Introduction

Qu'est ce une calculatrice en notation polonaise inversée ?

Inventée en 1924 par le mathématicien polonais Jan Łukasiewicz. La calculatrice en notation polonaise inversée permet d'écrire sans ambiguïté les formules arithmétiques sans parenthèses. Elle diffère d'une calculatrice conventionnelle, à savoir par l'ordre des termes où les opérandes sont présentés devant les opérateurs.

Objectif : Créer une calculatrice sous Java en introduisant le concept d'orienté objet, d'architecture MVC et des fonctionnalités qui rendent la calculatrice plus agréable à utiliser.

Développement

Nous avons tout d'abord commencé à construire le modèle de la calculatrice (classes Pile et Accumulateur), nous avons ensuite construit le contrôleur et enfin le view(classes GUI et IView).

1. Classe Pile

La classe **Pile** hérite de la classe **Stack<Double>** comportant des éléments de type double.

L'un des principaux avantages de l'héritage est que celui-ci permet de réutiliser le code d'une classe existante. Nous construisons la classe Pile à partir de la classe existante Stack. La classe Pile va ainsi hériter des variables et des méthodes de Stack.

2. Classe Accumulateur

L'accumulateur représente une mémoire qui contient le résultat du calcul en cours de la calculatrice.

C'est dans cette classe que les méthodes des différentes opérations y sont regroupées : addition, soustraction, multiplication et division.

Nous ajoutons à cela, les méthodes « swap », « négation », « Clear », « push » et « arrondir ».

La méthode arrondir permet d'avoir un affichage d'un nombre à quelques chiffres après la virgule. Dans notre projet, nous avons fait le choix d'afficher cinq chiffres après la virgule.

Afin de pouvoir avertir le contrôleur de tout changement de la pile, il est nécessaire de créer un support (PropertyChangeSupport) afin que ce dernier puisse envoyer des signaux au contrôleur avec la méthode FirePropertyChangeSupport. Les méthodes d'opérations et « Clear » possèdent.

3. Classe Contrôleur

La classe contrôleur doit assurer la liaison des informations qui circulent entre la classe accumulateur (mémoire de la pile) et la classe GUI l'interface graphique qui affiche le contenu de la pile. 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).

Le contrôleur doit ainsi implémenter les interfaces PropertyChangeListener et EventHandler<MouseEvent> :

- L'interface PropertyChangeListener permet au contrôleur de récupérer les signaux envoyés par l'accumulateur et d'avertir les classes accumulateur et GUI.
- L'interface EventHandler<MouseEvent> permet de récupérer toutes les informations liées à l'interface MouseEvent qui représente les événements liés à l'interaction de l'utilisateur avec une souris. Il récupérera donc les informations sur le type de bouton cliqué avec la souris avec MOUSE_CLICKED.

Alternativement, nous avons pu utiliser l'ActionEvent que les boutons peuvent utiliser pour déterminer directement l'action à exécuter lorsque le bouton est actionné. Nous avons choisi de conserver le MouseEvent pour gérer les principales fonctionnalités de la calculatrice plutôt que le ActionEvent car nous l'avions utilisé bien avant l'arrivée du TD2.

Toutefois, nous avons utilisé ActionEvent pour créer d'autres fonctionnalités de la calculatrice (bouton information par ex).

Une particularité de notre calculatrice est qu'elle dispose de commentaires selon les actions qui ont été effectuées sur la calculatrice. Par exemple, appuyer sur le bouton Effacer indiquera que la mémoire de la calculatrice a été supprimée. Nous avons intégré dans la calculatrice l'affichage de l'historique et l'état de la pile qui permettra à l'utilisateur de savoir ce qu'il a saisi dans la pile. Le rôle du contrôleur sera, ici, d'adapter les commentaires et les affichages en fonction des actions effectuées par l'utilisateur.

Le contrôleur doit également assurer le bon fonctionnement de la calculatrice et qu'aucune erreur ne se produise par exemple avec la division par 0 où l'utilisateur sera prévenu de cette erreur grâce au commentaire qui sera affiché. Des méthodes comme « push » ont été créées pour isoler ces erreurs. Notez que nous ne réécrivons pas les principales fonctionnalités de ces méthodes, nous isolons seulement les erreurs qui peuvent se produire.

