TP N°3 en Java

Cycle ingénieur : 2ème Année GI

Exercice 1 :

Écrire un programme qui effectue une division par zéro et ne contient aucun traitement d'exception.
Que se passe-t-il ? Pourquoi ? Quel est le type de l'exception générée.
2- Cette fois, réécrire le programme pour capturer l'exception.
3- Le modifier pour afficher un message d'erreur explicite.
4- Cette fois, le programme corrige lui-même et remplace la division par zéro par une division par 1.
Remarquons ainsi que lever une exception ne signifie forcement l'affichage d'un message d'erreur suivi de l'arrêt du programme mais qu'il peut y avoir poursuite normale de l'exécution.

Exercice 2 :

La méthode parseInt est spécifiéé ainsi : *public static int parseInt(String s) throws NumberFormatException*
Throws: NumberFormatException - si le String ne contient pas un entier à convertir.

Utiliser cette fonction pour convertir et faire la somme de n entiers (saisis sous la forme de chaine de caractères)

Exercice 3 :

Ecrire une classe **Pile** qui implémente une pile d'objets avec un tableau de taille fixe. On définira pour cela deux exceptions **PilePleine** et **PileVide**. On utilisera pour écrire les méthodes, l'exception ArrayOutOfBoundsException qui indique qu'on a tenté d'accéder à une case non définie d'un tableau. Les champs de la classe seront :
private final static int taille = 10;
private Object [] pile;
private int pos;
Ecrire une méthode main qui empile n objets entrés au clavier.

Exercice 4 :

Réaliser une classe **EntNat** permettant de gérer des entiers naturels (positifs ou nuls) et disposant :
- d'un constructeur avec un argument de type int ; il générera une exception de type ErrConst si la valeur de son argument est négative ;
- un accesseur en lecture getN() qui fournira sous forme d'un int la valeur encapsulée dans un objet de type EntNat ;
- un accesseur en écriture setN() qui modifiera la valeur de l'entier naturel grâce à un int passé en paramètre ; cette méthode générera une exception de type ErrModif si la valeur passée en paramètre est négative ;
- une méthode décremente() qui décrémente de 1 un objet EntNat ; cette méthode devra pouvoir lever une exception de type ErrModif ;
- une méthode de classe – statique donc – décremente(EntNat e) qui décrémente de 1 l'objet passé en paramètre (c'est juste pour que vous travaillez sur les méthodes de classe, il serait en effet normal d'en faire une méthode d'instance …)
Écrire une méthode main qui utilise les méthodes de la classe EntNat, en capturant les exceptions susceptibles d'être générées.
- Organisez vos classes d'exception pour qu'elles dérivent toutes d'une classe ErrNat.
- Une exception doit mémoriser la valeur erronée qui a entrainé sa génération. Modifiez vos classes d'exception de façon à ce qu'elles permettent le stockage de cette valeur, et fournissent une méthode permettant de consulter cette valeur. Testez.
- Telle qu'elle est écrite, la classe EntNat est très contraignante : par exemple, lors de la création d'une instance de EntNat, on est obligé de prendre en compte l'exception susceptible d'être générée par le constructeur, même si l'on sait que la valeur passée en paramètres est correcte. Comment rendre optionnelle la prise en compte de ces exceptions ?

Exercice 5 : (à rendre avant le lundi 16/11/2020)

On définit la classe Compte suivante :

```
Public class  Compte{
        String  nom ;
        String prenom ;
        int solde ;
        String motDePasse ;
        public Compte ( String unNom, String unPrenom , int s , String unMot){
                nom=unNom; prenom=unPrenom; solde = s ; motDePasse = unMot ; }
}
```

1. Ecrire une méthode d'instance depot(int somme) qui permet d'ajouter somme au solde de l'objet.
2. Ecrire une classe CodeIncorrect qui hérite de la classe Exception dont le constructeur affiche code incorrect.
3. Ecrire une classe RetraitInterdit qui hérite de la classe Exception dont le constructeur affiche retrait non autorisé.
4. Ecrire une méthode d'instance retrait (int somme, String unMot)
   - qui lance une exception CodeIncorrect si le paramètre unMot n'est pas le code de l'objet, puis si cette exception n'est pas lancée
   - qui permet de soustraire somme au solde de l'objet si le résultat est positif
   - sinon qui lance une exception RetraitInterdit.
5. Ecrire une classe CompteAvecDecouvert qui hérite de la classe Compte avec un attribut entier (positif) decouvertAutorise et avec un constructeur CompteAvecDecouvert(String unNom, String unPrenom, int s, String unMot, int decouvert).
6. Réécrire la méthode d'instance retrait(int somme, String unMot) pour qu'elle ne lance une exception RetraitInterdit que si le paramètre somme dépasse le solde de decouvertAutorise.
7. Tester votre classe.