Back to the Muffin Shop Part II

Sujet

Souvenez-vous, la pâtisserie Christian avait fermé, vous aviez décidé de la racheter et de la transformer en une boutique de muffins nommée **Muffinatort (et à travers)**. Et *Muffinatort (et à travers)* a connu un succès fulgurant.

Vous aviez entrepris de nombreuses modifications de votre application (elles sont détaillées juste après) et il est maintenant de votre ressort de :

- proposer un plan de test de l'application en cohérence avec la stratégie de test donnée ci-dessous ;
- concevoir et exécuter les tests unitaires et d'intégration selon le plan que vous aurez établi.

L'évaluation portera sur ces deux principaux points qui doivent donc être cohérents. Il ne vous est pas demandé de concevoir et exécuter les tests pour l'ensemble de l'application, ni de la redévelopper. Vous devez identifier les tests les plus stratégiques, les concevoir, puis les mettre en oeuvre dans le temps imparti.

Consignes générales et rendu

- Ce projet est à réaliser **en binôme**.
- Le rendu est à faire sur votre dépôt git, **dans une branche git nommé "rendu"**.
- Pensez à prendre le temps nécessaire pour réaliser le rendu final sur votre branche avant le **9 avril à 23h59**.
- Ce n'est pas votre rôle de développer l'application en elle-même, ni de corriger d'éventuels bugs si vous en trouvez lors de vos tests. Le code complet de l'application vous est fourni en ce sens.

Vous devrez fournir :

- le code source.
- un document markdown **pdt.md** détaillant votre plan de test et expliquant vos choix. Ce document doit également indiquer quels tests unitaires et d'intégration vous comptez idéalement concevoir et exécuter. Il doit aussi indiquer ceux que vous aurez effectivement mis en oeuvre, accompagnés d'une explication s'il y a un décalage entre les tests prévus et les tests réalisés. Vous devrez enfin discuter de la couverture de vos tests.
- un document markdown **rapport.md** où vous présenterez un rapport des tests unitaires et d'intégration que vous aurez mis en oeuvre. Ce sera aussi le lieu pour détailler d'éventuels bugs que vous auriez rencontrés. Vous pourrez aussi faire part de toute information pertinente

dans une section dédiée de ce document (problème, bug trouvé dans l'application *Muffinatort (et à travers)*, remarque, joie de revoir des muffins , etc.).

Un niveau de français correct est attendu et comptera dans la notation.

Votre dépôt devra être **mis à jour au minimum après chaque séance de TP** de façon à ce que votre démarche puisse être évaluée sur toute la durée du projet.

Préparation du dépôt Git

- Créez un groupe nom1-nom2 où nom1 et nom2 sont les noms de famille des deux membres du binôme.
- Forkez ce dépôt dans le groupe créé ci-dessus.
- Votre dépôt doit être privé et vous devez ajouter votre enseignant de TD comme Reporter.
- Pensez à "commiter", "pusher" et "merger" au fur et à mesure, ce sera vérifié.

Application *Muffinatort (et à travers)*

1. Fonctionnalités

L'application permet de :

- ouvrir, nettoyer et fermer la boutique ;
- vendre un muffin à un client renseigné ou non ;
- organiser la vitrine où sont vendus les muffins, celle-ci devant toujours être triée ;
- cuisiner un ou plusieurs muffins ;
- modifier les prix de vente des muffins ;
- modifier les coûts de production de la pâte à muffin ;
- afficher les dernières ventes de muffins ;
- afficher diverses informations autour de la vente de chaque type de muffin ;
- gérer les clients de la boutique.

1.0 Représentation d'un muffin

Un muffin est caractérisé par :

- un parfum ;
- une saveur sucrée ou salée ;
- une date de fin de cuisson ;
- une durée de cuisson ;
- une durée de conservation ;

- un coût de production ;
- un prix de vente.

Pour un parfum de muffin donné, la présence de sucre, la durée de cuisson, la durée de conservation et le prix de vente sont communs. Le coût de production (de la pâte à muffin) est quant à lui commun à toutes les variétés de muffin.

Il existe dans la boutique 11 variétés de parfum pour les muffins : Azuki, Myrtilles, Gâteau, Carotte, Fromage, Lardons, Citron, Champignons, Cacahuètes, Galaxie (sucre pétillant) et Licorne (barbe à papa). Il y a donc en vente 11 types de muffins avec des propriétés différentes.

