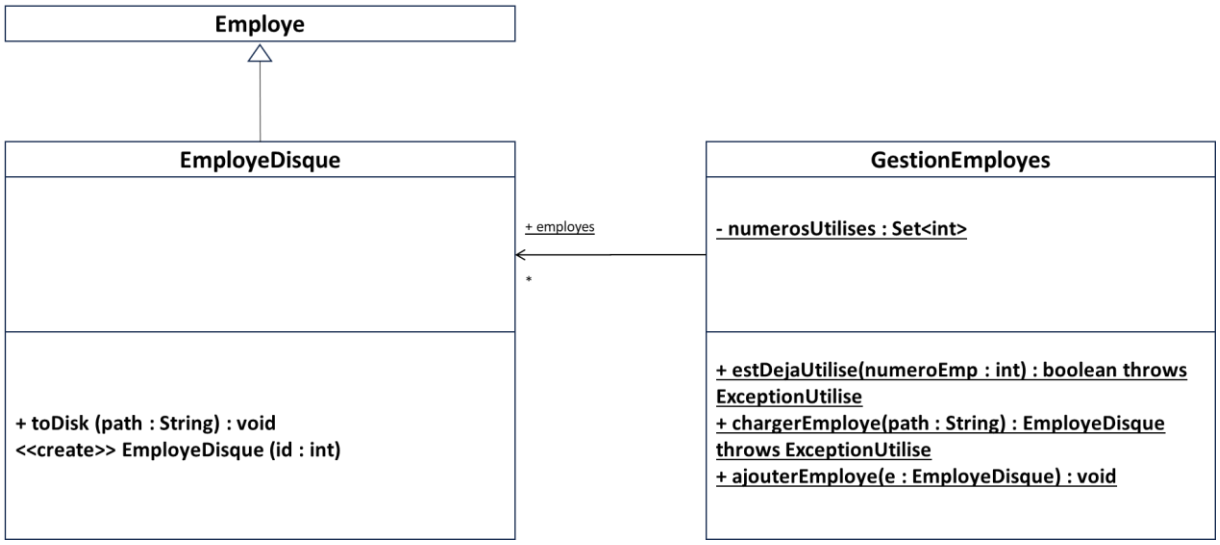
4ASTI – Programmation Java – B.Nguyen

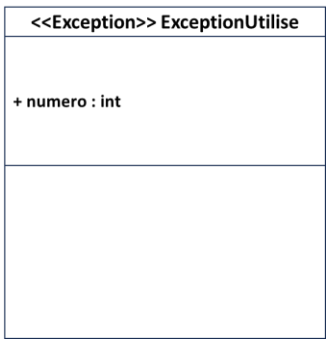
Question de cours : Arborescence de fichiers

Modifiez le fichier MainFichiers.java pour permettre une exploration récursive du système de fichiers. Donnez le chemin absolu de chacun des fichiers. Le paramètre d'entrée (l'attribut static path) doit être un répertoire, sinon lancez l'exception «FileNotFound ».

Exercice : Utilisation de fichiers

Afin de gérer des employés de manière persistante, nous allons sous-typer la classer Employe du TD précédent, en ajoutant les attributs et méthodes indiquées dans le diagramme UML ci-dessous, ainsi qu'une classe (statique) permettant de gérer les employés.





Questions modélisation –

Comment implémenter l'association `employes` ?

Faut-il passer certains attributs de `Employe` en protected ? (notation #) Si oui lesquels ?

Précisions

L'attribut de classe (static) `numerosUtilises` est un Set contenant l'ensemble des identifiants (donc des entiers) déjà utilisés. On ne demande pas forcément que tous les numéros utilisés partent de 0 de manière continue, mais on demande bien sûr à ce que le nombre d'employés soit correct.

La méthode `chargerEmploye()` doit charger un `EmployeDisque` depuis le disque, sachant que l'employé aura été écrit dans un fichier avec la méthode `toDisk()`, puis le créer en respectant son `numeroEmp` chargé.

La méthode `ajouterEmploye` rajoute un `EmployeDisque` déjà existant à `employes`.

On ne donne pas de précisions sur le format de stockage sur le disque (mais on devrait 😉), simplement toutes les informations nécessaires pour créer l'employé (y compris son numéro d'employé initial) doivent être stockés !

Question 1—

Implémentez et testez les classes `EmployeDisque` et `GestionEmploye`.

Le choix d'implémentation du format de stockage disque est le vôtre, ainsi que celui de l'implémentation de l'association `employes` de `GestionEmployes`.

Question 2—

1) Créez le programme principal suivant : création de 4 `EmployeDisque` avec les valeurs que vous voulez, puis écrire ces 4 `EmployeDisque` sur le disque.
2) Créez un @Test qui utilise `GestionEmploye` pour charger les 3 derniers employés créés, les stocker dans `employes` et vérifie que le `getNombreEmployes()` retourne 3.
3) Créez un @Test qui charge 2 fois le même employé et lance une exception `ExceptionUtilise`.
4) Ajoutez une méthode de classe à `GestionEmployes` qui écrit chaque `EmployeDisque` dans `employes` sur le disque avec le nom de fichier `nom-prenom-id.dat`
5) Ajoutez une méthode de classe à `GestionEmployes` qui charge l'ensemble des fichiers `.dat` d'un répertoire.
6) Ajoutez trois méthodes de classe à `GestionEmployes` permettant d'écrire tous les `Employe`, lire tous les `Employe` dans un fichier, et lire le i^{eme} `Employe` du fichier.

Question 4—

Ecrivrez un programme principal qui calcule le temps nécessaire en moyenne pour écrire un `EmployeDisque` sur le disque.