Autres fonctionnalités diverses qui ont été ajoutées à la calculatrice :

- Mettre une virgule
- Pourcentage qui permet de multiplier par 0.01 le nombre que l'on a saisi.
- Supprimer les chiffres 1 à 1.
- Changer le signe du nombre saisi sans l'avoir stocké dans la pile.

4. Classe GUI (Graphical User Interface)

La classe GUI gère l'interface graphique permettant les interactions avec l'utilisateur. Nous avons rassemblé toute la partie graphique dans cette classe. C'est dans cette classe que sont créés la fenêtre Stage, le conteneur d'éléments StackPane et les différents objets (Buttons, Labels, Images etc…).

Pour rendre la lecture du code plus agréable, nous avons créé les méthodes :

- « initialisation » pour la création des boutons Fermer et Minimiser, du titre et du logo.
- « createButton » pour la création des boutons avec les différentes personnalisations (styles, couleurs, tailles, polices etc…).
- « createLabel » pour la création de l'affichage des résultats, de la pile, des commentaires et de l'historique.

Pour relier les boutons, qui gèrent les fonctionnalités principales de la calculatrice, de la classe GUI et de la classe contrôleur, nous avons appliqué la méthode addEventHandler(MouseEvent, controleur) aux boutons. De même pour l'intégration du clavier, nous avons appliqué la méthode addEventFilter(KeyEvent.KEY_PRESSED, input) à la scène.

La taille maximale qu'on peut saisir sur la calculatrice est de 9 chiffres. Si on atteint la taille maximale le contrôleur enverra un signal à la classe GUI pour avertir l'utilisateur que la taille maximale a été atteinte.

Des méthodes comme updateAffichageResultat(), updateAffichageMessage() ou updateHistorique() ont été créés afin d'aérer le code.

Autres fonctionnalités diverses qui ont été ajoutées à la calculatrice :

- Les boutons sont « réactifs », ils s'illuminent d'une couleur différente lorsque l'on appuie dessus avec la souris ou une touche du clavier. La fonctionnalité est intégrée avec les méthodes updateButtonOnClick et updateButtonOnKey.
  NB : Elle ne fonctionne pas à 100% sur le clavier.

5. Classe Input

La classe Input traite les évènements indiquant qu'une frappe a eu lieu dans un composant (ici les boutons). Nous pouvons ainsi affecter à chaque touche du clavier les actions que nous voulons exécuter.

➔ Lire le guide d'utilisation des différents raccourcis sur le ReadMe ou sur GitHub.

6. Interface IView

L'interface IView possède les méthodes affiche(), change(List<String>) et change(String).

7. Easter Egg

Vous pouvez vous apercevoir des différentes fonctionnalités diverses qui ont été intégrées sur la calculatrice :

- Calculatrice inspirée du thème de la calculatrice de l'iPhone.
- La fenêtre est déplaçable sur n'importe quel endroit de l'écran.
- Le bouton « _ » pour minimiser la fenêtre.
- Le bouton « i » pour afficher les crédits des contributeurs du projet, le numéro de version de la calculatrice, le lien sur GitHub et le logo de notre école IMT Mines Alès.
- Les boutons w et b pour activer le thème blanc ou noir en fonction des goûts de l'utilisateur. La couleur d'arrière-plan et la couleur du texte ont été modifiées pour les rendre plus visibles en noir ou en blanc.
- Le bouton d'une image d'un œuf de Pâques pour accéder à un tout nouveau thème entièrement personnalisé avec de nouvelles couleurs, des textes animés et un tout nouveau curseur.

Conclusion

Ce projet de la calculatrice nous a permis de découvrir l'architecture MVC et nous comprenons pourquoi le modèle MVC est utilisé pour développer des applications modernes à l'aide d'interfaces utilisateur. Il fournit des éléments importants pour la conception d'une application de bureau ou mobile, ainsi que des applications Web modernes.

Enfin, nous voulons mettre l'accent sur le travail d'équipe, car il a permis de réduire nos erreurs et nous a permis de passer en revue le code entre nous pour améliorer la qualité.

Améliorations possibles :

Diviser la classe GUI en plusieurs sous classes ?

Liens du projet :

GitHub: https://github.com/Ayfred/Calculatrice_polonaise