1.1 - Ouvrir, nettoyer et fermer la boutique

L'ouverture de la boutique permet la mise en vitrine de muffin ainsi que leur vente. Inversement, une boutique fermée ne permet pas la vente, ni la mise en vitrine : le cas échéant, de nouveaux muffins cuisinés sont jetés. La fermeture de la boutique entraîne aussi le vidage de tous les muffins de la vitrine. D'une ouverture à une autre, le montant total issu de la vente de muffins est affiché à l'utilisateur de l'application ainsi que les bénéfices réalisés. Les muffins ayant une durée de conservation, celle-ci doit être accessible à l'utilisateur de l'application. Le nettoyage de la boutique consiste à retirer de la vitrine tout muffin dont la durée de conservation est dépassée. Le nettoyage des muffins entre une perte sèche de revenus quand des muffins doivent être jetés : la caisse de la boutique doit alors prendre en compte cette perte entre deux ouvertures.

1.2 - Vendre un muffin à un client renseigné ou non

La vente de muffin consiste à échanger l'un des muffins en vitrine contre sa valeur à un client. Le client peut faire parti de la base de données client de la boutique, auquel cas il peut être renseigné lors de la vente d'un muffin. Il est logiquement impossible de vendre un muffin qui ne serait pas en vitrine. Fort de l'intarissable succès de la boutique, il n'est pas envisagé de vendre directement plusieurs muffins à un client, qu'il soit renseigné ou non : le Muffin Gâteau (Cakffin) est là pour les plus gourmands. Qu'il est bon de pouvoir partager vos muffins avec le plus grand nombre ! Une unique caisse comptabilise et affiche le montant total des ventes de tous les muffins entre une ouverture et une fermeture de la boutique. Elle est remise à zéro à chaque ouverture de boutique. La caisse renseigne également sur les bénéfices liées à la vente des muffins qui se traduisent par la différence entre le prix de vente d'un muffin et les coûts de production liés à la pâte à muffin et les parfums.

1.3 - Organiser la vitrine où sont vendus les muffins

Votre vitrine de muffins est caractérisée par une quantité de muffins qu'il est possible de disposer. Si cette quantité est atteinte, il n'est alors pas possible de garder de nouveaux muffins et ceux-ci sont alors jetés (je sais c'est dommage), ce qui se traduit aussi par une perte de revenus comptabilisée par votre caisse. Votre petite boutique ne disposant pas d'autres espaces de stockage, tout nouveau muffin cuit et prêt à vendre est alors directement mis en vitrine si possible.

Pour votre confort ou celui des clients, il est bien sûr possible de ranger de différentes façons vos muffins :

- selon leur prix de vente ;
- selon leur parfum ;
- selon leur durée de conservation ;
- selon la saveur sucrée ou salée.

Les muffins de la vitrine n'ont pas besoin de pouvoir être triés par ordre croissant ou décroissant.

1.4 - Cuisiner un ou plusieurs muffins

La boutique dispose d'un four caractérisé par une capacité pour cuisiner les muffins. Il est possible de préparer une commande visant à cuisiner un ou plusieurs muffins selon la durée de cuissons de chacun des muffins et aussi la capacité du four. Si plusieurs muffins sont à cuire, les muffins dont la durée de conservation la plus longue doivent d'abord être mis au four. Si le nombre de muffins à cuire est supérieur à la capacité du four, alors les muffins sont cuits en plusieurs fournées. Les muffins d'une fournée sont directement ajoutés en vitrine si possible et si la boutique est ouverte bien sûr. Un temps de préparation d'une fournée est à prendre en compte avant d'enfourner des muffins (5 secondes). Il est également possible de lancer une cuisson express d'un muffin pour lequel il n'y pas de temps de préparation et qui est cuit indépendamment des fournées de muffins. Il n'y a pas de priorité entre différentes préparations de commandes de muffins. Si le four est utilisé pour cuire des muffins, la cuisson ne peut être interrompue pour en démarrer une nouvelle. Enfin, la durée de cuisson d'une fournée de muffins est définie par la durée maximale de cuisson des muffins de la fournée. L'interface de l'application doit mettre à disposition de l'utilisateur le nombre de muffins actuellement en cours de cuisson, le nombre de muffins restant à cuire et indiquer l'indisponibilité du four.

1.5 - Modifier les prix de vente des muffins

Chaque variété de muffin a son prix de vente qu'il est possible de modifier à tout moment. Il n'est en revanche pas possible de définir un nouveau prix de vente pour une variété de muffin si ce nouveau prix est inférieur aux coûts de production du muffin.

1.6 - Modifier les coûts de production de la pâte à muffin

Tous les muffins partagent la même pâte à muffin qui a son propre coût. Celui-ci est modifiable à tout moment. En revanche si ce prix est suffisamment augmenté de telle sorte à ce qu'un prix de vente d'une variété de muffin est inférieur à son coût de production, alors le prix de vente de cette variété doit être automatiquement augmenté afin de ne jamais vendre à perte. Grâce à votre potager et à vos partenaires, le coût de production des différentes saveurs est unique, non modifiable et est fixé à 1 euro.

