SINF1121 – Algorithmique et structures de données

Mission 2

Arbres et arbres binaires

1 Contexte général de la mission

La nouvelle calculatrice `Icalc3D` pour tablette tactile `Ijoke` est en voie de conception et devrait, dans un avenir proche, bénéficier d'un succès considérable auprès des utilisateurs. Cette tablette dispose de nombreuses caractéristiques intéressantes comme un écran couleur à haute définition, une capacité mémoire de 128 Giga-octets, un contrôle par GPS, Bluetooth, WIFI, 3G, reconnaissance digitale, faciale et génomique (test de paternité, en option).

L'application `Icalc3D` inclut de nombreuses fonctions de calcul scientifique, notamment statistique et graphique, et la possibilité d'être programmée en Java. Parmi ses fonctionnalités alphanumériques, la calculatrice `Icalc3D` permettra également de faire de la *dérivation formelle*. Vous êtes en charge de la conception et de la mise en oeuvre de cette fonctionnalité.

Le type abstrait ***arbre*** permet notamment de représenter et de manipuler une expression analytique. La mise en oeuvre d'un programme de dérivation formelle sera donc l'occasion de se familiariser avec ce type abstrait ainsi que certains algorithmes de parcours et de transformation d'arbres.

2 Objectifs poursuivis

À l'issue de cette mission chaque étudiant sera capable :
- de ***décrire*** avec exactitude et précision les concepts présents dans le chapitre du livre de référence qui traite de ***structures arborescentes***,
- de ***mettre en oeuvre*** des algorithmes basés sur les types abstraits `Arbre` (**`Tree`**) et `Arbre Binaire` (**`BinaryTree`**),
- d'***évaluer*** et ***mettre en oeuvre*** les différentes représentations classiques d'arbres et d'arbres binaires,
- concevoir et adapter différents ***algorithmes récursifs de parcours*** d'un arbre.

3 Prérequis

Les conditions à remplir pour pouvoir aborder cette mission sont :
- avoir rempli avec succès la mission précédente,
- se débrouiller en anglais technique et scientifique (lecture).

4 Ressources

- Livre : *Data Structures and Algorithms in Java*, Goodrich and Tamassia, (*DSAJ-5*) : chapitre 7 [1]
- Document : *Spécifications en Java, Préconditions et postconditions : pourquoi ? comment ?*, P. Dupont.
  Sur le site iCampus, suivre `Documents et liens/pdf/specif.pdf`

5 Calendrier

- lundi 06 octobre : démarrage de la mission
- vendredi 10 octobre, avant ***18h00*** : ***envoi réponses questions***
- lundi 13 octobre : séance intermédiaire
- vendredi 17 octobre, avant ***18h00*** : ***remise des produits***
- lundi 20 octobre : séance de bilan

6 Rappel sur la marche à suivre

Vous répartirez durant la séance de mise en route les questions entre les différents membres du groupe de telle sorte que chaque étudiant soit responsable d'au moins une question. Chaque étudiant préparera la réponse à la (aux) question(s) dont il est responsable. Ceci n'empêche pas de s'informer des réponses apportées aux autres questions d'autant plus que certaines questions sont liées entre elles. En outre, chaque étudiant se préparera à débattre avec les autres membres du groupe de toutes les questions.

7 Questions

1. Qu'entend-on par les notions de *hauteur*, de *profondeur* et de *niveau* dans le contexte de structures arborescentes ? Décrivez précisément ces notions et les liens éventuels entre elles ? Ces notions dépendent-elles du fait que l'on parle d'arbres binaires ou d'arbres en général ? Ces notions dépendent-elles de la structure de données utilisée pour représenter des arbres ? Justifiez vos réponses.
2. Un arbre dont chaque noeud possède au plus deux fils est-il nécessairement binaire ? Qu'entend-on par arbre ordonné ? L'ordre dépend-il des valeurs mémorisées dans l'arbre ? Un arbre binaire impropre est-il désordonné ?
   Si l'on s'intéresse à des arbres dont la profondeur maximale est connue et fixée, y a-t-il une structure de données particulièrement bien adaptée à ce cas ? Si oui laquelle, sinon pourquoi ? Cela dépend-il du fait que l'arbre soit binaire ou non ? Cela dépend-il des opérations effectuées sur l'arbre ?
3. Qu'est-ce qu'un arbre *équilibré* ? Un arbre binaire équilibré est-il nécessairement propre ? Si oui, démontrez pourquoi, sinon donnez un exemple d'arbre équilibré impropre ? Comment définiriez-vous ce qu'est un arbre binaire *(essentiellement) complet* ? Pourquoi, à votre avis, est-il utile de préciser "(essentiellement)" ? Un arbre complet est-il toujours équilibré ? Un arbre équilibré est-il toujours complet ? Justifiez vos réponses.

