Compte Rendu

Projet Programmation Fonctionnelle

Amiour Amir Tahar

Ouared Malik

Application de gestion de recettes JavaFX

Introduction

Ce projet consiste en la création d'une application Java pour manipuler des informations provenant d'un fichier de recettes, en utilisant le pattern Map/Filter/Reduce et l'API Stream de Java. L'application est réalisée en utilisant JavaFX pour l'interface graphique.

Prérequis

- Java Development Kit (JDK) installé sur votre machine
- Un éditeur de texte ou un environnement de développement intégré (IDE) tel qu'Eclipse ou IntelliJ IDEA
- adapter le path de fichier data/recipes.xml dans les classes Controller selon son emplacement dans votre machine

Utilisation de l'application

Une fois l'application lancée, vous aurez les options suivantes :

- **Bouton Graphique** : Permet de visualiser les recettes et les questions relatives aux recettes sous forme graphique.
- **Bouton Texte** : Permet d'afficher différentes informations sur les recettes dans la console texte.

Lorsque vous appuyez sur le bouton Texte, une série de questions vous est posée dans la console texte. Vous pouvez saisir le numéro de la question pour obtenir la réponse correspondante. L'application continue de poser des questions jusqu'à ce que vous décidiez de quitter en répondant "non" ou "n".

Remarque importante : Pour lancer l'application, il faut exécuter la classe HelloApplication, qui lance l'interface graphique principale. L'utilisateur est d'abord invité à sélectionner le mode de présentation (textuel ou graphique) puis à choisir parmi les traitements possibles sur la collection de recettes.

Organisation du Projet

Le projet est organisé en plusieurs packages :

- ***models* :** Contient la classe Recipe qui modélise une recette.
- ***repositories* :** Contient la classe RecipeRepo qui modélise une collection de recettes et fournit les méthodes pour effectuer les traitements demandés.
- ***presentation* :** Contient les classes qui fournit l'interface utilisateur en utilisatn JavaFX

- ***data :*** Contient le fichier recipes.xml

Traitements Disponibles

<u>Les traitements disponibles dans l'application incluent :</u>

- Afficher les titres des recettes.
- Calculer le nombre total d'œufs utilisés.
- Retourner les recettes utilisant l'huile d'olive.
- Calculer le nombre d'œufs par recette.
- Retourner les recettes fournissant moins de 500 calories.
- Retourner la quantité de sucre utilisée par la recette "Zuppa Inglese".
- Afficher les 2 premières étapes de la recette "Zuppa Inglese".
- Retourner les recettes qui nécessitent plus de 5 étapes.
- Retourner les recettes qui ne contiennent pas de beurre.
- Retourner les recettes ayant des ingrédients en commun avec la recette "Zuppa Inglese".
- Afficher la recette la plus calorique.
- Retourner l'unité la plus fréquente.
- Calculer le nombre d'ingrédients par recette.
- Retourner la recette comportant le plus de matière grasse (fat).
- Calculer l'ingrédient le plus utilisé.
- Afficher les recettes triées par nombre d'ingrédients.
- Afficher pour chaque ingrédient, les recettes qui l'utilisent.
- Calculer la répartition des recettes par étape de réalisation.
- Calculer la recette la plus facile (avec le moins d'étapes).
- Remarques Supplémentaires
- L'application a été développée en utilisant Java 11 et JavaFX pour l'interface graphique.
- Les dépendances nécessaires sont incluses dans le projet.
- Pour toute question ou problème, veuillez contacter les développeurs.

Processus de développement :

<u>Travail individuel :</u> Chaque membre de l'équipe a travaillé individuellement sur sa propre

version du projet, en développant les fonctionnalités selon sa compréhension des exigences et en utilisant ses propres compétences techniques.

Proposition de fonctionnalités : À la fin de la phase de développement individuel, chaque membre a présenté ses fonctionnalités. Cela comprenait une démonstration des fonctionnalités implémentées et une explication de leur pertinence par rapport aux objectifs du projet.

Comparaison et évaluation : Après avoir examiné les fonctionnalités proposées par chaque membre, nous avons procédé à une comparaison approfondie pour évaluer leur efficacité, leur qualité et leur adéquation par rapport aux besoins du projet. Nous avons également pris en compte des aspects tels que la facilité d'utilisation, la maintenance et la scalabilité.

Sélection des meilleures fonctionnalités : En utilisant les critères d'évaluation établis, nous avons identifié les fonctionnalités les plus performantes et les mieux adaptées aux objectifs du projet. Cette sélection a été basée sur un consensus au sein de l'équipe, en tenant compte des avantages et des inconvénients de chaque proposition.

Intégration dans le produit final : Les fonctionnalités sélectionnées ont été intégrées dans le produit final, en veillant à ce qu'elles fonctionnent de manière harmonieuse et cohérente avec les autres éléments du projet. Des tests supplémentaires ont été effectués pour garantir la qualité et la fiabilité des fonctionnalités intégrées.

En adoptant cette approche collaborative, on a pu tirer parti des compétences individuelles tout en garantissant que seules les meilleures fonctionnalités étaient incluses dans le produit final. Cela nous a permis de créer un produit de haute qualité qui répond aux besoins et aux attentes de nos utilisateurs.

Remarque supplementaire :

L'application a été développée en utilisant Java et JavaFX(version 17) pour l'interface graphique .

Les dépendances nécessaires sont incluses dans le projet .

Conclusion :

En conclusion, notre projet s'est déroulé de manière collaborative et productive, avec chaque membre apportant sa contribution unique. Grâce à un processus de sélection rigoureux, nous avons intégré les meilleures fonctionnalités de chaque proposition, ce qui a abouti à un produit final robuste et bien équilibré. Cette approche nous a permis de tirer parti des forces individuelles de chaque membre tout en garantissant la qualité et l'efficacité de notre solution.