TP 2 : Exceptions, Généricité et Collections

Exercice 01 :

1) Sans utiliser les exceptions, écrire une classe Java (appelée *Division*) qui possède deux attributs entier privés numérateur et dénominateur (*num* et *deno*) qui seront initialisés uniquement par le constructeur. Cette classe contient une méthode (*division*()) qui affiche les deux attributs puis effectue leur division. Ecrire une autre classe (appelée *Main*) possédant une méthode *static void main*(*String[] args*) qui instancie un objet *Divion*. Les paramètres du constructeur (numérateur et dénominateur) sont passés comme arguments d'appel. Cette méthode invoque la méthode *division*() et affiche le résultat de la division.

2) En supposant qu'on ne prend aucune précaution pour la question 1. Quels sont les deux problèmes qui risquent de se produire. Dans chaque cas, donnez le nom de l'exception qui sera générée.

3) Réécrire la classe précédente (appelée *MainExcep*) de manière à ce que les erreurs possibles soient attrapées. Un message devra être affiché pour indiquer chaque type d'erreur qui s'est produite.

4) On souhaite que le logiciel obtenu ne soit valable uniquement lorsque le numérateur est plus grand que le dénominateur. Définir une nouvelle classe (appelée GererException) qui est utilisée lorsque le numérateur est plus petit que le dénominateur. Le message à afficher « Logiciel non valable » sera initialisé à l'instanciation de l'objet GererException.

Exercice 2 :

Soit à développer une application pour la gestion du stock. Un article est caractérisé par son numéro de référence, son nom, son prix de vente et une quantité en stock. Le stock est représenté par une collection d'articles.

Travail à faire:

Créer la classe Article contenant les éléments suivants :

Les attributs.
Un constructeur d'initialisation.

Dans la classe Main créer :

Le stock sous forme d'une collection d'articles de type ArrayList. Un menu présentant les fonctionnalités suivantes :

1. Rechercher un article par numéro.
2. Ajouter un article au stock en vérifiant l'unicité du numéro.
3. Supprimer un article par numéro.
4. Quitter.

Exercice 3 :

On s'intéresse à la représentation de données binaire qui est représenter par une suite de zéro et un. On représente un bit par un seul caractère '0' ou '1'.

1. Donner l'implémentation de la classe Bit comportant un constructeur avec paramètre, une méthode Inverse qui inverse un bit et une méthode toString qui retourne la valeur du bit.

On représente un octet par un vecteur de 8 bits.

2. Donner l'implémentation de la classe Octet comportant : un constructeur avec un paramètre avec une exception de type « Valeur de bit invalide » générée dans le cas où le maramètre passé comporte des caractères non binaires (0 ou 1). Une méthode toString qui retourne la succession de bits formant l'octet. Une méthode statique OrLogique (…) qui effectue le ou logique entre deux octets de type Octet (bit par bit incluant le bit du signe)et retourne un objet de type Octet (réutiliser les méthodes de la classe Bit).
3. Ecrire un programme qui créer une collection de type ArrayList d'objets Octet comportant n objets de type Octet. Effectuer le ou logique de tous les objets de la collection pour obtenir un seul octet à la fin.
4. On voudrait éviter l'arrêt du programme généré suite à l'introduction d'un octet non valide (capturer l'exception) et remplacer l'octet par un octet nul (complètement à zéro). Ecrire la solution que vous proposez.