TP 0 – Les Exceptions – les expressions Lambda

Excercice 1:

On se propose d'écrire une classe utilitaire nommée **`OperationArithmetique`** offrant des méthodes statiques d'addition, de soustraction, de multiplication et de division de deux entiers.

Etant donnée qu'un entier en java est codé sur 32 bits alors on ne pourra pas faire des opérations sur des valeurs dépassant 2^{31}-1 (Integer.MAX_VALUE). Dans ce cas de figure les méthodes déclenche une exception **TooLargeValueException** (qu'on définira).

Dans le cas d'une division par zéro la méthode lance une exception **IllegalDivisionException** (qu'on définira) qui portera le message "division par zéro impossible".

Exercice 2:

La suite de Fibonacci est définie par :

$$\begin{cases} f_0 = 1 \\ f_1 = 1 \\ f_n = f_{n-1} + f_{n-2} \end{cases}$$

Ecrire une fonction calculant le Nième nombre de la suite.

L'entier saisi doit être impérativement supérieur à 0.

On demande donc de créer une classe d'exception adaptée à cette erreur pour la traiter.

Exercice 3:

On cherche, au cours de cet exercice, à vérifier si une expression est correctement parenthésée. Voici quelques exemples :

```
((a+b)-(c+d))        correct
a+(b+(c+d))          correct
((a+b)+c        incorrect
((a+b)+c))           incorrect
)a+b(                incorrect
```

Ecrire une classe **ExpressionParenthesee** qui est composée d'une String pour mémoriser l'expression. Ecrire un constructeur pour lequel on passe une chaîne de caractères en paramètre.

Encapsuler la String en privé et écrire une méthode **String getString();**

3 cas peuvent se présenter :

- l'expression est correcte
- il y a des parenthèses ouvrantes en trop
- il y a des parenthèses fermantes en trop

On va cherche à lever des exceptions en cas d'erreur. Ecrire 3 classes d'exceptions :

- **ParentheseException** héritant de la classe Exception
- **ParentheseOuvranteException** héritant de la classe ParentheseException
- **ParentheseFermanteException** héritant de la classe ParentheseException

Dans le constructeur de la classe **ExpressionParenthee**, écrire un algorithme qui vérifie si l'expression est correctement parenthésée ou qui déclenche une exception appropriée en cas d'erreur.

Ecrire une méthode main permettant de vérifier le fonctionnement de votre classe.

Exercice 4:

Soit une classe Personne définit par firstName, lastName, age, les méthodes Getters/Setters et la méthode toString().

1. Implémenter la classe Personne contenant un constructeur prenant 3 paramètres.

2. Ecrire une classe principale TestPersonnes qui permet de:

(a) Créer une liste de personnes avec les valeurs suivantes: (Utiliser la classe Arrays pour construire la liste)

(b) Trier les personnes selon leur lastName en utlisant la méthode sort() de la classe Collections. Une des versions de cette méthode prend deux paramètres le premier est la liste à trier et le 2ème est un Comparator (une interface fonctionnelle qu'on peut utiliser comme type pour les expressions Lambda).

(c) Ecrire une méthode statique afficherSelon() qui permet d'afficher toutes les personnes ou bien un ensemble de personne selon un critère d'affichage.

Penser à utiliser les interfaces fonctionnelles pour faire passer vos expressions Lambda. On dispose de deux interfaces fonctionnelles intéressantes; Predicate<T> et Consumer<T>.

On pourra avoir une méthode ayant cette signature :

```
public static void afficherSelon(List<Personne> personnes, Predicate<Personne> predicate, Consumer<Personne> consumer)
```

predicate servira pour faire passer une expression Lambda qui fixe le critère d'affichage, et *consumer* servira pour faire passer l'expression Lambda qui affichera une personne.