Projet - POO en java

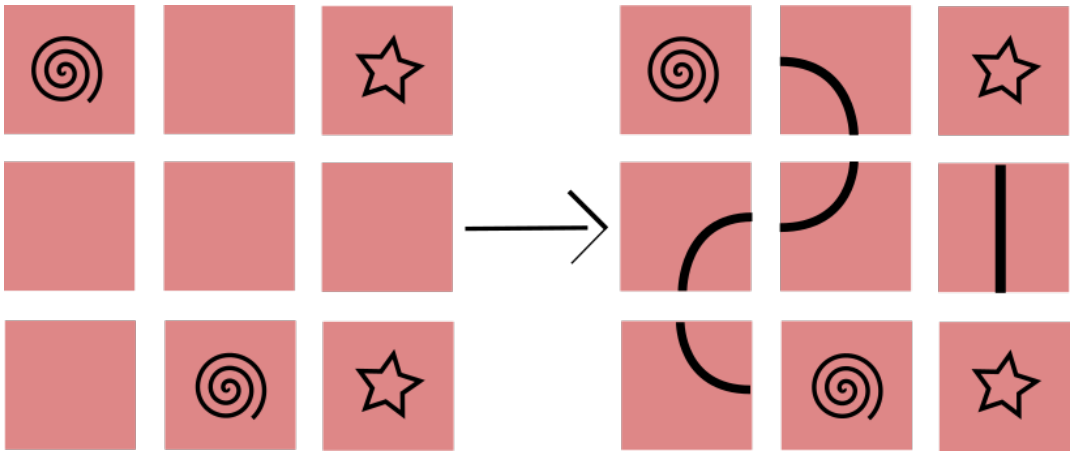
Casse tête - Lignes

Critères d'évaluation :
- Qualité de l'analyse et du code associé
- Respect du Modèle Vue Contrôleur Strict
- Modularité
- Extensions proposées

1 Sujet

1.1 Casse tête Lignes

Configuration initiale : Soit une grille de taille variable. Soit des paires de symboles placées sur la grille.



Configuration résolue : toutes les paires de symboles sont connectées. Toutes les cases de la grille sont parcourues.

1.2 Travail en binôme

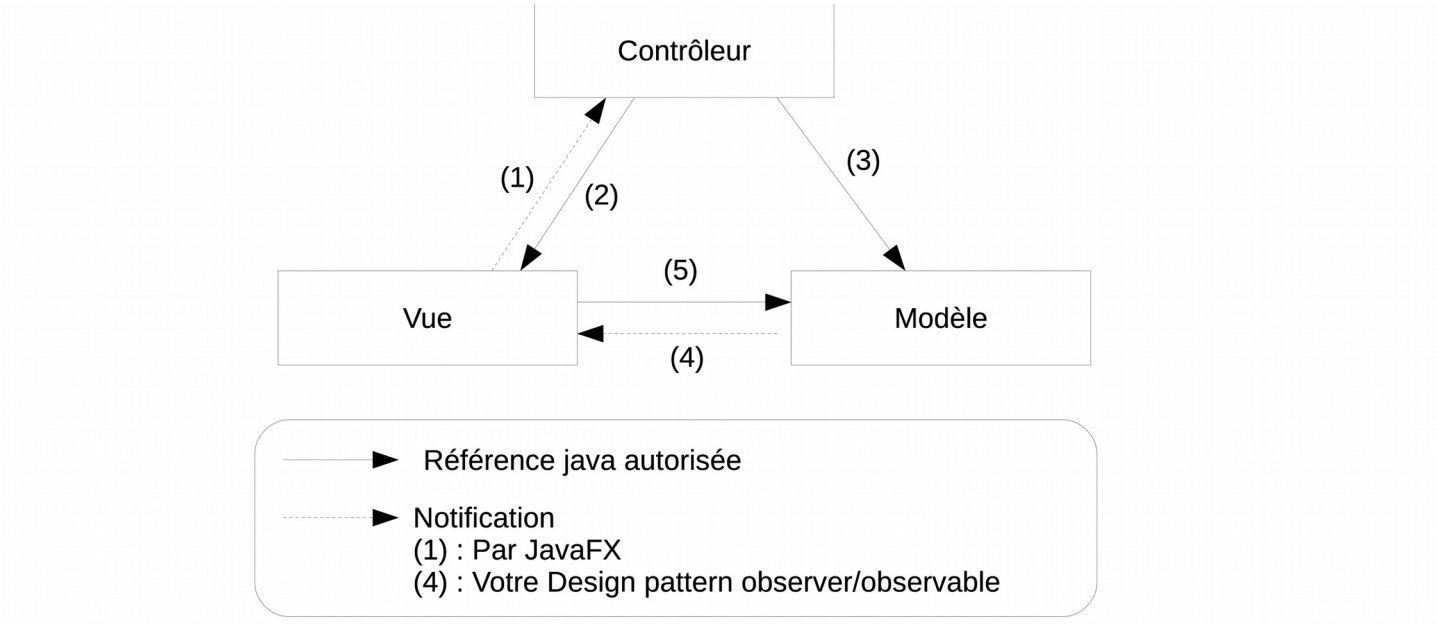
- Travail personnel entre les séances
- Évaluations individuelles
- Rapport par binôme (8 pages au maximum, listes de fonctionnalités et extensions (proportion de temps associée à chacune d'elles), documentation UML, justification de l'analyse, copies d'écran)
- Démonstration lors de la dernière séance encadrée (présence des deux binômes obligatoire)

