TP2

GL52

stephane.galland@utbm.fr

Implantation d’un arbre binaire et d’un arbre binaire de recherche

00 | Objectif

L'objectif de cette session de TP est de créer un structure d'arbre d'information en appliquant les principes d'implantation issus du génie logiciel.

La configuration d'Eclipse doit être la même que pour les sessions précédentes.

Ce sujet de travaux pratiques sera réalisé durant

DEUX SEANCES

de travaux pratiques.

01 | Exercices

01.1 Réalisation d'une implantation d'un arbre générique

Une arbre est une structure de données informatique qui peut se représenter sous la forme d'une hiérarchie dont chaque élément est appelé nœud (node), le nœud initial étant appelé racine. Dans un arbre, chaque élément possède au plus n éléments fils au niveau inférieur. Du point de vue de ces éléments fils, l'élément dont ils sont issus au niveau supérieur est appelé père. Chaque nœud possède également une référence sur son nœud père (le nœud au dessus de lui dans la hiérarchie), à l'exception du nœud racine qui ne possède aucun nœud père.

Un arbre binaire est un arbre pour lequel n = 2.

Dans le cadre de ce TP, nous ne considérerons que un arbre contient de l'information, ses données. Ces données n'ont pas de type prédéterminé. Ce type doit donc être considéré comme un type générique (T).

Le diagramme de classes UML ci-dessous vous donnera la structure et les relations entre les types à implanter à la fin du TP.

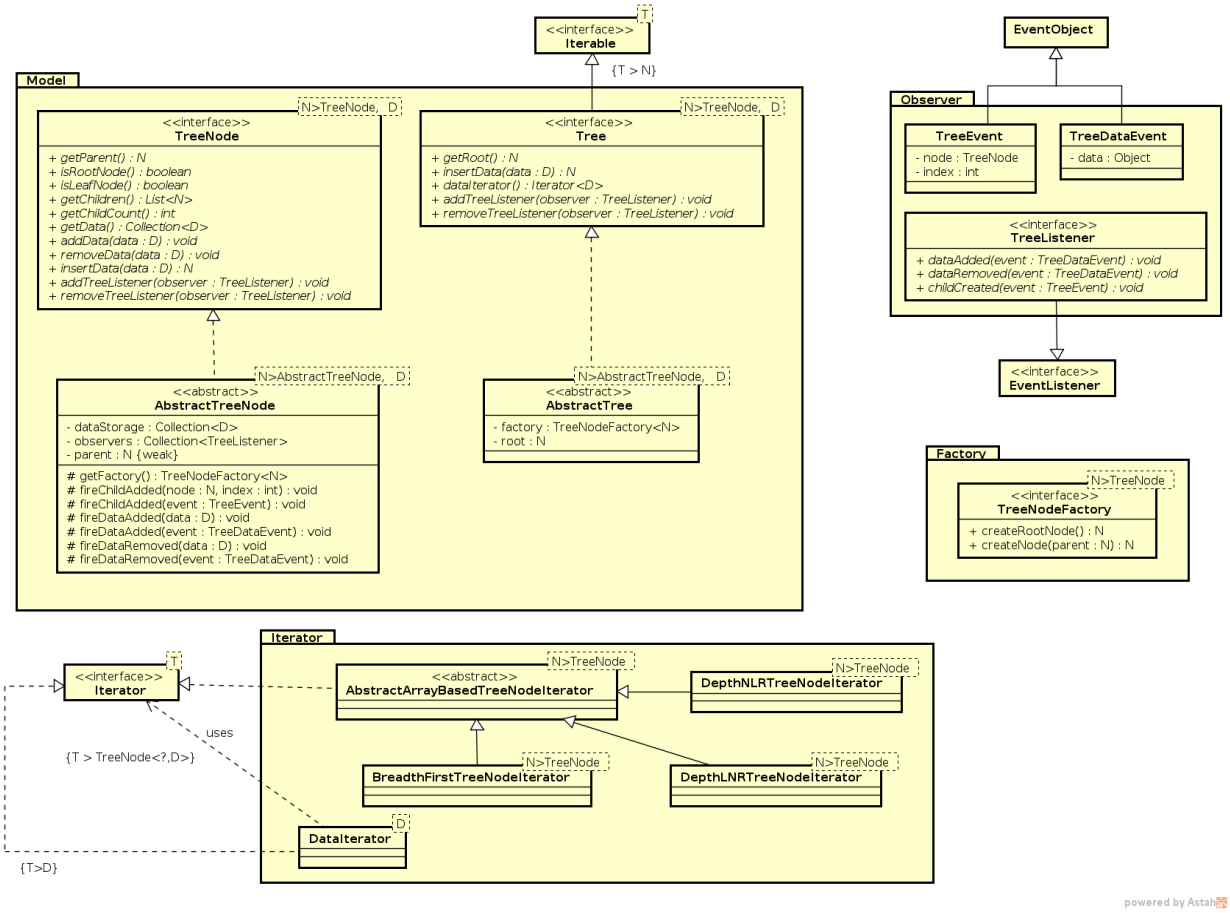

Figure 1 : Diagramme de classes pour les arbres génériquesVous devrez respecter les bases des principes de la programmation inspirées des bonnes pratiques du génie logiciel, à savoir :

1. Utiliser des noms clairs et en anglais, car votre code doit pouvoir être lu par tout développeur, quelque soit son langage maternel.

