Correction DS prog fonc 2024

par Tom

le 16/07/2024

1.

```
double average(Lst<Lst<Integer>> l) {
    return averageHelper(l.car(), l.cdr(), 0, 0);
}

double averageHelper(Lst<Integer> cur, Lst<Lst<Integer>> rest, int sum, int count) {
    if (cur == null) {
        if (rest == null) {
            return (double)sum / count;
        } else {
            return averageHelper(rest.car(), rest.cdr(), sum, count);
        }
    }

    return averageHelper(cur.cdr(), rest, sum + cur.car(), count + 1);
}
```

2.

```
<T> UnaryOperator<T> retainer(T e) {
    var current = new Object() {
        T value = e;
    };
    return (new_e) -> {
        var old = current.value;
        current.value = new_e;
        return old;
    };
}
```

3.

```
Stream<Character> toStream(String s) {
    return IntStream.iterate(0, i -> i + 1)
        .limit(s.length())
        .mapToObj(i -> s.charAt(i));
}
```

4.

1. Crée un record `SN` avec les membres `s` et `n` entiers.
2. Déclare une fonction `mysterious` prenant un Stream de IntStream.
3. Définit un opérateur `red` qui pour deux SN `p` et `q` renvoie un nouveau SN constitué de leur somme élément par élément.
4. Définit un Stream `sub` qui, pour chaque IntStream passé en entrée (dans le Stream `stream`), transforme chaque Integer en un `SN` contenant ce nombre et 1, puis réduit tous les `SN` résultant à l'aide de l'opérateur `red`.
    - Pour un IntStream qui donne (a, b, c, ...) on obtient (SN(a, 1), SN(b, 1), SN(c, 1), ...). Ensuite, on réduit à l'aide de `red`, ce qui donne un `SN` composé de la somme de tous les `s` et tous les `n`. On obtient donc SN(a+b+c+...), SN(1+1+1+...).
    - En fait, on calcule ici la somme et le nombre des éléments de l'IntStream (`SN` = Sum & Number).
5. Parcourt `sub` pour ne garder que les valeurs présentes (ici, ça veut dire les valeurs issues de `IntStream` non vides). On réduit à nouveau en faisant la somme pour obtenir au final la somme de tous les éléments de tous les IntStream, et leur nombre.
6. On divise la somme par le nombre pour obtenir la moyenne, ou si le résultat était vide (donc qu'il n'y avait aucun `IntStream` non vide), on renvoie simplement `NaN`.

Bref, la fonction `mysterious` calcule la moyenne des (sous-)éléments d'un `Stream` d'`IntStream`, comme la fonction de la question 1

5.

```
record Option<T>(T value) {
    static <U> Option<U> empty() {
        return new Option<>(null);
    }

    static <U> Option<U> of(U value) {
        return new Option<>(value);
    }

    T orElse(T other) {
        return value == null ? other : value;
    }

    <U> Option<U> map(Function<T, U> f) {
        return value == null ? empty() : of(f.apply(value));
    }
}
```