Activité pratique : Les entrées sorties

Exercice 1 :

On souhaite développer un programme java permettant de simuler la commande «ls», cette commande est utilisée pour lister les informations sur les fichiers et les répertoires d'un répertoire donné.

Le chemin complet du répertoire est saisi par l'utilisateur, puis la liste des fichiers et des répertoires contenus dans ce répertoire est affichée. Si ce répertoire contient des sous répertoires, il faut explorer de manière récursive ces sous répertoires.

Afficher pour chaque répertoire et fichier le type <DIR> pour répertoire et <FILE> pour les fichiers, ainsi que les modes d'accès permis **'r'** si accessible en lecture, **'w'** si accessible en écriture, **'h'** si c'est un fichier cache.

Exemple d'affichage :

..\xampp\htdocs\tp1\index.php <FICH> rw-

..\xampp\htdocs\tp1\accueil.htm <FICH> rw-

..\xampp\htdocs\tp1\images < DIR > rw-

Exercice 2 :

On souhaite créer un programme java qui permet de gérer la liste des numéros des contacts des clients d'une entreprise. On suppose que les numéros sont sauvegardés dans un dossier téléphonique, chaque numéro est sauvegardé dans un fichier séparé. Le nom du fichier porte le nom du contact et contient le numéro du contact.

L'objectif de cet exercice et de développer un programme d'annuaire téléphonique qui conserve la liste de noms et de numéros de téléphone des contacts dans des fichiers. L'utilisateur du programme doit pouvoir rechercher un nom dans le répertoire pour trouver le numéro de téléphone associé. L'utilisateur doit également être en mesure d'apporter des modifications aux données du répertoire. Chaque fois que le programme démarre, il doit lire les données des fichiers. Avant la fin du programme, si les données ont été modifiées pendant l'exécution du programme, le fichier doit être réécrit avec les nouvelles données.

- Créer une classe **DossierContact** qui contient une liste de noms et une liste de numéros de téléphone associés. Ajouter les méthodes qui vont permettre d'ajouter un contact, de supprimer un contact, de rechercher un contact par nom, et de changer le numéro de téléphone d'un contact.
- Ecrire un programme main pour gérer les contacts. Dans une boucle while, le programme présente à l'utilisateur un menu d'options :
    1. Rechercher un numéro de téléphone.
    2. Ajouter un nouveau contact.
    3. Supprimer un contact.
    4. Changer le numéro de téléphone d'un contact.
    5. Quitter ce programme.

Exercice 3 :

L'objectif de cet exercice est d'utiliser des fichiers en lecture et en écriture pour sauvegarder et relire une collection d'objets de type client.

- Créez une classe **Client** avec les attributs id, nom, prénom, adresse, tel, et email. La classe client doit implémenter l'interface **Serializable**.
- Créer une Interface **IMetierClient** qui va déclarer les méthodes pour gérer les clients. Cette interface contient les méthodes suivantes :
    - public Client addClient(Client c) : qui permet d'ajouter un objet de type client à la liste.

- public List<Client> getAllClients() : qui charge la liste des clients à partir d'un fichier et les retournent sous forme d'une liste.
- public Client findClientById(long id) : qui retourne un client par id.
- public void deleteClient(long id) : qui supprime un client par id.
- Public void saveAllClients() : qui permet de sauvegarder tous les clients dans le fichier.

- Créer une classe **MetierClientImpl** qui implémente l'interface **IMetierClient**. Cette classe contient un attribut qui représente une liste de clients et un attribut qui contient le nom de fichier pour sauvegarder les clients.
- Ecrire une classe **Application** contenant la méthode main qui propose à l'utilisateur dans une boucle while le menu suivant :
    1. Afficher la liste des clients.
    2. Afficher un client par son id.
    3. Ajouter un nouveau client dans la liste.
    4. Supprimer un client par id.
    5. Sauvegarder les clients : cette option permet de sauvegarder la liste des clients dans fichier nommé **clients.dat**.
    6. Quitter ce programme.
-