Exercices avancés sur les structures récursives

Exercice 8.3.1 *utilisation de listes*

Nous allons utiliser la classe `ListeEntier` suivante.

```
public class ListeEntier {
    private int valeur;
    private ListeEntier suivant;
    public ListeEntier(int val, ListeEntier suiv) {
        this.valeur = val;
        this.suivant = suiv;
    }
    public ListeEntier(int val) {
        this.valeur = val;
        this.suivant = null;
    }
    public int getValeur() {
        return valeur;
    }
    public void setValeur(int val) {
        this.valeur = val;
    }
    public ListeEntier getSuivant() {
        return suivant;
    }
    public void setSuivant(ListeEntier suiv) {
        this.suivant = suiv;
    }
}
```

1. Ecrire une méthode qui calcule la somme des éléments positifs d'une liste d'entiers.
2. Ecrire une méthode qui vérifie si une liste est triée en ordre croissant.

Exercice 8.3.2 *égalité de structures récursives*

Il existe deux notions d'égalités de structures récursives : l'égalité absolue, testée par == qui teste en fait si deux expressions Java sont deux moyen de parler de la même structure, du même objet.

Le plus souvent, on s'intéresse à une égalité du contenu des structures : deux structures sont égales si

– elles ont le même nombre d'éléments
– pour toute position dans la structure, les objets des deux structures situé à cette position ont des variables égales.

Pour cet exercice, on va utiliser une classe permettant de représenter le classement d'une course à pied. On suppose qu'un coureur (ou une coureuse) est représentée par un nom et un numéro de dossard. Les numéros de dossard peuvent être différents d'une course à l'autre : on ne doit pas les prendre en compte dans la comparaison des classements.

```
public class Coureur{
    private String nom;
    private int dossard;
    private Coureur suivant;
    public Coureur(String n, int d, Coureur suiv) {
        this.nom = n;
        this.dossard = d;
        this.suivant = suiv;
    }
    public String getNom() {
        return nom;
    }
    public int getDossard() {
        return dossard;
    }
    public Coureur getSuivant() {
        return suivant;
    }
    public void afficheClassement(){
        int rang = 1;
        Coureur aux = this;
        while (aux != null){
            Terminal.ecrireStringln(rang + ": " + aux.getDossard() + " " +
                                    aux.getNom());
            rang++;
            aux = aux.getSuivant();
        }
    }
}
```

1. écrire une méthode qui teste si deux courses ont le même classement.
2. écrire une méthode qui teste si deux courses ont le même podium (trois premiers arrivés).

Exercice 8.3.3 *voyage*

On veut représenter un itinéraire comme une suite de moyens de transports publics à emprunter (métro, bus, train, autocar, avion) avec éventuellement des transitions à pied. Sur cet itinéraire, il y a des lieux représentés par le nom de la ville et un complément qui peut-être le nom d'une gare, d'une station de métro, d'un arrêt de bus ou une adresse. On va représenter les lieux au moyen d'objets d'une classe des lieux et les liaisons entre lieux au moyen d'objets d'une autre classe. Une liaison comprendra le type de transport, la distance et éventuellement, les heures de départ et d'arrivée si cela

a un sens pour le moyen de transport considéré. Par exemple, un train ou un avion ont des horaires précis, mais un métro ou la marche à pied n'ont pas d'horaire précis.

Un itenéraire va être une structure contenant alternativement des lieux et des liaisons (un lieu, une laison, un lieu, une liaison, un lieu, etc).

Ecrivez les classes nécessaires avec une méthode d'affichage d'un itinéraire. Ecrivez une méthode main pour prendre le cas d'une marche à pied du 3 rue Conté, 75003 Paris à la station de métro République (Paris), puis la ligne 5 jusqu'à gare du Nord et le train de 18H17 de Paris à Lille, avec affichage de cet itinéraire.