1. chapitre **8** dans *DSAJ-6*

4. Qu'entend-on par une implémentation d'un arbre par une *structure chaînée* ? En quoi cette notion de *structure chaînée* est-elle différente (ou plus générale) par rapport à celle de *liste chaînée* ? Quels sont les points communs entre liste et structure chaînée ? Quelle est la classe qui implémente un arbre par une structure chaînée dans *DSAJ-5* ? Serait-il possible de remplacer cette implémentation par une autre utilisant une liste chaînée ?

5. On considère une variante de la représentation d'un arbre binaire en structure chaînée (*DSAJ-5*, pages 301 et suivantes [2]). Dans cette variante, un noeud ne contient pas de référence vers son parent, ni de méthodes `parent` ou `iterator`. Ce fait constitue-t-il un problème pour réaliser des parcours de l'arbre binaire ? Pourquoi ?
   Donnez un algorithme pour réaliser la méthode `parent` sur base des autres méthodes disponibles dans l'interface. Si toutes les méthodes déjà disponibles s'exécutent en temps constant, quelle est la complexité temporelle de votre algorithme ?

6. Un arbre binaire peut également être défini *récursivement* comme suit. Un arbre binaire est :
   – soit vide (sans aucun noeud),
   – soit il contient un noeud racine, un fils gauche qui est un arbre binaire et un fils droit qui est un arbre binaire.

   On considère l'interface **`RBinaryTree`** qui spécifie les méthodes essentielles d'un arbre binaire conformément à cette définition. Celle-ci fait appel à l'interface **`Position`** décrite dans *DSAJ-5*. Le code de ces deux interfaces est disponible sur le site iCampus, suivre `Documents et liens/missions/m2/`.
   Écrivez en Java la classe **`LinkedRBinaryTree`** qui implémente l'interface **`RBinaryTree`**.

7. Une expression *arithmétique* peut contenir les quatre opérateurs fondamentaux $+, -, *, /$ et des scalaires, comme par exemple : $3 * 10 - 2/4$. Une expression *analytique*, telle que $x^2 + x \sin x - 3$ peut contenir également des variables $x, y, \ldots$, d'autres opérateurs comme la fonction puissance entière ^ ou d'autres fonctions mathématiques comme $\sin$ ou $\cos$.

   Une expression arithmétique peut être représentée par un arbre. Quelles sont les caractéristiques de cet arbre ? Pourquoi cette représentation est-elle utile ? Citez deux exemples de manipulation d'une expression arithmétique et exprimez comment ces manipulations sont mise en oeuvre à l'aide de cette représentation. Quelles sont les caractéristiques supplémentaires pour un arbre représentant une expression analytique ?

8. En supposant qu'un arbre représente une expression analytique, quel type de parcours de cet arbre permet-il de construire l'expression analytique complètement parenthésée qui lui correspond ? Donnez le pseudo-code d'une méthode `public String toString()` qui réalise cette construction.

9. Représentez graphiquement toutes les opérations de dérivation formelle (voir section 8) comme des opérations de manipulation d'un arbre et donnez une description sous forme de pseudo-code de ces opérations.

2. pages 297 et suivantes dans *DSAJ-6*

Les réponses aux questions doivent être soumises sur iCampus ***avant*** la séance ***intermédiaire***[a]. Cela suppose une étude individuelle et une mise en commun en groupe (sans tuteur) préalablement à cette séance. Un document (au format ASCII ou PDF) reprenant les réponses aux questions devra être soumis sur iCampus ***au plus tard*** pour le vendredi 10 octobre à ***18h00***. Les réponses seront discutées en groupe avec le tuteur durant la séance intermédiaire. Ces réponses ne doivent pas explicitement faire partie des produits remis en fin de mission. Néanmoins, si certains éléments de réponse sont essentiels à la justification des choix d'implémentation et à l'analyse des résultats du programme, ils seront brièvement rappelés dans le *rapport de programme*.

a. à l'exception, le cas échéant, de question(s) spécifiquement liée(s) au problème traité.

