Réalisé par :

KANOBLAUCH Marc

ABDOULAYE Ablo

SAKHI Mehdi

LAHLOU Mohammed

Projet qualité de programmation

Calculatrice HP



Encadré par :

Mr. M. RIVIERE Stephane

# Sommaire

[Sommaire 2](#_Toc471655041)

[Introduction 3](#_Toc471655042)

[1. Présentation générale 4](#_Toc471655043)

[1/ Présentation : 4](#_Toc471655044)

[2/ Modélisation : 5](#_Toc471655045)

[3/ traitement des données : 6](#_Toc471655046)

# Introduction

Notre projet s’inscrit dans le cadre de notre 3 ème année licence MIAGE/INFO à la Faculté des Sciences et Techniques de Mulhouse. Le but étant de mettre en pratique les connaissances acquises durant la formation et spécifiquement en cours de qualité de programmation. Des connaissances telles que la programmation orientée objet, GIT et surtout programmer un code lisible et robuste.

Trois projets étaient disponibles:

* Gestion d’emploi du temps.
* Réalisation d’une calculatrice HP programmable.
* Logiciel de gestion de formes géométriques vectorielles.

Ce projet a été réalisé en groupe. Notre groupe se compose de 4 personnes : KNOBLAUCH Marc, DIALLO Abdoulaye, LAHLOU Mohammed et SAKHI Mehdi.

Disposant de toutes les compétences nécessaires afin de mener à bien ce projet et suite à une réunion avec les membres de l’équipe, nous nous sommes mis d’accord sur le projet suivant :  **réalisation d’une calculatrice HP programmable.**

Le rapport détaillera trois grands chapitres, dans une première partie nous allons présenter notre projet, dans une seconde partie nous allons citer les outils et les technologies utilisées pour réaliser ce projet, finalement avant de conclure la troisième partie sera consacrée aux difficultés et problèmes que nous avons rencontrés durant la programmation.

# Présentation générale

## 1/ Présentation :

Calculatrice à notation polonaise inversée. Ce type de calculatrice a pour principe la notation post-fixée. Nous avons pour habitude d’utiliser des calculatrices à notation infixée.

* Version infixée : 2 + 3 égale à 5.
* Version post-fixée : 2 3 + toujours égale à 5.

La calculatrice gère plusieurs types de constantes :

* Des nombres
* Des expressions

Une expression est une chaîne de caractères encapsulant des nombres et/ou des opérateurs. Quant aux nombres, ils peuvent être soit des nombres entiers, soit des nombres réels ou des rationnels (un rationnel étant un réel).

La calculatrice gère une pile (de constantes) qui affiche les constantes empilées et les résultats de calculs.

Elle gère aussi la sauvegarde des données, c'est à dire qu'à sa fermeture, l'état de la pile ainsi que ses paramètres sont sauvegardés, afin qu'à sa réouverture on retrouve l'environnement de travail précédent.

Il est possible d’effectuer certaines fonctions sur la calculatrice. Les fonctions « vider », « supprimer dernier » ont respectivement pour but de supprimer toutes les valeurs de la pile et de ne supprimer que la dernière valeur de la pile. Ainsi l'utilisateur peut retrouver à n’importe quel moment l'état de pile avant le dernier empilement.

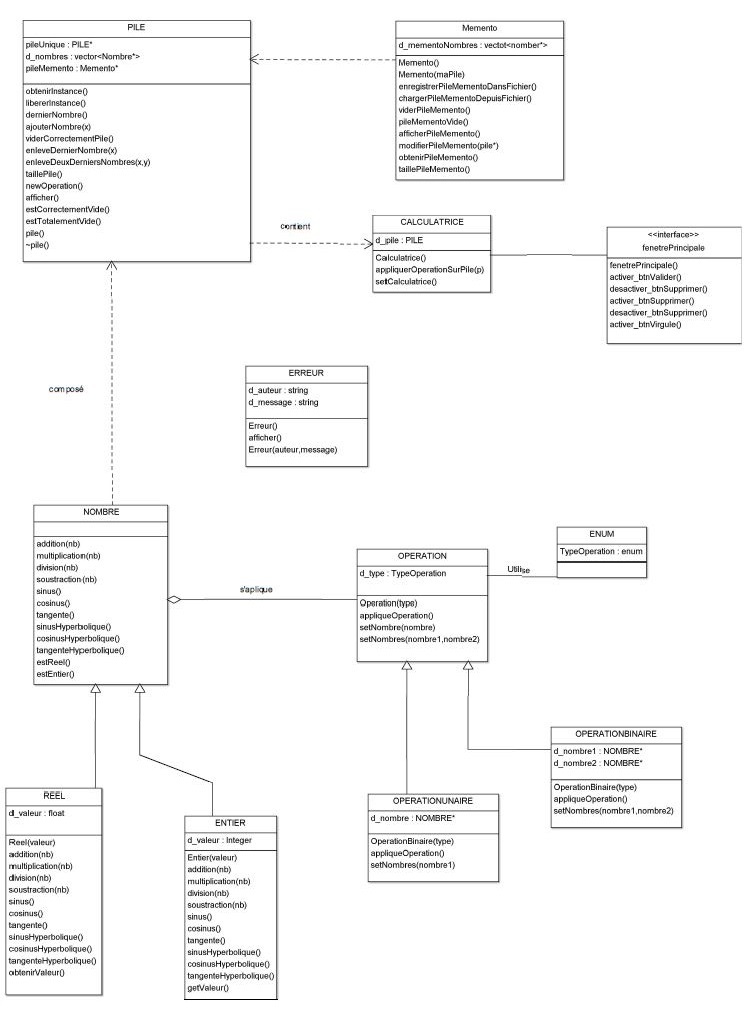
Pour finir, la calculatrice permet de faire des opérations de types unaires ou binaires. On peut ainsi appliquer une opération sur les valeurs stockées dans la pile de la calculatrice. Elle gère également son affichage, elle affiche les valeurs empilées dans la pile, et notamment après l’exécution d’une opération sur cette dernière.

