TP 5 : Gestion des exceptions

Exercice 1:

On souhaite réaliser une application Java contenant une classe EntierNaturel permettant de gérer des entiers naturels (positifs ou nuls) et un nouveau type d'exception personnalisé en écrivant une classe NombreNegatifException qui spécialise la classe Exception. La classe EntierNaturel dispose :

- d'un constructeur avec un argument de type int pour initialiser l'attribut val; il générera une exception de type NombreNegatifException si la valeur de son argument est négative ;
- un accesseur en lecture getVal() qui fournira sous forme d'un int la valeur encapsulée dans un objet de type EntierNaturel;
- - un accesseur en écriture setVal() qui modifiera la valeur de l'entier naturel grâce à un int passé en paramètre; cette méthode générera une exception de type NombreNegatifException si la valeur passée en paramètre est négative;
- une méthode decrementer() qui décrémente de 1 l'attribut val de l'objet EntierNaturel; cette méthode devra pouvoir lever une exception de type NombreNegatifException;

Écrire une méthode main qui utilise les méthodes de la classe EntierNaturel, en capturant les exceptions susceptibles d'être générées.

On souhaite également mémoriser la valeur erronée qui a entrainé sa génération. Modifier la classe d'exception NombreNegatifException de façon à ce qu'elle permet le stockage de cette valeur, et fournir une méthode permettant de consulter cette valeur. Testez à nouveau.

Exercice 2:

Créez une classe CompteBancaire avec les attributs suivants : numéro de compte, solde, nom du titulaire.

Écrivez un constructeur et des méthodes pour effectuer les opérations suivantes:

- Dépôt d'argent sur le compte.
- Retrait d'argent du compte (gérez les exceptions pour les fonds insuffisants).
- Affichage du solde du compte.
- Transfert d'argent entre deux comptes (gérez les exceptions pour les fonds insuffisants et les comptes inexistants).

Créez deux classes, CompteCourant et CompteEpargne, qui héritent de CompteBancaire et ajoutent des fonctionnalités spécifiques. Par exemple, CompteCourant pourrait autoriser un découvert tandis que CompteEpargne pourrait générer des intérêts.

Gérez les exceptions dans ces classes :

- Si un retrait est effectué sur un solde insuffisant, lancez une exception personnalisée FondsInsuffisantsException.
- Si un transfert est effectué vers un compte inexistant, lancez une exception personnalisée CompteInexistantException.

Créez une classe Main qui déclare une liste de type ArrayList et qui effectuer les opérations suivantes :

- Ajouter des comptes
- Supprimer des comptes,
- Effectuer des opérations sur les comptes. Assurez-vous de gérer les exceptions correctement en affichant des messages d'erreur appropriés lorsque cela est nécessaire.