Rapport de PROJET

**Calculatrice à notation polonaise inverse**

LO21

## Présentation du projet

Le projet de lo21 consiste en la réalisation d’une calculatrice graphique à notation polonaise inverse, la notation polonaise inverse étant une méthode de notation mathématique permettant de se passer de l'utilisation de parenthèses.

Exemples :

* 1+1 s'écrit 1 1 +
* 2 x 2 + 1 s'écrit 2 2 x 1 +
* (2 +3) x 4 s'écrit 2 3 + 4 x

## Choix techniques

Pour mener à bien ce projet, nous avons utilisé le paradigme de programmation informatique de la POO (programmation orienté objet) à l’aide du langage C++.

Nous avons également utilisé le framework Qt qui est employé pour concevoir des interfaces graphiques.

Pour la sauvegarde du contexte d’une pile, nous avons utilisé l’XML, via l’objet DOM de Qt.

## Choix Conceptuel

Nous avons choisi de ne pas implémenter les modes d’exécutions de la calculatrice. Ceci implique de pouvoir travailler avec tous les types en même temps, et a donc engendré une petite complexité supplémentaire pour l’implémentation des opérateurs binaires (utilisation de dynamic\_cast pour reconnaitre le type, levée d’exception en cas d’erreur). Afin de gérer les erreurs, nous avons créé notre classe d’exception « Type\_exception » qui dérive de std ::Exception.

Pour les divisions, nous avons privilégié la création de fraction en résultat. Afin de pouvoir obtenir une valeur approchée, nous avons défini la méthode eval, qui renvois un reel à partir d’une fraction.

## Design pattern utilisés

Dans la réalisation de notre projet, nous avons utilisé les design pattern suivant :

* Singleton
* Factory
* Memento

De plus, l’utilisation de Qt, nous a également fait manipuler, bien que de manière indirecte, les design pattern observer et iterator.

Détaillons chacun de ces design pattern :

**Singleton :** le singleton est un design pattern créationnel dont l'objet est de restreindre l'instanciation d'une classe à un seul objet (ou bien à quelques objets seulement). Ce design pattern est utilisé dans le projet pour la classe collection pile, qui stocke toute les piles en activité, ainsi que pour la classe « type\_factory ».

**Factory :** Factory est un [patron de conception](http://fr.wikipedia.org/wiki/Patron_de_conception) créationnel qui permet d'instancier des objets dont le type est dérivé d'un type abstrait. La classe exacte de l'objet n'est donc pas connue par l'appelant.

On l’emploie ici afin d’instancier une des classes dérivées de Type (entier, reel, complexe, etc.) à partir d’une chaîne de caractères.

**Memento :** Le patron mémento est un [patron de conception](http://fr.wikipedia.org/wiki/Patron_de_conception) comportemental qui permet de restaurer un état précédent d'un objet (retour arrière) sans violer le principe d'encapsulation.

Le mémento est utilisé par deux objets : le créateur et le gardien.

Le créateur est un objet ayant un état interne (état à sauvegarder).

Le gardien agira sur le créateur de manière à conserver la possibilité de revenir en arrière. Pour cela, le gardien demande au créateur, lors de chaque action, un objet mémento qui sauvegarde l'état de l'objet créateur avant la modification.

Dans notre projet le design pattern Memento sert à implémenter la fonction undo/redo en sauvegardant le contexte de la pile à chaque nouvelle opération.

# UML

# 