2/ Modélisation :

Afin de mener à bien notre projet, il était important de se fixer une ligne directrice, qui a été éventuellement retravaillée durant le projet.

Pour ce faire, une phase de formalisation précédente l’étape du développement à proprement parler était essentielle. C’est grâce au langage UML que nous avons pu clarifier nos idées.

Ci-dessous, le diagramme UML final relatif à notre projet.



## 3/ traitement des données :

**La classe calculatrice**

Nous avons utilisé le design pattern “singleton” pour coder cette classe. Ce dernier permet l’instanciation unique de l’objet. Celle-ci nous permet de faire l’interface entre les opérations et sa pile via la fonction appliquerOperationSurPile.

**La classe pile**

Nous avons aussi utilisé le design pattern “singleton” pour coder cette classe.

La  classe pile contient notamment un vector de Nombre qui  stocke les nombres utilisés par la calculatrice. Elle contient aussi une pileMemento qui est un pointeur sur la classe Memento, qui permet d’enregistrer notre pile dans un fichier texte, pour ensuite pouvoir la récupérer grâce aux méthodes sauvegarderPileDansMemento() et restaurerMementoDansPile(). De plus, deux pointeurs sur Nombre, à savoir x et y, permettent de récupérer les deux derniers nombres retirés de la pile. Ce sont sur ces nombres que les opérations sont effectués, en ce sens sur x pour une opération unaire, sur x et y pour une opération binaire.

**La classe Memento**

Nous avons utilisé le design pattern “memento” pour réaliser l’objet mémoire de la pile. A tout moment, l’utilisateur peut enregistrer sa pile dans le memento. Ce dernier peut utiliser ses fonctions enregistrerPileMementoDansFichier() et chargerPileMementoDepuisFichier().

**La classe nombre**

La classe nombre est une classe abstraite. Elle englobe les différents types de nombres (Entier, Réel, Rationnel, ...) et contient aussi la déclaration des opérations unaires et binaires qui seront définies dans les classes dérivées. Nous avons pour le moment uniquement développé la classe Réel, un réel est un nombre.

Cette modélisation permet donc d’ajouter d’autres types de nombres tel que des nombres de type entier, rationnel, complexes...

**La classe operation**

Elle aussi abstraite, elle utilise un type énuméré de toutes les opérations : nous définissons toutes les opérations faisables. Les opérations unaires s’appliquent sur un nombre (sinus, cosinus, tangente,...) et celles binaires s’effectuent sur deux nombres (addition, soustraction, multiplication, division, …). Les classes opérations unaires et binaires sont donc des opérations. Elles héritent de la classe operation.

# 

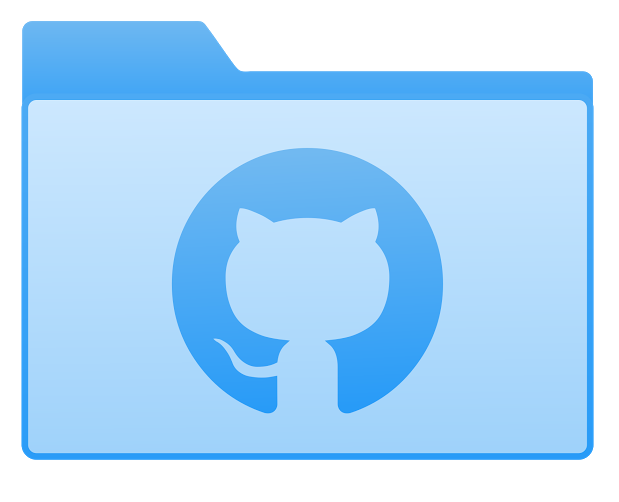
# Outils et technologies utilisés

Pour mener à bien le projet, nous nous sommes mis d’accord sur l’utilisation des technologies suivantes:



**Dev-C++**

Dev-C++ est un [environnement de développement intégré](https://fr.wikipedia.org/wiki/Environnement_de_d%C3%A9veloppement_int%C3%A9gr%C3%A9) (IDE) permettant de programmer en [C](https://fr.wikipedia.org/wiki/C_%28langage%29) et en [C++](https://fr.wikipedia.org/wiki/C_plus_plus). C’est cet environnement que nous avons utilisé durant notre projet pour le développement de nos différentes classes.