2. Corriger le code pour qu'il n'y ait plus d'erreur ni de warning car chaque erreur ou warning indique une anomalie en relation avec les bonnes pratiques de programmation.

3. Ecrire une documentation claire et en anglais (vous aurez des warnings pour vous aider).

4. Appliquer le plus possible les design patterns : https://fr.wikipedia.org/wiki/Patron_de_conception

Travail à faire:

A) Structure d'arbre générique

- Écrire les interfaces Tree, TreeNode représentant un arbre et ses noeuds. Ces types doivent prendre 2 paramètres génériques :

  (a) N : représente le type des nœuds

  (b) D : représente le type des données stockées dans les nœuds

- Il existe une différence entre les fonctions addData et insertData :

(a) addData(data : D) ajoute la donnée dans le nœud courant (sur lequel est appelé la fonction). Elle ne tente pas d'ajouter la donnée dans un autre nœud.

(b) insertData(data : D) ajoute la donnée dans le sous-arbre dont la racine est le nœud courant. En d'autres termes, cette fonction ajoute la donnée soit dans un nœud fils, soit dans un nœud courant, selon la sémantique et le type d'arbre implanté (binaire, ternaire, arbre de recherche, etc.à

- Écrire les classes abstracts correspondantes à chacune des deux interfaces implantées ci-dessus.

B) Design Pattern : Factory

- Le patron de conception « Factory » permet d'externaliser la création d'objets (operateur *new*) dans une classe particulière appellée *Factory.*

- Définissez l'interface TreeNodeFactory.

- Intégrer un champs contenant la factory dans les classes spécifiées dans A).

C) Design Pattern : Observateur

- Écrire l'interface TreeDataEvent comme un objet contenant des informations correspondant à une modification des données dans l'arbre.

- Écrire l'interface TreeEvent comme un objet contenant des informations correspondant à une modification de la structure de l'arbre.

- Écrire l'interface TreeListener représentant un observateur sur l'arbre ou sur un nœud de l'arbre. Les événements à prendre en compte sont :

  (a) l'ajout d'une donnée dans un nœud

  (b) la suppression d'une donnée d'un nœud

  (c) la création d'un nœud fils.

- Intégrer le patron de conception « Observateur » dans la structure de l'arbre et des noeuds. Les observateurs peuvent observer un nœud, ou l'arbre dans sa totalité.

D) Design Pattern : Iterateur

- Deux implantation du patron de conception « Iterateur » sont nécessaire : une pour itérer sur les nœuds de l'arbre, et une seconde pour itérer sur les données stockées dans l'arbre.

- Itération sur les nœuds dans l'arbre : il existe différents algorithmes pour parcourir les nœuds d'un arbre (https://en.wikipedia.org/wiki/Tree_traversal)

  (a) Créer l'itérateur de parcours en largeur d'abord (breadth first), qui retourne les nœuds étage par étage du haut (la racine) vers le bas (les feuilles). A chaque appel de l'itérateur, un nœud est retourné.

  (b) Créer l'itérateur de parcours en profondeur d'abord avec pré-ordre (NLR) qui retourne le nœud père avant ses nœuds fils.

  (c) Créer l'itérateur de parcours en profondeur d'abord avec ordre infixé (LNR) qui retourne le nœud père après sont nœud gauche et avant son nœud droit. Cet itérateur ne fonctionne que pour les arbres binaires.

- Créer l'itérateur sur les données stockées dans l'arbre en utilisant une référence vers un itérateur sur les nœuds. En effet, pour parcourir l'ensemble des données dans un arbre, il faut tout d'abord itérer sur les nœuds de l'arbre eux même, car les données sont stockées dans les noeuds.

01.2 Réalisation d'une implantation d'un arbre binaire

Un arbre binaire est un arbre pour lequel n = 2.

La partie supérieure du diagrammes de classes UML ci-dessous vous donnera la structure et les relations entre les types à implanter à la fin du TP concernant l'exercice 01.2.

