TP Java avancé

Exercice 1 : Gestion des exceptions dans un calcul de division

Dans cet exercice, vous allez écrire un programme Java qui effectue une division de deux nombres saisis par l'utilisateur et gère les exceptions qui pourraient se produire en cas de division par zéro ou de saisie incorrecte.

Voici les étapes à suivre :

1. Créez une classe Java appelée "DivisionCalculator".
2. Dans la méthode principale "main", demandez à l'utilisateur de saisir deux nombres (dividende et diviseur) à l'aide de la classe Scanner.
3. Utilisez un bloc try-catch pour effectuer la division des deux nombres saisis et gérer les exceptions qui pourraient survenir :
    - Si le diviseur est égal à zéro, affichez un message d'erreur indiquant que la division par zéro n'est pas autorisée.
    - Si l'utilisateur saisit un nombre invalide (par exemple, une chaîne de caractères non numérique), capturez l'exception NumberFormatException et affichez un message d'erreur indiquant que les nombres doivent être des valeurs numériques valides.
    - Si tout se passe bien, affichez le résultat de la division.

Exercice 2 : Calcul de moyenne pondérée

Dans cet exercice, vous allez écrire un programme Java qui calcule la moyenne pondérée de notes saisies par l'utilisateur. Le programme doit également gérer les exceptions qui pourraient survenir en cas de saisie incorrecte des notes.

Voici les étapes à suivre :

1. Créez une classe Java appelée "WeightedAverageCalculator".
2. Dans la méthode principale "main", demandez à l'utilisateur de saisir le nombre de notes à entrer (n) à l'aide de la classe Scanner.
3. Créez deux tableaux : un tableau pour stocker les notes (notes) et un autre tableau pour stocker les coefficients correspondants (coefficients). La taille de ces tableaux devrait être égale à "n".
4. À l'aide d'une boucle for, demandez à l'utilisateur de saisir les "n" notes et les "n" coefficients. Utilisez la classe Scanner pour cela.
5. Utilisez un bloc try-catch pour gérer les exceptions qui pourraient se produire lors de la saisie des notes et coefficients :
    - Si l'utilisateur saisit une note ou un coefficient invalide (par exemple, une chaîne de caractères non numérique), capturez l'exception NumberFormatException et affichez un message d'erreur indiquant que les valeurs doivent être des nombres valides.
6. Calculez la moyenne pondérée en utilisant la formule :

```
Moyenne pondérée = (note1 * coefficient1 + note2 * coefficient2 + ...
+ noteN * coefficientN) / (coefficient1 + coefficient2 + ... +
coefficientN)
```

Notez que "note1" correspond à la première note saisie par l'utilisateur et "coefficient1" correspond au premier coefficient associé à cette note, et ainsi de suite.

7. Affichez la moyenne pondérée calculée avec deux décimales.

Exercice 3 : Lecture et écriture dans un fichier texte

Dans cet exercice, vous allez écrire un programme Java qui lit des données à partir d'un fichier texte, effectue un traitement sur ces données, puis écrit le résultat dans un autre fichier texte.

1. Créez un fichier texte appelé "input.txt" et placez-y quelques lignes de nombres (un nombre par ligne). Assurez-vous que certains nombres sont négatifs et d'autres positifs.
2. Créez une classe Java appelée "FileIOExample".
3. Dans la méthode principale "main", utilisez la classe Scanner pour lire les données à partir du fichier "input.txt".
4. Utilisez un bloc try-catch pour gérer les exceptions qui pourraient se produire lors de la lecture du fichier :
    - Capturez l'exception FileNotFoundException et affichez un message d'erreur si le fichier n'est pas trouvé.
    - Capturez l'exception IOException pour gérer toute autre erreur d'entrée/sortie qui pourrait se produire lors de la lecture du fichier.
5. Effectuez un traitement sur les nombres lus du fichier. Par exemple, vous pouvez calculer la somme des nombres positifs et la somme des nombres négatifs.
6. Créez un fichier texte appelé "output.txt" pour écrire les résultats du traitement.
7. Utilisez la classe FileWriter et BufferedWriter pour écrire les résultats dans le fichier "output.txt".
8. Utilisez un bloc try-catch pour gérer les exceptions qui pourraient se produire lors de l'écriture dans le fichier :
    - Capturez l'exception IOException pour gérer toute erreur d'entrée/sortie qui pourrait se produire lors de l'écriture dans le fichier.
9. Fermez les flux de lecture et d'écriture après avoir terminé la lecture et l'écriture.

Exercice 4 : Gestion d'une liste de tâches à faire (To-Do List)

Dans cet exercice, vous allez créer un programme Java qui permet à l'utilisateur de gérer une liste de tâches à faire (To-Do List) en utilisant une collection appropriée.

Voici les étapes à suivre :

1. Créez une classe Java appelée "TodoListManager".

2. Déclarez une collection appropriée pour stocker les tâches à faire (par exemple, ArrayList, LinkedList, HashSet, etc.). Choisissez une collection qui vous semble la mieux adaptée pour une To-Do List.
3. Dans la méthode principale "main", affichez un menu pour permettre à l'utilisateur de choisir parmi les options suivantes :
    - Ajouter une nouvelle tâche à la liste
    - Afficher la liste des tâches
    - Marquer une tâche comme terminée
    - Supprimer une tâche de la liste
    - Quitter le programme
4. Implémentez chaque option du menu en utilisant les fonctionnalités de la collection choisie :
    - Pour ajouter une nouvelle tâche, demandez à l'utilisateur d'entrer le nom de la tâche et ajoutez-la à la collection.
    - Pour afficher la liste des tâches, parcourez la collection et affichez les tâches une par une.
    - Pour marquer une tâche comme terminée, demandez à l'utilisateur d'entrer le nom de la tâche et recherchez-la dans la collection. Si elle est trouvée, modifiez son état pour indiquer qu'elle est terminée.
    - Pour supprimer une tâche de la liste, demandez à l'utilisateur d'entrer le nom de la tâche et recherchez-la dans la collection. Si elle est trouvée, supprimez-la de la collection.
    - Pour quitter le programme, affichez un message de sortie et terminez l'exécution du programme.