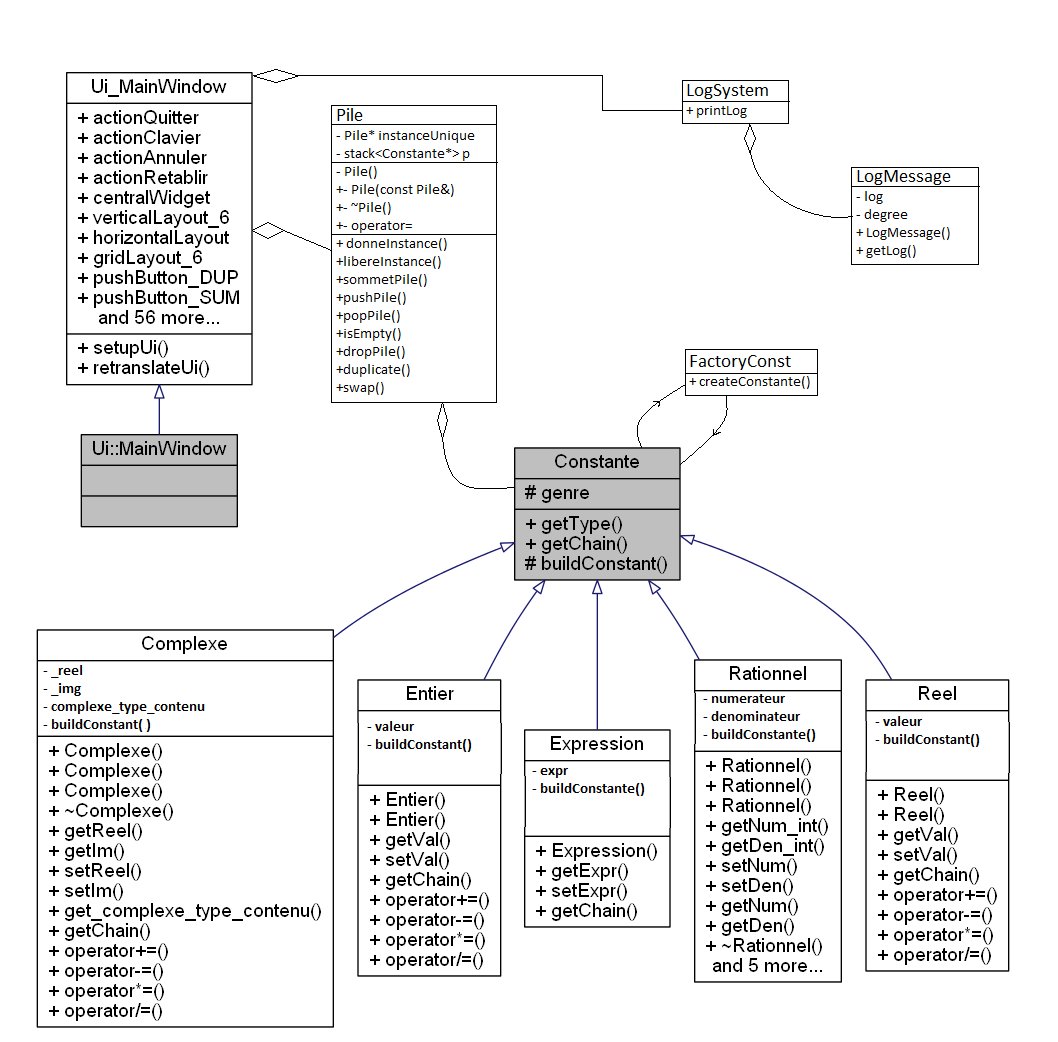
Projet LO21 – Calculatrice à cotation polonaise inversée

