

# TP 4 : Arbres Binaires

## Objectif du TP

L'objectif de cette séance est de développer des structures de données récursives (majoritairement des *arbres*) et des algorithmes les manipulant.



## EXERCICES SUR PAPIER !

Vous **devez** répondre aux exercices suivants **sur papier** et y travailler **seul**. Cela évite le bavardage, donc le bruit et favorise grandement la **concentration** des autres. Une fois que vous avez **fini un exercice**, vous vous manifestez auprès de votre enseignant pour qu'il **juge** votre travail sur une échelle de 0 à **4 points** (0=aucun travail, ..., 4=exercice complètement juste sans assistance de l'enseignant).

### Exercice 1 Code Morse



L'alphabet morse ou code morse, est un code permettant de transmettre un texte à l'aide de séries d'impulsions *courtes* (symbolisée ici par •) et *longues* (symbolisées ici par —), qu'elles soient produites par des signes, une lumière, un son ou un geste. Exemple : le mot « S.O.S. » sera traduit en morse par •••—•••.

Plutôt que d'utiliser une table qui associe un caractère à son code morse, on souhaite construire un arbre binaire dont les nœuds sont des caractères et de telle sorte que si on passe par le fils gauche, on ajoute un — au code ou si on passe par le fils droit, on ajoute un • au code. Dans le cas où on aurait besoin de créer un nœud dans l'arbre qui ne correspond à aucune lettre, on choisira l'espace.

- Sans considérer les chiffres, dessiner (sur papier) l'arbre qui représente l'alphabet morse suivant nos hypothèses.
- Ecrivez le code qui permet de construire cet arbre en mémoire en se basant sur le constructeur qui utilise un tableau. Indice : il faudra utiliser la classe `Character`.
- Proposer une méthode récursive `decodeLetterInMorse` qui étant donné une chaîne de caractères composé de '.' et de '-', renvoie le caractère de notre alphabet correspondant en utilisant l'arbre binaire précédent.
- En supposant que les lettres sont séparées par le caractère '/' par exemple et les mots par l'espace, déduisez de la question précédente une nouvelle méthode récursive `decodeInMorse` qui permet de décoder un message en morse et renvoie une chaîne de caractères de notre alphabet. Exemple : `decodeInMorse (".-/..-/--- ./...")= "ALGO AV"`.
- Proposer une méthode récursive qui trouve le code morse d'un caractère de notre alphabet.
- Proposer une nouvelle méthode récursive qui trouve le code morse d'un caractère de notre alphabet mais sans utiliser `contains`. Indices : il faut à la fois parcourir l'arbre et construire en même temps la chaîne de caractères qui devra être considérée comme une `ArrayList<Character>`. La méthode est une fonction qui renvoie un booléen qui précise si le caractère a été trouvé dans l'arbre.



## EXERCICES SUR MACHINES !

À partir de maintenant, vous **pouvez** (je n'ai **pas** dit *devez* !) implémenter l'exercice sur machine. Il est fortement conseillé de résoudre le problème sur papier avant de l'implémenter dans le langage de votre choix (C, D, Java, Python, Julia, ... [https://fr.wikipedia.org/wiki/Liste\\_de\\_langages\\_de\\_programmation](https://fr.wikipedia.org/wiki/Liste_de_langages_de_programmation)).

### Exercice 2 Implémentation de ABin

- Terminer l'implémentation ainsi que toutes les méthodes vues de ABin.
- Tester Code Morse !



## POUR ALLER PLUS LOIN

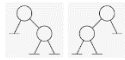
### Exercice 3 ANaire


Créer une classe d'arbre N-aire qui permet de stocker plusieurs fils pour chaque nœud plutôt que 2 fils uniquement.

### Exercice 4 ABR

- Créer la classe ABR vue en cours.
- Créer une méthode d'insertion `add` qui insère une valeur au bon endroit dans l'arbre.
- Créer le constructeur qui construit un arbre à partir d'un tableau de valeurs.
- Créer la méthode récursive `contains` qui précise si une valeur est déjà dans le tableau.
- En utilisant la javadoc, comparer votre class avec `TreeSet<E>`.

### Exercice 5 Liste d'arbres

On souhaite construire un programme récursif qui construit la liste de TOUS les arbres à  $n$  nœuds ( $n$  étant un paramètre). Par exemple, si on souhaite l'ensemble des arbres possibles de taille 2, on obtient  et à 3

nœuds, on obtient 

- Identifiez à la main, tous les arbres à 4 nœuds.
- Identifiez la récurrence.
- Programmer la méthode `arbresDeNNoeud(int n)` qui renvoie une liste d'arbres de  $n$  nœuds.

### Exercice 6 Itérateurs d'arbres

Comme pour les listes, on souhaite créer des itérateurs sur des arbres. Néanmoins, il existe plusieurs parcours possibles pour un même arbre. Proposez plusieurs solutions pour réaliser cet objectif.

## Sources pour ce TP