Rapport de Projet : Calculatrice/Convertisseur Franc CFA - Euro

Étudiants : DOGBE KODJO Hénoc et YERIMA Alassani
Date de remise : 04 avril 2025

---

1. Introduction

Ce projet, réalisé dans le cadre du cours de Programmation Orientée Objet (POO) à l'ESAG-NDE, vise à développer une application combinant une calculatrice simple et un convertisseur de devises (Franc CFA ↔ Euro). L'objectif est de mettre en pratique les concepts de POO tout en créant un outil fonctionnel et intuitif. Les principales fonctionnalités incluent :

- **Calculatrice** : opérations arithmétiques de base (+, -, ×, ÷, pourcentage).
- **Convertisseur** : conversion instantanée entre Franc CFA et Euro avec des taux ajustables.
- **Historique** : sauvegarde des derniers calculs et conversions.
- **Interface graphique** : inspirée de l'iPhone, avec animations et design ergonomique.

---

2. Analyse des besoins

Le projet répond aux exigences suivantes :

- **Calculatrice** :
  - Opérations de base (addition, soustraction, multiplication, division).
  - Gestion des nombres décimaux, effacement (AC), et correction (⌫).
- **Convertisseur** :
  - Conversion bidirectionnelle Franc CFA (XOF) ↔ Euro (EUR).
  - Modification dynamique des taux de change (par défaut : 1 EUR = 655,96 XOF).
- **Interface utilisateur** :
  - Design moderne avec animations interactives.
  - Affichage clair des opérations en cours et des résultats.

---

3. Conception

Architecture du projet :

- **Package :** com.calculatrice (regroupe toutes les classes).
- **Classes principales :**
    - **CalculatriceConvertisseur.java** :

      La classe principale qui sert de point d'entrée à l'application
        - Contient la méthode main()
        - Gère l'interface principale de la calculatrice
        - Délègue certaines fonctionnalités aux autres classes
    - **ConvertisseurManager.java** :

      Gère le convertisseur de devises
        - Contient la logique du convertisseur
        - Gère la modification des taux de change
    - **HistoryManager.java** :

      Gère l'historique des calculs
        - S'occupe de l'affichage de la fenêtre d'historique
        - Permet de mettre à jour la vue d'historique
    - **StyleManager.java** :

      Gère les styles d'interface
        - Centralise tous les styles des boutons
        - Contient la méthode showAlert() pour afficher des messages d'erreur

Diagramme de classes simplifié :

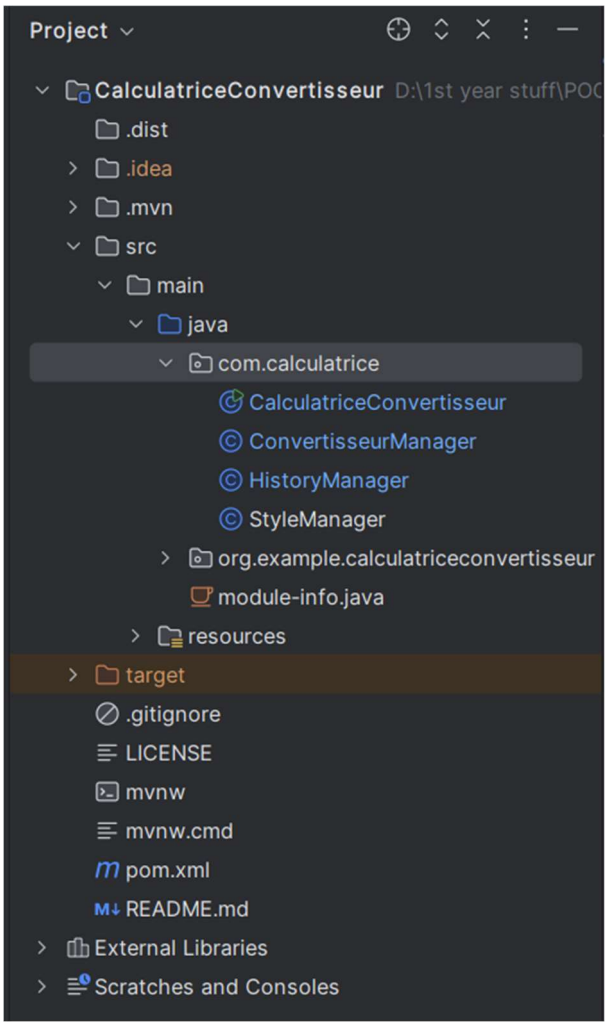
CalculatriceConvertisseur (UI)

