ENIAD BERKANE

TP 5 : POO en Java : Gestion des flux E/S

Exercice 1 : Calculatrice simple

Écrivez un programme qui lit deux nombres entiers et un opérateur (+, -, *, /). Le programme doit afficher le résultat de l'opération correspondante. Si l'opérateur n'est pas valide, affichez un message d'erreur.

Exercice 2 : Compter les mots dans un fichier

Écrivez un programme Java qui lit un fichier nommé `data.txt` et compte le nombre total de mots dans ce fichier. Affichez le résultat dans la console. Puis supprimer le fichier (avec delete()).

Exercice 3 : List des fichiers dans un répertoire

Créez un programme Java qui liste tous les fichiers et dossiers dans un répertoire donné (par exemple, `mon_repertoire`). Si le répertoire n'existe pas, affichez un message d'erreur. Renommer le premier fichier vers `ENIAD.txt`

Exercice 4 : Trouver un mot spécifique dans un fichier

Écrivez un programme qui recherche un mot spécifique, donné par l'utilisateur, dans un fichier nommé `recherche.txt`. Si le mot est trouvé, affichez le nombre d'occurrences et les lignes où il apparaît.

Exercice 5 : Fusionner deux fichiers

Créez un programme qui lit le contenu de deux fichiers (`fichier1.txt` et `fichier2.txt`) et écrit leur contenu combiné dans un nouveau fichier nommé `fusion.txt`.

Exercice 6 : Outil de sauvegarde de fichiers (Backup Tool)

Développer un programme Java permettant de sauvegarder un fichier binaire (PDF, image, ZIP) dans un dossier dédié.

Vous devez développer une application Java qui :

1. Lit un fichier binaire source en utilisant FileInputStream
2. Copie son contenu vers un fichier de destination en utilisant FileOutputStream
3. Effectue la lecture par blocs de 4096 octets
4. Crée automatiquement un dossier backup/ s'il n'existe pas
5. Conserve le nom original du fichier sauvegardé
6. Calcule et affiche :
    - la taille totale copiée (en octets)
    - la durée de la copie
7. Gère les exceptions et ferme correctement les flux

Le projet doit contenir :

- un fichier BackupManager.java
- un dossier backup/ contenant le fichier sauvegardé

À l'exécution, le programme doit afficher un message de confirmation indiquant le succès de la sauvegarde.

Exercice 7 : Analyseur de fichiers logs (Log Analyzer)

Analyser un fichier de logs afin d'identifier les erreurs et générer un rapport simple.

Vous devez développer une application Java qui :

1. Lit le fichier texte app.log ligne par ligne à l'aide de BufferedReader
2. Compte le nombre total de lignes
3. Identifie et compte les lignes contenant ERROR et WARNING
4. Écrit toutes les lignes contenant ERROR dans un fichier errors.log à l'aide de BufferedWriter
5. Ferme correctement les flux après traitement
6. Affiche dans la console un rapport récapitulatif

Le projet doit contenir :

- un fichier LogAnalyzer.java
- un fichier app.log
- un fichier errors.log généré à l'exécution

Exercice 8 : Gestion persistante des utilisateurs (Sérialisation)

Mettre en place une persistance simple des données utilisateurs sans base de données.

Vous devez développer une application Java qui :

1. Définit une classe User contenant id, username et email, et implémentant Serializable
2. Stocke les utilisateurs dans une ArrayList<User>
3. Sauvegarde la liste des utilisateurs dans un fichier binaire users.dat à l'aide de ObjectOutputStream
4. Charge automatiquement les utilisateurs depuis users.dat au démarrage à l'aide de ObjectInputStream
5. Permet l'affichage de la liste des utilisateurs chargés
6. Gère correctement les exceptions et la fermeture des flux

Le projet doit contenir :

- User.java
- UserService.java
- Main.java
- un fichier users.dat créé lors de la sauvegarde

Après redémarrage de l'application, les utilisateurs précédemment enregistrés doivent toujours être disponibles.