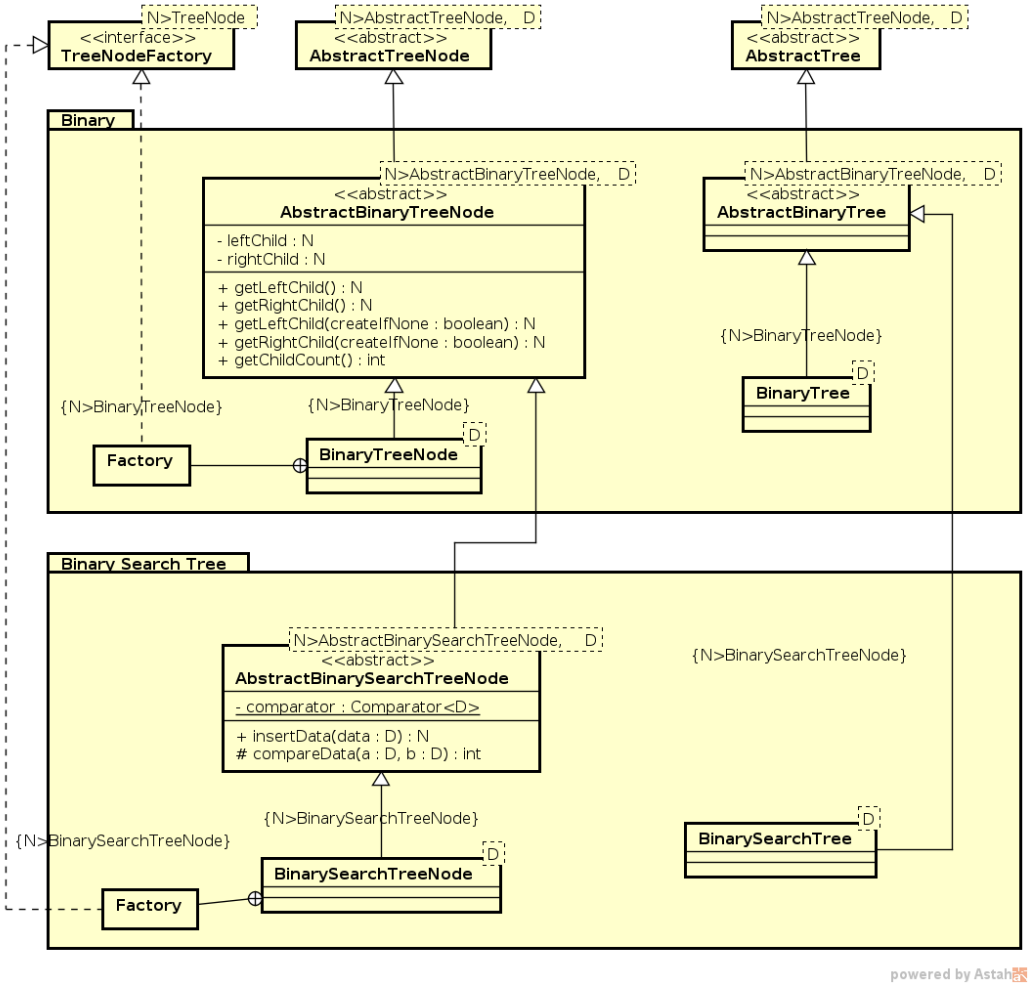

Figure 2 : Diagramme de classes pour les arbres binaires

A) Structure d'arbre binaire

- Créer les classes abstraites communes à toutes les implantation d'abres binaires : AbstractBinaryTreeNode, AbstractBinaryTree.

- Créer les classes concrètes pour un arbre binaire : BinaryTreeNode et BinaryTree.

- Créer la Factory de BinaryTreeNode comme une « inner class » de la classe BinaryTreeNode et l'utiliser pour créer les nœuds d'un arbre binaire.

01.3 Réalisation d'une implantation d'un arbre binaire de recherche

Un arbre binaire de recherche est un arbre binaire dont chaque nœud contient une donnée et les données stockées dans la branche gauche sont inférieures à la donnée stockée dans la racine et les données stockées dans la branche droite sont strictement supérieures à la donnée stockée dans la racine. Cette notion d'ordre est répétée récursivement pour chaque nœud de l'arbre (https://fr.wikipedia.org/wiki/Arbre_binaire_de_recherche)

A) Structure d'arbre binaire de recherche

- Créer la classe abstraite AbstractBinarySearchTreeNode incluant les outils permettant de comparée des données selon leur ordre naturel (entier, etc.) ou à l'aide d'un comparateur (outils de l'API Java). Ces outils serviront de support à la notion d'ordre intrasect aux arbres binaires de recherche.

- Créer les classes concrètes pour un arbre binaire de recherche : BinarySearchTreeNode et BinarySearchTree.

- Créer la Factory de BinarySearchTreeNode comme une « inner class » de la classe BinarySearchTreeNode et l'utiliser pour créer les nœuds d'un arbre binaire de recherche.

B) Programme de test

- Ecrire une function « main » réalisant les actions suivantes :

(a) création d'un arbre binaire de recherche de nombres entiers

(b) ajout de nombres entiers aléatoires dans l'arbre.

(c) Affichage la liste ordonnée des entiers stockés dans l'arbre en utilisant l'un des itérateurs implantés dans les 01.1.

A LA FIN DES DEUX SEANCES DE TP, ENVOYEZ UN FICHIER ZIP CONTENANT LE CODE JAVA QUE VOUS AVEZ PRODUIT DURANT LE TP EN UTILISANT LA PLATE-FORME MOODLE.UTBM.FR