TP 1 : Exceptions

Matière : ATELIER PROGRAMMATION OBJET AVANCEE | **Niveau :** DSI 2

Enseignants : Equipe pédagogique

Exercice 1 :

1. Que donne le programme CalculMoyenne lancé avec les paramètres suivants : x, -1, 30, 2.5, 18, 16 ?
2. Quelles sont les exceptions prédéfinies et capturées par ce programme ?
3. Quelles sont les exceptions crées par le programmeur ?

```
package ex1;
public class CalculMoyenne {
    static int moyenne(String[] liste) throws ExceptionRien {
        int s = 0, entier, nbNotes = 0;
        for (int i = 0; i < liste.length; i++) {
            try {
                entier = Integer.parseInt(liste[i]);
                if (entier < 0)
                    throw new ExceptionIntervalle("petite");
                else if (entier > 20)
                    throw new ExceptionIntervalle("grande");
                else {
                    s += entier;
                    nbNotes++;
                }
            } catch (NumberFormatException e) {
                System.out.println("note non entière");
            } catch (ExceptionIntervalle e) {
                System.out.println("la note " + (i + 1) + " est trop " +
e.getMessage());
            }
        } // fin for
        if (nbNotes == 0)
            throw new ExceptionRien();
        return (int) s / nbNotes;
    }

    public static void main(String arg[]) {
        try {
            System.out.println("La moyenne est :" + moyenne(arg));
        } catch (ExceptionRien e) {
            System.out.println(e);
        }
    }
}
```

```
class ExceptionRien extends Exception {
    public String toString() {
        return ("aucune note n'est valide");
    }
}

class ExceptionIntervalle extends Exception {
    ExceptionIntervalle(String s) {
        super(s);
    }
}
```

Exercice 2 :

Soit la classe Temps caractérisée par les attributs heures, minutes et secondes. Les méthodes de cette classe sont :

- un constructeur avec les trois paramètres de manière à ce qu'il lance une exception de type TempsException si les heures, les minutes ou les secondes ne correspondent pas un temps valide ;
- une méthode de lancement de manière à ce que l'exception TempsException soit traitée en affichant le message « Temps invalide ».

Ecrire la classe Temps.

Exercice 3 :

Ecrire une classe TestFactorielle qui calcule la factorielle d'un entier donné en argument, en attrapant les différentes exceptions de façon à préciser la difficulté à l'utilisateur, lorsque le calcul de factorielle est impossible.

Les exceptions à prévoir sont :

- Pas d'argument lu : ArrayIndexOutOfBoundsException
- La valeur de l'argument n'est pas entière : NumberFormatException
- La valeur de l'argument est négative : créer une classe ExceptionNegative ;

Exercice 4 :

Définir une classe Pile, permettant de représenter une pile d'entiers stockée sous forme d'un tableau. Il fallait lancer et traiter les exceptions suivantes : « Pile pleine » et « Pile vide » lors des opérations d'empilement, de dépilement et d'affichage.

Les méthodes à prévoir sont :

- Constructeur de la pile qui reçoit en arguments la taille de la pile ;
- void empiler(int e) : met l'entier à la fin de la pile ;
- void depiler() : enlève le dernier élément de la pile ;
- boolean pilePleine() : teste si la pile est pleine ;
- boolean pileVide() : teste si la pile est vide ;
- void afficher() : affiche le contenu de la pile si la pile contient des éléments sinon affiche le message « La pile est vide ».

Définir la classe TestPile permettant de tester la création d'une pile de 10 entiers, appeler les méthodes empiler(...), depiler() et afficher() ; tout en traitant les exceptions prévues.