1.7 - Afficher les dernières ventes de muffins

L'application doit afficher les informations relatives des 50 aux 100 dernières ventes de muffin. Une vente est caractérisée par une date, la variété de muffin, son prix de vente, le coût de production de sa pâte et de son parfum, et d'un email permettant d'identifier un client s'il a été renseigné.

1.8 - Afficher diverses informations autour de la vente de chaque type de muffin

L'application doit afficher les informations autour de la vente de chaque type de muffin, à savoir : le nombre vendu,le prix moyen, minimal et maximal de vente, ainsi que le bénéfice moyen réalisé pour la variété.

1.9 - Gérer les clients de la boutique

L'application doit simplement permettre de gérer une base de données de clients qui sont désignés par leur nom, leur prénom et une adresse mail qui doit être unique. Par gérer, il est entendu qu'il doit être possible de créer, modifier, afficher et supprimer des clients de la base de données et de disposer d'une interface permettant de réaliser ces opérations.

2. Conception

L'application a été développée en java et l'interface graphique avec Swing. Elle suit le motif d'architecture logicielle MVC.

2.1 Modèle

Le modèle contient une classe abstraite `Muffin` dont hérite chacune des variétés de muffin regroupées dans un package `model.muffin`. Le parfum est représenté à l'aide d'un énumération `Flavor`. L'ensemble de la boutique et des interactions entre les différents éléments constituant la boutique est définie dans la classe `MuffinShop`. Le patron de conception `Strategy` est employé pour ranger votre vitrine dont le comportement est défini dans sa propre classe `Vitrine`. La `Strategy` est définie dans son package `model.sorting` et utilise l'énumération `MuffinSorting`. Vous retrouverez également :

- une classe `Sale` pour définir une vente ;
- une classe `SaleResume` pour décrire les diverses informations autour de la vente de chaque variété de muffin ;
- une classe `SalePriceManager` pour gérer la modification des prix de vente des variétés de muffins ;
- une classe `SaleManager` pour gérer l'affichage des ventes des muffins ;
- une classe `Oven` pour gérer la cuisson des muffins ;
- une classe `Customer` pour décrire un client ;
- une classe `CustomerManager` pour gérer les clients ;
- une classe `CashRegister` pour représenter la caisse de la boutique.

L'application utilise une base de données dont la connexion et l'initialisation est gérée par la classe `DBInitialisation`. Cette base de données possède deux tables relatives aux ventes et aux clients dont les opérations sont respectivement gérées par les classes `DBSales` et `DBCustomers`.

Patron de conception `Observer` oblige, le package `model.observers` regroupe toutes les interfaces utiles à la mise en de ce patron de conception dans l'idée où la vitrine, la boutique, le four, la caisse, le gestionnaire de clients, le gestionnaire de prix de vente des muffins et le gestionnaire de vente sont observés.

Le modèle possède également une classe utilitaire `SortedComboBoxModel` associé aux listes déroulantes utilisées par l'interface graphique et propre à Swing.

2.2 Vue

L'interface graphique est composée d'une fenêtre principale regroupant plusieurs onglets, chacun étant associé à :

- la boutique (ouverture, fermeture, nettoyage, tri, vente, cuisson express) ;
- le four ;
- le gestionnaire de prix de vente des muffins ;
- l'affichage des ventes ;
- le gestionnaire de clients.

2.3 Contrôleur

Chaque onglet dispose de son propre contrôleur et l'application pouvant être fermée à tout moment, un contrôleur supplémentaire `MuffinDBController` existe afin de pouvoir fermer proprement la connexion à la base de données.

3. Stratégie de test

L'approche à adopter consiste à repartir de la description de l'ensemble des fonctionnalités afin d'identifier quels sont les besoins des utilisateurs. Les tests à concevoir doivent être priorisés en fonction de l'importance des éléments testés pour un utilisateur. La question à se poser est donc de savoir quelles fonctionnalités répondent le plus à ces besoins. La priorisation des tests s'effectuera selon trois niveaux **stratégique, important et secondaire**, où le niveau le plus fort de priorité est désigné par *stratégique* et le plus faible par *secondaire*.

Parmi tous les types de test possibles, vous devez vous concentrer sur les tests unitaires et d'intégration. En revanche, il n'est pas du tout attendu à ce que vous réalisiez : des tests d'Interface Homme Machine ; des tests de Performance ; des tests de Sécurité ; des tests de Charge ; des tests Fonctionnels / Systèmes ; des tests de compatibilité ; des tests de Limite de fonction ou sur les Types.

Votre environnement de test est constitué de votre machine sous Windows, Linux ou MacOS, de Java 17, d'une base de données locale SQLite et de FlatLaf 3.0 (dont les bibliothèques logicielles sont fournies dans le dossier *lib* de ce dépôt). Vous devez utiliser JUnit 5, et si besoin Mockito ou AssertJ, pour réaliser vos tests.