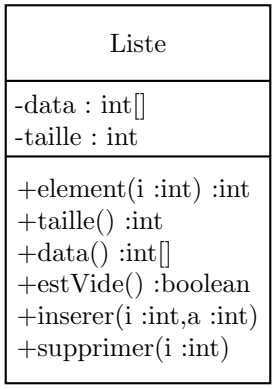
# TP n° 2: Classes et objets

## 1 Partie 1 : Implémentation des listes à l'aide des tableaux statiques

Une liste est une strucure qui permet de regrouper et de structurer un ensemble de données pouvant être de même ou de type différent.
Nous voulons modéliser une structure de liste en utilisant les tableaux **statiques** de taille fixe, en Java.

Soit la représentation UML, de la classe `Liste`, suivante :




Définir la classe Liste qui regroupe un ensemble de valeurs entières, suivant le diagramme de classe précédent. Les méthodes sont définies comme suit :

1. `element` retourne la valeur de la ième position.
2. `taille` retourne la taille du tableau de données.
3. `data` retourne le tableau de données
4. `estVide` vérifie si le tableau de données est vide.
5. `inserer` permet d'insérer un entier `a` à la ième position.
6. `supprimer` permet de supprimer l'élement de la ième position.

Un objet de la classe `Liste` peut être créé selon quatre différents manières :

1. Créer une liste de donnée vide d'une taille égale à 0.
2. Créer une liste de donnée vide mais d'une taille donnée.
3. Créer une liste de donnée à partir d'un tableau d'entier
4. Créer une liste de donnée à partir d'un tableau de chaîne de caractères.

Dans la méthode principale `main`, définie dans la classe Tp2, exécuter les instructions suivantes :

1. Créer un objet `liste1`, une liste vide de taille 0.
2. Ajouter un à un les éléments suivants à `liste1` : 1, 2, 3, 4.
3. Afficher la taille de `liste1`.
4. Ajouter l'élement 5 à la troisième position dans `liste1`.
5. Afficher la nouvelle taille de `liste1`
6. Créer une nouvelle liste `liste2`, à partir du tableau suivant : [6, 5, 4, 3, 2, 1].
7. Afficher la taille de `liste2`.
8. Ajouter l'élément 7 au début de `liste2`.
9. Afficher la nouvelle taille de `liste2`.
10. Supprimer le dernier élément de `liste2`
11. Créer une nouvelle liste `liste3` à partir du tableau suivant : ["1", "6", "2", "5", "3", "4"].
12. Afficher le contenu de `liste1`, `liste2` et `liste3`.

## 2 Partie 2 : Les listes triées

On veut utiliser l'algorithme de tri implémenté dans TP n° 1 pour trier les listes définies dans la première partie.

Dans la classe Liste :

1. Ajouter une variable booléenne `trie`, initialisée à `false`, qui permet de savoir si la liste est triée.
2. Définir la méthode `public void estTriee()` qui permet de vérifier si la liste est triée.
3. Définir une méthode `public void trier()` qui implémente un algorithme de tri pour ordonner les valeurs de la liste, si celle-ci n'est pas triée.
4. Définir la méthode `public void insererTri(int a)` qui permet d'ajouter l'élement `a` à la liste, si elle est triée, selon l'ordre de tri.
5. **Question bonus :** Si la liste n'est pas triée, la méthode `insererTri` doit relever une **Exception**