**Git :**

Git est un [logiciel de gestion de versions](https://fr.wikipedia.org/wiki/Logiciel_de_gestion_de_versions) [décentralisé](https://fr.wikipedia.org/wiki/Gestion_de_versions#Gestion_de_versions_d.C3.A9centralis.C3.A9e). On l’a utilisé pour gérer l'historique du projet et pour que chacun puisse programmer sa partie sur son portable.



**ArgoUML:**

ArgoUML est un [logiciel libre](https://fr.wikipedia.org/wiki/Logiciel_libre) de création de diagrammes [UML](https://fr.wikipedia.org/wiki/Unified_Modeling_Language), ce logiciel nous a servi pour la modélisation UML.



**QT:**

QT est une [API](https://fr.wikipedia.org/wiki/Interface_de_programmation) [orientée objet](https://fr.wikipedia.org/wiki/Programmation_orient%C3%A9e_objet) et développée en [C++](https://fr.wikipedia.org/wiki/C%2B%2B) par [Qt Development Frameworks](https://fr.wikipedia.org/wiki/Qt_Development_Frameworks), filiale de [Digia](https://fr.wikipedia.org/w/index.php?title=Digia&action=edit&redlink=1). Qt offre des composants d'[interface graphique](https://fr.wikipedia.org/wiki/Interface_graphique) ([widgets](https://fr.wikipedia.org/wiki/Composant_d%27interface_graphique)),on l’utilise pour l’interface graphique avec un afficheur.



**Doxygen**:

Doxygen est un [générateur de documentation](https://fr.wikipedia.org/wiki/G%C3%A9n%C3%A9rateur_de_documentation) sous [licence libre](https://fr.wikipedia.org/wiki/Licence_libre) capable de produire une [documentation logicielle](https://fr.wikipedia.org/wiki/Documentation_logicielle) à partir du [code source](https://fr.wikipedia.org/wiki/Code_source) d'un [programme](https://fr.wikipedia.org/wiki/Programme_informatique). On l’utilise pour la documentation du programme.

# **Problèmes rencontrés et solutions apportées:**

    Notre application étant fonctionnelle, elle a été construite pour permettre des améliorations et extensions possibles. Néanmoins il reste des imperfections et une certaine amélioration à faire.

Premièrement, la conception de l’UML n’a pas été évidente. Autant le but du projet était clairement identifié, autant convenir d’une conception était une tout autre épreuve. Nous nous sommes forcés à élaborer un UML de départ qui a fortement évolué depuis le début du projet. En effet au fur et à mesure nous nous sommes rendu compte que certaines entités étaient encore inconvenables et qu’il fallait les modifier.

Deuxièmement, nommer les fonctions et méthodes selon leurs utilités. Car ces dernières évoluaient elles aussi au fur et à mesure du projet. Il fallait alors que le code ne perd pas son sens. Ce ne fut donc pas évident d’omettre les commentaires au profit de noms clairs.

Troisièmement, la duplication du code a été pour la plus grande partie du programme éradiquée. Certaines d’entre elles n’ont pourtant pas encore été remaniées, faute de solution. Une amélioration possible serait de préférable.

Quatrièmement, concernant la classe abstraite Nombre. Nous avons décidé de l’implémenter de cette manière, dans le but de faire évoluer le programme de la vers d’autres types de nombre.

Finalement, le temps occasionné par les remaniements a été conséquent. Le code peut très certainement être encore amélioré, mais celui-ci est surtout lisible, ce qui permettra à quiconque de facilement le prendre en main pour l’améliorer.

# **Conclusion:**

En somme, le projet a été bénéfique dans le sens où nous avons pu mettre en pratique les notions abordées en cours et aussi il nous a permis de nous familiariser avec le gestionnaires de version Git, et surtout de faire des tests unitaires avec CATCH.

Le fait d’apporter des solutions pour éviter la duplication de certaines parties de code, nous a permis d’avoir un code lisible et facile à comprendre.

D’un autre point de vue, le travail en groupe est une bonne chose. Car la réunion de connaissance produit toujours une connaissance tierce. Nous avons donc confronter nos idées pour bien cerner le projet et le découpage des tâches nous a permis d’avancer plus vite.