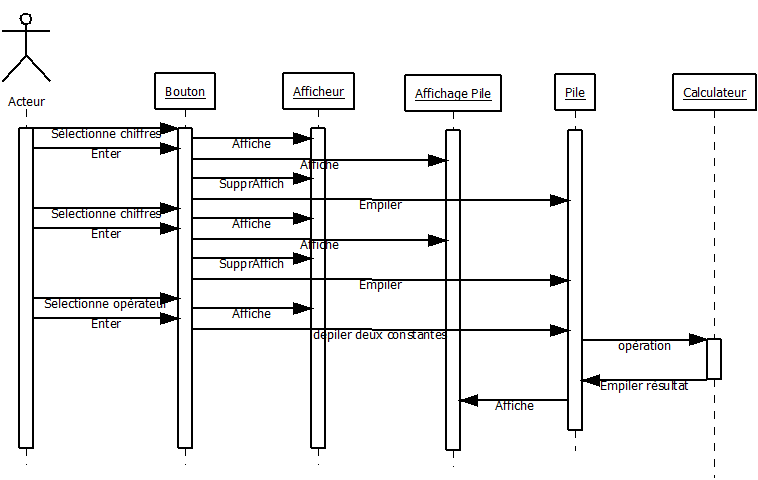
# Introduction

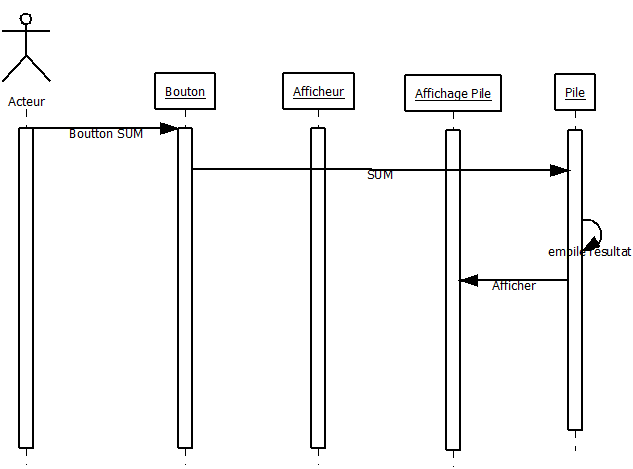
Dans le cadre de notre projet, nous avons réalisé une calculatrice à notation polonaise inversée respectant les principales contraintes du sujet. Nous avons tout d’abord défini l’UML modélisant le projet selon notre point de vue. Puis nous avons réalisé les principaux diagrammes de séquences correspondant à cet UML afin que nous puissions commencer à implémenter le code de notre calculatrice.

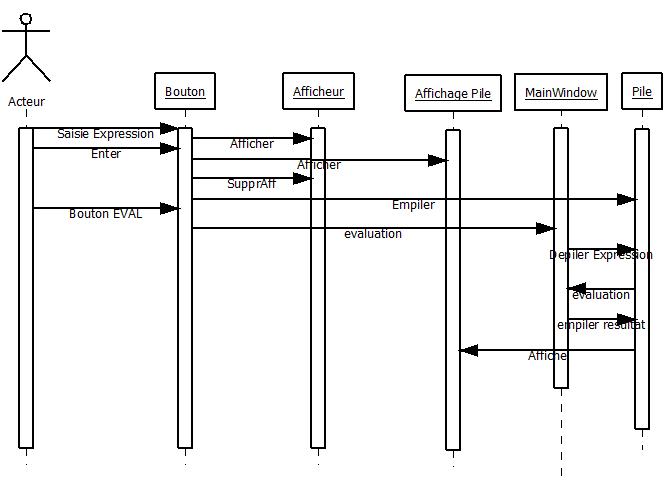
# UML



# Diagrammes de séquences

L’utilisateur effectue une opération nécessitant le dépilement de deux constantes, comme par exemple une addition, soustraction, pow, mod, etc.

L’utilisateur effectue la somme d’un certain nombre de constantes de la pile

L’utilisateur entre une expression puis l’évalue

# Explication des choix de conception

Nous avons créé plusieurs objets afin de respecter les principes généraux de la programmation orientée objet. Ici le principe d’encapsulation est donc bien respecté. La classe MainWindow correspond à notre interface graphique et donc à ce que nous pourrions le plus assimiler à l’objet calculatrice. En effet, au sein de la classe MainWindow nous avons implémenté toutes les opérations disponibles sur la calculatrice. Cette calculatrice est composée d’une pile qui elle est composée de constante. Nous avons ici créé une classe abstraite regroupant tous les types de constantes que nous utilisons. Ainsi la classe « Constante » admet quatre filles qui sont : « Reel », « Rationnel », « Entier », « Complexe » et « Expression ». Dans cet héritage nous avons utilisé le design pattern Template method afin de factoriser le code. Par exemple, pour des fonctions telles que la fonction membre privée et surtout virtuelle pure « buildConstant » qui construit une constante à partir de n’importe quel type afin de permettre à ces constantes d’être stocké dans la pile lors d’une futur utilisation. Ce design pattern est aussi utilisé pour les fonctions membres « getChain » qui est un accesseur retournant la chaine correspondante au type utilisé. La pile est ici manipulée via le design pattern singleton qui permet une instanciation unique de la classe. Ceci est plus sûr que d’utiliser une variable globale.

Concernant la fonction d’évaluation des expressions, suite à un problème quant à l’implémentation du raccourcis clavier pour la simple quote, nous avons implémenté ce raccourci avec « @ ».

Malgré des débuts difficiles, nous avons su nous adapter à ce nouveau langage de programmation qu’est le C++ et aux concepts orientés objet qui en découle. Nous avons pu nous familiariser d’avantage avec la notion de projet informatique en partant de la conception jusqu’à la réalisation. Ce projet nous a permis de comprendre l’importance de la répartition du travail et les notions de travail en binôme sur un projet nouveau.

Ce projet fût une réelle application dont nous pouvions voir les résultats après avoir travaillé. L’adaptation au framework Qt nous a permis de nous familiariser avec les librairies graphiques que nous ne connaissions pas.

En somme, ce projet fût très intéressant et nous a permis d’acquérir de nouveaux automatismes en C++ mais aussi en management de projet informatique