Rapport POO Labo 07

# Modélisation UML mise à jour

Une image contenant texte, diagramme, Parallèle, Plan

Description générée automatiquement

# Choix de conception

Notre classe *State* est composée de deux attributs majeurs étant *stack* et *currentValue* et différents « flags » permettant de lire de la manière la plus correcte possible les valeurs.

Les flags *hasDecimalPoint* et *decimalPlaces* permettent, comme leurs noms l’indique, de déterminer l’état décimal de la valeur courante afin d’entrer correctement de nouveaux nombres.

Le flag *isIntermediateResult* permet de savoir si la valeur courante est le résultat intermédiaire d’une opération afin de pousser la valeur dans la pile si l’utilisateur entre un nouveau nombre au lieu de l’écraser.

Pour la superclasse *Operateur* nous avons simplement ajouté en paramètre de la fonction *execute* un *State* permettant ainsi à nos différents opérateurs de récupérer les différentes données de l’état actuel défini précedemment.

Pour chaque operateur nous avons créé une nouvelle classe correspondante héritant de la superclasse *Operateur* a l’exception des opérateurs *MemoryStore*, *MemoryRecall* et *Backspace* qui sont déclaré directement dans la classe *JCalculator* afin d’accéder aux données de cette dernière.

Afin que le programme fonctionne aussi bien avec l’interface graphique qu’en CLI, nous avons décidé, pour chaque opérateur, de pousser la valeur courante dans la pile si celle-ci est non-nulle pour ensuite récupérer toutes les opérandes nécessaires à l’opération depuis la pile.

Le CLI ne poussera donc rien et récupérera les valeurs dans la pile directement car il ne possède pas de valeur courante et l’interface graphique va pousser la valeur courante effectuant la manipulation expliquée précédemment.

# Tests et résultats

|  |  |  |  |
| --- | --- | --- | --- |
| **Numéro** | **Description du test** | **Résultat** | **Commentaires** |
| 1 | Exemple d’utilisation avec l’expression (3.5 + 4) / (2.5^2 + 1) | OK | L’expression est correctement interprétée est le résultat est correct. |
| 2 | Exemple d’éxecution de l’extension CLI avec l’expression (sqrt(3 + 2) + 1) | OK | L’expression est correctement interprétée est le résultat est correct. |
| 3 | Test unitaire de chaque opération en mode graphique | OK | Toutes les opérations sont fonctionnelles. |
| 4 | Test unitaire de chaque opération sur l’extension CLI | KO | Nous n’avons pas implémenté les opérateurs de mémoire et de clear dans l’extension CLI. |
| 5 | Entrée d’une opération inexistante sur l’extension CLI | OK | Un message d’erreur « Opération inconnue » est renvoyé propose une nouvelle entrée. |

## Output des tests :

### Test numéro 1 :

Une image contenant texte, capture d’écran, nombre, Police

Description générée automatiquement

### Test numéro 2 :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

### Test numéro 5 :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

# Listing du code

Le listing du code se situe à la fin de ce document.

# Annexes

Le code source de ce projet est disponible dans le répertoire « Code source ».