8 Problème

L'opération de dérivation formelle consiste à calculer à partir d'une expression analytique, telle que $x^2+x\sin x$, l'expression analytique de sa dérivée : $2x+\sin x+x\cos x$. Un certain nombre d'hypothèses simplificatrices sont suggérées :

- on considère des expressions analytiques qui, outre les 4 opérateurs arithmétiques, représentés respectivement par `+`, `-`, `*`, `/`, ne peuvent contenir que des entiers, une variable représentée par `x`, la fonction puissance entière représentée par `^` et les fonctions $\sin$ et $\cos$ représentées respectivement par les identificateurs `sin` et `cos`.
- on considère des expressions analytiques complètement parenthésées, comme par exemple :

```
10
(10+x)
sin(x)
((10*2)+(4-x))
(10*(x+sin(x)))
(x+cos((x+2)))
((x^3)/5)
```

Pour rappel, ci-dessous quelques règles de dérivation :

expression	dérivée
$f+g$	$f'+g'$
$f-g$	$f'-g'$
fg	$gf'+fg'$
f/g	$(gf'-fg')/g^2$
$\sin(f)$	$f'\cos(f)$
$\cos(f)$	$-f'\sin(f)$
f^a	$af^{a-1}f'$

Écrivez un programme qui prend en entrée une expression analytique et renvoie sa dérivée. Pour ce faire, définissez une classe implémentant l'interface **`FormalExpressionTree`**[3]. Vous êtes totalement libres d'adapter/d'étendre une implémentation d'un

arbre décrite dans *DSAJ-5* ou d'écrire votre propre implémentation. Une possibilité à considérer serait d'étendre votre implémentation de l'interface **`RBinaryTree`** (voir question 6). ***Quelque soit votre choix d'implémentation il est important de pouvoir justifier son intérêt et ses éventuelles limites.*** Testez votre programme sur les expressions contenues dans le fichier `expression.txt` [3].

Comment tirer parti dans votre implémentation des hypothèses simplificatrices mentionnées ? Serait-il possible d'éviter de faire certaines de ces hypothèses ? Que devriez-vous adapter dans votre implémentation ? Comment feriez-vous pour considérer également des expressions analytiques contenant les fonctions exp et log ? En quoi votre conception facilite-t-elle ce genre d'extension ?

Indications pour la bonne réussite de la mission

- Les réponses aux questions posées ne se trouvent pas toutes dans le chapitre du livre de référence concerné par cette mission. N'oubliez pas de consulter l'index de cet ouvrage ou toute autre ***ressource complémentaire*** mentionnée dans ce document ou sur le site WEB du cours.

- Veillez à citer précisément vos sources lorsque vous répondez aux questions.

- Veillez à vous concentrer d'abord sur les questions avant de passer au problème à résoudre. Veillez à y répondre de manière précise et concise. Quand vous attaquez le problème, gardez à l'esprit les concepts abordés dans les questions.

- Pendant le travail d'analyse du problème et de conception d'une solution, l'écriture d'un ***diagramme de classes*** est très utile pour
  - vérifier l'exactitude, la cohérence et la complétude du découpage en plusieurs classes,
  - préciser les liens entre les classes concrètes, les classes abstraites et les interfaces,
  - vous ***répartir*** le travail afin que chaque étudiant soit responsable de la programmation d'une ou plusieurs classe(s).

- Chaque étudiant d'un groupe doit être en charge d'une partie du travail de programmation. Vous indiquerez ***toujours*** en commentaires d'une classe, le ou les auteur(s) de la classe.

- Les fragments de code extraits du livre de référence sont mis à votre disposition sur le site WEB du cours. Vous n'êtes pas obligés de les utiliser et vous êtes totalement libres de réimplémenter certaines classes déjà écrites dans ce livre si vous le désirez. Cela suppose néanmoins que :
  - vous répondiez effectivement au problème tel qu'il est formulé, en particulier en respectant les interfaces qui vous sont communiquées.
  - vos nouveaux choix d'implémentation ne vous éloignent pas de l'objet proprement dit de la mission,

3. Disponible sur le site iCampus, suivre `Documents et liens/missions/m2/`.

  - vous soyez capables de justifier ce qui vous semble améliorable dans les classes proposées dans le livre de référence et en quoi votre implémentation est meilleure.

- Chaque étudiant est responsable de la bonne organisation de la mission et de l'équilibre entre son travail personnel et sa participation active au groupe.

9 Remise des produits

Les produits de la mission (***sources***[a] des programmes Java et leur documentation) sont à soumettre sur iCampus pour le vendredi 17 octobre, à ***18h00*** **dernier délai**. Le tuteur est déchargé de toute correction pour tout produit non remis dans ce délai. Le rapport de programme[b] contenant un commentaire (1 ou 2 page(s) max.) sur la solution apportée au problème doit être disponible au format `PDF`[c]. Les produits de l'ensemble du groupe doivent être soumis sur iCampus en ***un seul*** fichier archive (au format `zip` ou `tar.gz`).

a. ***PAS de*** `.class`
b. incluant les réponses aux questions spécifiquement liées au problème traité.
c. en ***un seul*** document.