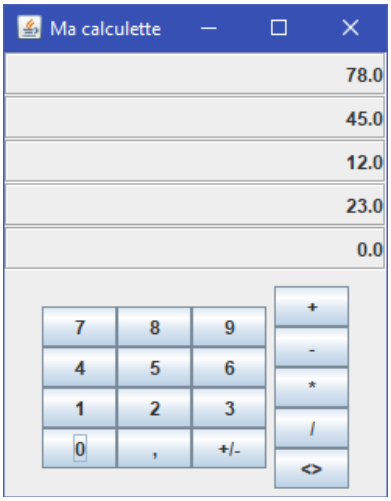
TD POO - Java (première partie)

IMT Mines Ales – M1 – Pré-département 2IA
Octobre 2021

Christelle Urtado
Sylvain Vauttier

Sujet :

L'objet de ce TD est la mise en œuvre des bonnes pratiques de programmation vues en cours. Vous devrez donc être particulièrement attentifs à la séparation modèle / interface (la mise en œuvre du patron MVC est expliquée dans la suite du sujet) ainsi qu'à la bonne utilisation des concepts objet (héritage, destruction, ...). Le cas d'étude proposé est la réalisation d'un accessoire de bureau de type calculette. Elle sera réalisée en deux étapes au fil des TDs.



Cahier des charges simplifié :

- Implémenter dans un premier temps une calculatrice permettant d'effectuer les opérations arithmétiques de base (addition, soustraction, multiplication, division) sur des nombres décimaux**.**

Cette calculatrice doit fonctionner en « notation polonaise inverse » : les différents opérandes sont empilés sur une pile de données (opération « push »), puis les différents opérateurs appliqués. Par exemple, pour réaliser l'addition de deux nombres :

- un premier nombre est saisi puis mis sur la pile (touche push),
- un deuxième nombre est saisi puis mis sur la pile (touche push),
- l'opération d'addition est appliquée (touche +).

Cette logique de fonctionnement est très pratique car elle permet de gérer des expressions complexes (par exemple 3*(2+5*(4+7)) sans le recours à des « mémoires » ou aux parenthèses.

- Une fois cette première calculatrice implémentée, une première extension pourra être de lui ajouter un traceur de fonctions. Une autre extension pourra être de lui ajouter une fonctionnalité de calcul formel sur des polynômes.

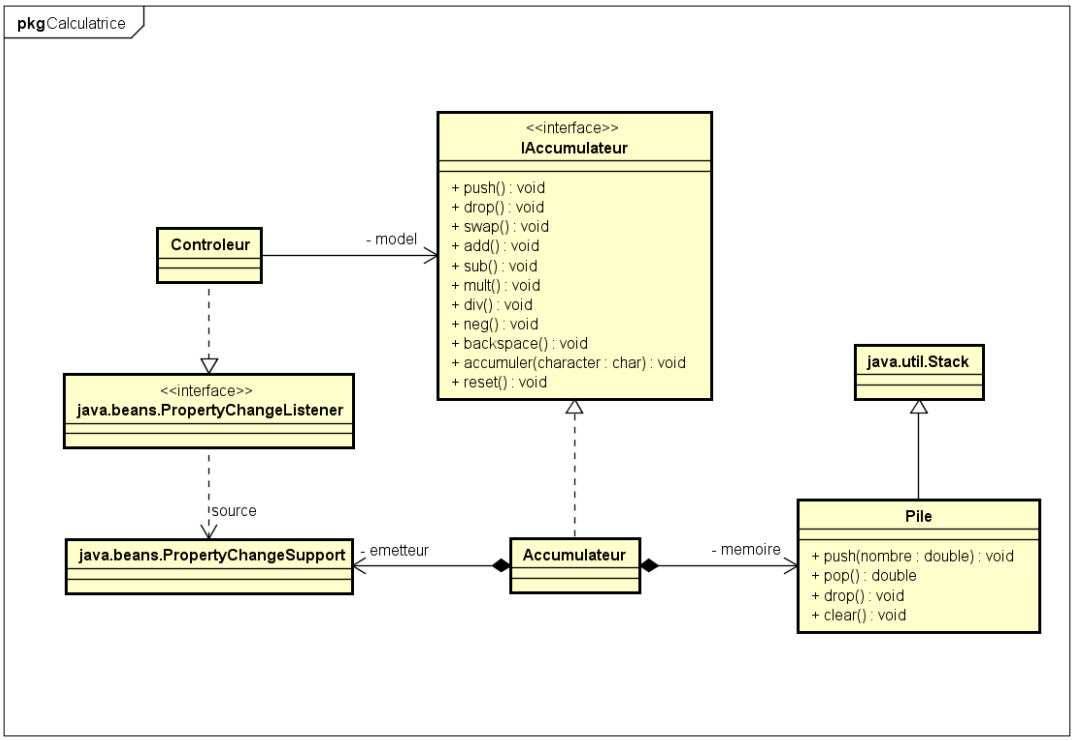
Spécifications simplifiées de la calculatrice

