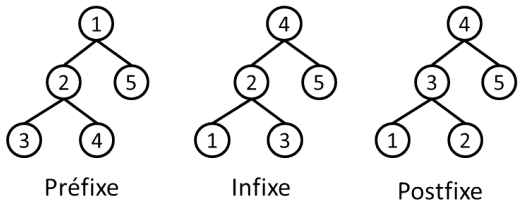
TD6

Itérateur

1. Récupérez dans le l'archive `tree.jar` la classe `Tree<V>` qui représente un arbre binaire dont les nœuds contiennent des valeurs de type `V`. Notez que chaque nœud contient des références vers son parent, son sous-arbre gauche et son sous-arbre droit. Cette classe implémente l'interface `java.lang.Iterable<V>` qui contient la méthode de fabrique `iterator()`, et peut donc être utilisée avec la nouvelle syntaxe de la boucle `for`.
2. Réalisez trois itérateurs (implémentant l'interface `java.util.Iterator<E>`) qui parcourent respectivement les nœuds d'un arbre dans l'ordre infixe, préfixe et postfixe. La figure ci-dessous illustre l'ordre dans lequel les nœuds de l'arbre doivent être parcourus selon l'itérateur.



3. Écrivez un programme de test qui parcourt différents arbres avec les trois itérateurs.

Visiteur

Une expression booléenne est constituée d'opérateurs logiques et de constantes booléennes (`true` et `false`). Par exemple, l'expression booléenne `((true Or false) And (Not true))` vaut `false`. Une telle expression peut être représentée sous la forme d'une arbre où chaque nœud de l'arbre est soit un opérateur logique, soit une constante booléenne. Les nœuds représentant un opérateur logique possèdent un nombre de nœuds fils égal à l'arité de l'opérateur (*i.e.*, son nombre d'arguments). Dans la suite de cet exercice, nous considérons uniquement les deux opérateurs binaires `And` (conjonction) et `Or` (disjonction), ainsi que l'opérateur unaire `Not` (négation).

1. Récupérez dans l'archive `tree.jar` le *package* `boolexpr` qui contient des classes permettant de représenter les différents nœuds d'un arbre d'une expression booléenne.
2. Modifiez les classes du diagramme afin d'y ajouter les opérations nécessaires à l'affichage et à l'évaluation d'une expression booléenne.
3. Quels sont les inconvénients de cette modélisation, en particulier par rapport à l'ajout de nouvelles opérations sur les expressions booléennes ?
4. Proposez un diagramme de classes qui modélise et résout le problème en utilisant le pattern *Visiteur*.
5. Écrivez une implémentation des classes du diagramme.
6. Réalisez deux visiteurs permettant respectivement d'afficher et d'évaluer une expression booléenne.
7. Écrivez un programme de test qui affiche et évalue différentes expressions booléennes.