Université de Technologie de Compiègne - Printemps 2012

Hamici Mathilde
Suzanne Aurélie

Projet LO21 : Calculatrice à notation polonaise inversée

Choix de conception

Mode complexe

Lorsque le mode complexe est activé toutes les constantes sont traduites en complexe, on peut donc rentrer des entiers, réels et rationnels, mais ceux-ci seront traduits en complexe. Au contraire, lorsque ce mode n'est pas activé, tous les complexes entrés sont supprimés.

Lorsque l'on active le mode complexe, toutes les constantes déjà présentes dans la pile sont transformées en complexe. Lorsqu'on le désactive elles sont toutes supprimées, car il n'existe pas de conversion convenable.

Mode type de constante

Les types de constantes sont utilisés lors de l'évaluation d'une expression. Les opérateurs +, -, *, /, pow, mod, sign, sqr, cube, fact renverrons un résultat du type sélectionné par l'utilisateur.

Inv renverra un rationnel ou un réel, le type entier est ignoré.

Les opérations trigonométriques renverront toujours un réel. Ainsi que les fonctions ln, log, sqrt.

Pour les opérations sur la pile sum renvoie une constante du type selectionné par l'utilisateur, et mean renverra un rationnel si le type entier est selectionné, et sinon le type demandé.

Implémentation du LogMessage

Le LogMessage peut générer un message dans un fichier ou un message dans la console (et une QMessageBox suivant l'importance du message). Dans tous les cas, les deux seront toujours, dans le fonctionnement choisi, générés en même temps. Les logger générant les LogMessages sont déclarés comme globaux afin de pouvoir y accéder de partout dans le code.

Domaine de définition des fonctions

Certaines fonctions possèdent un domaine de définition limité. Ainsi lorsque l'on rentrera une constante : <= 0 pour ln, log et sqrt ; = 0 pour / et inv, la calculatrice déclenchera plusieurs LogMessage destinés a la console, à un fichier et à l'utilisateur. Les constantes dépilées seront rempilées en attente d'une meilleure utilisation.

Gestion de la surcharge des opérateurs

Les opérateurs sont de manière générale surchargés avec toutes les possibilités, ceci afin de permettre une plus grande malléabilité. Ainsi si l'on veut par la suite autoriser l'addition de réel avec des complexes, cela reste possible. Lorsque le résultat retourné est invalide soit la fonction renvoie une Constante égale à 0, soit elle renvoie un pointeur de Constante NULL. Dans tous les cas, un LogMessage est généré (sur le même principe que précédemment).

Implémentation de annuler, rétablir

L'historique permet de gérer annuler et rétablir, c'est une pile de pointeurs de pile. Elle garde en mémoire chaque instance de la pile pour pouvoir rétablir. On utilise deux piles de pointeurs de pile, une pour annuler, et une autre pour rétablir. Lorsque l'on push une pile sur l'un des pointeurs de pile on la pop sur l'autre.

Gestion de la mémoire

Afin de pouvoir implémenter l'historique, il nous faut conserver les Constantes générées tout au long du programme. On supprime donc les constantes à la fermeture de la calculatrice. Les seuls delete sont effectués à la destruction de la pile.

Design Pattern Factory

Afin de pouvoir créer des constantes et des nombres facilement (rappelons que ces classes sont abstraites), on utilise le design pattern factory. Celui-ci nous permettra d'adopter différentes stratégies de création : normale, on utilise la chaine de caractère envoyée ; selon un type ; créé un complexe, recopier une constante ou un nombre.

Design Pattern Modèle Vue Contrôleur

Dans notre projet, nous avons surtout modélisé le modèle (CalculatriceModele) et la vue (MainWindow). Les actions du contrôleur sont effectuées par chacune des deux entités. En effet, chacune contrôle elle-même ce qu'elle envoie à l'autre.

Design Pattern Itérateur

Afin de faciliter la gestion de la pile (que nous avons décidé de coder nous-mêmes pour y placer swap et clear), nous avons décidé de doter celle-ci d'un itérateur. Le début de l'itérateur est en haut de la pile, et sa fin, en bas.