│

├── ConvertisseurManager (Conversion + Taux)

├── HistoryManager (Historique)

└── StyleManager (Styles + Alertes)



---

4. Implémentation

Technologies utilisées :

- **JavaFX** : Pour l'interface graphique (fenêtres, boutons, animations).
- **Java 17** : Pour la logique métier et la gestion des événements.
- **Modèle MVC** : Séparation claire entre l'interface (Vue) et la logique (Contrôleur).

Fonctionnalités clés :

- **Calculatrice :**
  - Chaînage d'opérations (ex : $5 + 3 \times 2 = 16$).
  - Gestion des erreurs (division par zéro).
- **Convertisseur :**
  - Synchronisation automatique des taux (si 1 EUR = 655 XOF, alors 1 XOF = 1/655 EUR).
  - Fenêtre de modification des taux avec validation des entrées.
- **Historique :**
  - Stockage temporaire en mémoire (liste de chaînes).

- **Animations :**
  - Effets de zoom sur les boutons au clic.
- **Style :**
  - Design inspiré de l'iPhone (couleurs, bordures arrondies, ombres portées).

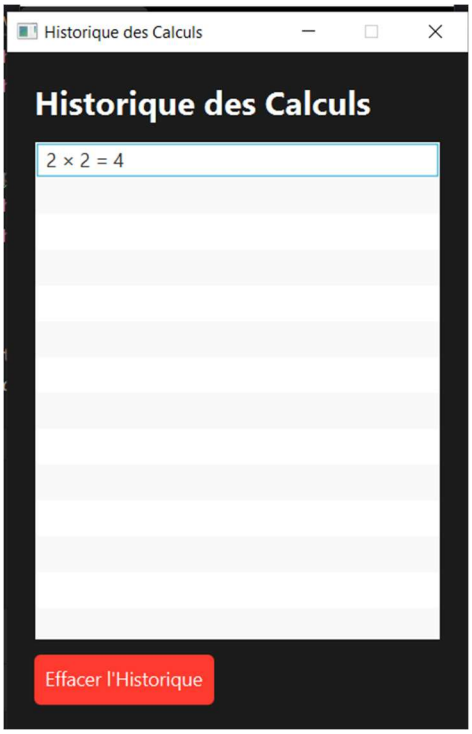
---

5. Tests et validation

Les tests suivants ont été effectués :

- **Calculatrice :**
  - Vérification des opérations arithmétiques ($10 \div 2 = 5$, 50% de $200 = 100$).
  - Test de la gestion des erreurs (ex : $5 \div 0$ affiche "Error").
- **Convertisseur :**
  - Conversion avec taux par défaut (100 EUR → 65 596 XOF).
  - Modification manuelle des taux (ex : 1 EUR = 700 XOF → conversion validée).
- **Historique :**
  - Vérification de la limite de 5 entrées et de l'effacement.
- **Interface :**
  - Tests de réactivité (clic, animations, transitions fluides).

6. Captures d'écran



Capture 1 **Capture 2** **Capture 3**

- **Capture 1 :** Interface principale de la calculatrice.
- **Capture 2 :** Fenêtre de conversion avec taux modifiables.
- **Capture 3 :** Historique des opérations.

---

7. Difficultés rencontrées et solutions

- **Problème 1 : Synchronisation des taux de change.**
  - **Solution :** Implémentation de `PropertyListener` pour lier les champs de saisie.
- **Problème 2 : Affichage des nombres décimaux.**
  - **Solution :** Formattage conditionnel (ex : afficher "5" au lieu de "5.0").