L'implémentation de la calculatrice doit respecter le patron de conception MVC (« Modèle », « Vue », « Contrôleur »). Ce patron spécifie qu'une application est composée de trois types d'éléments :

- des modèles, qui gèrent la représentation des concepts du systèmes (données et opérations de manipulation de ces données)
- les vues, qui gèrent les interfaces graphiques permettant les interactions avec l'utilisateur,
- les contrôleurs, qui gèrent les collaborations entre les vues et les modèles. Ainsi, les contrôleurs avertissent les vues des changements de valeurs d'un modèle afin qu'elles mettent à jour leur affichage. Inversement, les contrôleurs demandent aux modèles des changements de valeur pour refléter les actions des utilisateur (par exemple la saisie d'une valeur).

L'intérêt de ce patron de conception est de découpler (rendre indépendants) les vues et les modèles et de renforcer ainsi la modularité de l'application. Il est alors possible de changer les vues (modifier une interface, ajouter une interface) sans changer les modèles et inversement. L'impact de tels changements est alors limité au « connecteur » (le contrôleur) qui assure le lien entre la vue et le modèle.

Nous allons dans un premier temps implémenter le « modèle » de la calculatrice, indépendamment d'une future interface. Le diagramme suivant offre une vue conceptuelle des principaux éléments du modèle :



- l'accumulateur : représente une mémoire qui contient le résultat du calcul en cours. Il est possible d'empiler le contenu de l'accumulateur sur la pile (Push), de vider le contenu de l'accumulateur (Clear). Les différentes opérations sont demandées à l'accumulateur. En fonction de leur arité, l'accumulateur dépile le nombre d'opérandes nécessaires au

calcul. Si l'accumulateur est plein, le contenu de l'accumulateur est utilisé comme opérande.
- la pile : permet de mémoriser les opérandes nécessaires aux calculs (push). Il est possible d'éliminer le dernier opérande placé sur la pile (drop), d'échanger l'ordre des deux derniers opérandes (swap) et de vide la pile (clear). La classe Pile est implémentée à l'aide de la classe java.util.Stack.

Pour faciliter l'implémentation d'une structure MVC, Java propose l'infrastructure JavaBeans. Ainsi, l'accumulateur et la pile utiliseront une instance de la classe PropertyChangeSupport pour avertir le contrôleur des changements de valeurs utiles. Cette classe permet à un objet de venir s'enregistrer en tant qu'écouteur d'événements puis d'envoyer à ces écouteurs des événements Le contrôleur implémentera l'interface java.beans.PropertyChangeListener pour être capable de recevoir ces événements (vous reporter à la documentation de Java pour le détail de ces classes).

Dans un premier temps, le contrôleur implémentera un programme principal (main) permettant de tester le modèle de la calculatrice et se contentera d'afficher dans la console le nouvel état du modèle en réponse aux événements.