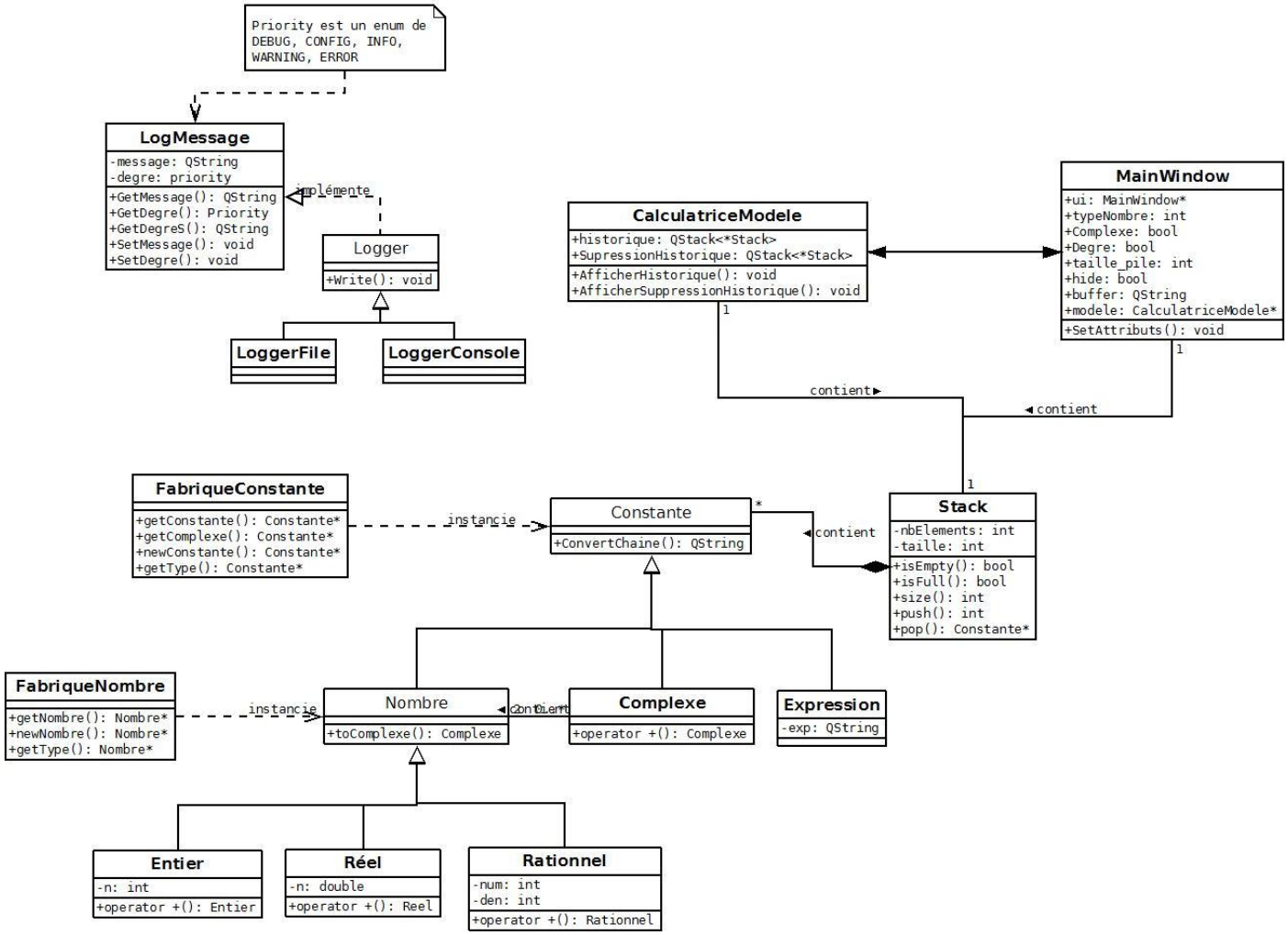
Sauvegarde de contexte

La sauvegarde de contexte s'effectue dès que l'on appuie sur enregistrer. Si à l'ouverture du programme on souhaite la récupérer, il suffit de cliquer sur ouvrir.

UML

Voici l'UML général de notre projet, les méthodes ne sont pas toutes indiquées, pour ne pas nuire à la lisibilité. Les constructeurs et destructeurs ne sont pas indiqués mais existent pour toutes les classes.

Les loggers sont accessibles depuis toutes les classes.



Diagrammes de séquence

Voir pdf annexe (diagramme de séquence).

Accès au système de dépôt sur GitHub : https://github.com/lilireli/LO21_LilireliAfsanee