- **Problème 3 : Gestion des événements JavaFX.**
    - **Solution :** Utilisation de `Lambda Expressions` pour simplifier le code.

---

8. Perspectives d'amélioration

- Ajouter d'autres devises (USD, GBP).
- Sauvegarder l'historique dans un fichier ou une base de données.
- Implémenter des opérations scientifiques (racine carrée, exponentiel).
- Améliorer l'accessibilité (mode sombre, taille de police ajustable).

---

9. Conclusion

Ce projet a permis de consolider nos compétences en POO et en développement d'interfaces graphiques avec JavaFX. Nous avons appris à :

- Structurer une application en classes spécialisées.
- Gérer des événements complexes et des animations.
- Travailler en binôme avec Git pour le versionnement.
  À terme, cette expérience sera utile pour des projets plus ambitieux intégrant des bases de données et des APIs externes.

Annexe : Procédure d'installation de JavaFX sur NetBeans et Eclipse

A. Installation sur NetBeans

1. **Télécharger JavaFX SDK** :
    - Rendez-vous sur le site Gluon HQ et téléchargez la version de JavaFX compatible avec votre JDK (ex : JavaFX 21 pour JDK 21).
    - Extrayez le dossier `.zip` dans un répertoire de votre choix (ex : `C:\javafx-sdk-21`).
2. **Configurer un projet JavaFX dans NetBeans** :
    - Créez un nouveau projet Java > *Java with Ant* > *Java Application*.
    - Cliquez droit sur le projet > *Properties* > *Libraries* > *Add JAR/Folder*.
    - Sélectionnez tous les fichiers `.jar` du dossier `lib` de JavaFX (ex : `javafx-sdk-21\lib\*`).
3. **Ajouter les arguments VM** :
    - Dans *Properties* > *Run*, ajoutez ces arguments VM (adaptez le chemin) :

--module-path "C:\javafx-sdk-21\lib" --add-modules javafx.controls,javafx.fxml

4. **Vérifier la configuration** :
    - Redémarrez NetBeans et exécutez un exemple de code JavaFX pour confirmer l'installation.

B. Installation sur Eclipse

1. **Installer le plugin e(fx)clipse (optionnel mais recommandé)** :
    - Dans Eclipse, allez dans *Help* > *Eclipse Marketplace*.
    - Cherchez *e(fx)clipse* et installez le plugin.
2. **Télécharger JavaFX SDK** :
    - Suivez les mêmes étapes que pour NetBeans (téléchargement et extraction du SDK).
3. **Configurer un projet JavaFX** :
    - Créez un nouveau projet Java > *JavaFX Project*.
    - Si le plugin e(fx)clipse est installé, le projet sera préconfiguré. Sinon :
        - Cliquez droit sur le projet > *Build Path* > *Configure Build Path*.
        - Sous *Libraries*, ajoutez les fichiers `.jar` de JavaFX (dossier `lib`).
4. **Définir les arguments VM** :
    - Cliquez droit sur le projet > *Run As* > *Run Configurations*.
    - Sous *Arguments*, dans *VM arguments*, ajoutez :

```
--module-path "C:\javafx-sdk-21\lib" --add-modules javafx.controls,javafx.fxml
```

5. **Tester l'installation** :
    - Exécutez un exemple de code JavaFX (ex : une fenêtre vide avec un bouton).

Remarques importantes

- **Compatibilité** : Assurez-vous que la version de JavaFX correspond à votre JDK (ex : JavaFX 21 pour JDK 21).
- **Erreurs courantes** :
    - Si Eclipse affiche `Error: JavaFX runtime components are missing`, vérifiez les arguments VM.
    - Sous Linux/Mac, utilisez des barres obliques (`/`) dans les chemins (ex : `/home/user/javafx-sdk-21/lib`).
- **Documentation** : Consultez OpenJFX pour des guides détaillés.

Cette procédure garantit une configuration optimale de JavaFX pour développer et exécuter l'application *Calculatrice/Convertisseur*