1.3 Travail à réaliser

Développer une application graphique java la plus aboutie possible (utilisant JavaFX) du puzzle, en respectant le modèle MVC Strict. Vous êtes responsables de votre analyse, et pouvez proposer des fonctionnalités. Veillez à proposer ces fonctionnalités incrémentalement, afin d'avoir une démonstration opérationnelle le jour de la démonstration. Il est recommandé de suivre les consignes de développement données dans la partie « Étapes de l'implémentation ». **Le code doit être le plus objet possible. Privilégiez donc une programmation objet plutôt qu'un algorithme central long et complexe.**

2 Rappel Modélisation MVC Strict

2.1 MVC Strict



MVC Strict :	**Application Calculette :**
(1) Récupération de l'événement JavaFX par le controleur	(1) récupération clic sur bouton de calculette
(2) Répercutions locale directe sur la vue sans exploitation du modèle	(2) construction de l'expression dans la vue (1,2...9, (,))
(3) Déclenchement d'un traitement pour le modèle	(3) déclenchement calcul (=)
(4) Notification du modèle pour déclencher une mise à jour graphique	(4) Calcul terminé, notification de la vue
(5) Consultation de la vue pour réaliser la mise à jour Graphique	(5) La vue consulte le résultat et l'affiche

Remarque : le code associé au contrôleur et à la vue peut être réalisé dans le même fichier .java tel que dans l'application *Calculette* (utilisation de classes anonymes ou de classes internes)

2.2 Modèle

Données :

Grille, Cases (états), etc.

Processus :

Validation d'un déplacement par glissé : peut-on positionner une ligne sur une case (calculé à partir des voisins de la case concernée) ? Quel direction de ligne proposer ?
Tester la fin de partie
etc.

2.3 Vue Plateau

Pour commencer la vue de votre projet, inspirez-vous de l'application *Calculette* en JavaFX qui est disponible sur Spiral, qui respecte le MVC Strict et faites évoluer son code.

<u>Ne pas utiliser FXML de JavaFX</u>, ce n'est pas la priorité pour le projet (à moins de connaître très bien JavaFX, et de vouloir approfondir vos connaissances à ce sujet de façon autonome)

Exemple de rafraîchissements :

Afficher l'état de la grille (différentes cases à afficher, différentes paires de symboles, etc.), temps, etc.

3 Étapes suggérées pour l'analyse et l'implémentation

3.1 Analyse sur papier : Modélisation Objet du problème (classes, périmètres des fonctionnalités, principaux traitements) : étudié en CM

3.2 Connexion MVC – Modèle presque vide

Utiliser l'application calculette comme base (MVC + javaFX), et faite la évoluer (chaque le type de cases graphique suivant vos besoins, changer le modèle, etc.).

Pour cette première phase, on souhaite simplement connecter la vue à un modèle « vide », afin d'obtenir le comportement suivant :

- Contrôleur : Réceptionnez les glissés sur les cases
- Modèle : grille de positions à rafraîchir
- Vue : mettre à jour l'affichage de la grille (suivant la grille du modèle)

Ceci permet de valider un comportement MVC Strict (avec modèle très simple) opérationnel pour la suite du développement. À cette étape les processus métiers ne sont pas implémentés.

Remarque : Il est normal d'avoir une structure de grille côté modèle et une structure de grille côté vue (gérées par javaFX), c'est nécessaire pour garantir l'indépendance du modèle et de la vue. Les rôles des grilles sont différents, il ne s'agit pas d'une redondance.

3.3 Écrire les traitements du modèle

Implémenter les processus métier :
- calcul validité du glisé et état résultant pour la case concernée
- calcul position de fin
- etc.

3.4 Ajouter une ou plusieurs extensions suivant votre avancement

- Une extension de vous proposez

- IA pour générer des positions valides de paires de symboles sur différents formats de grilles

- IA pour résoudre partiellement ou globalement le puzzle

- Gestion de niveaux

- Variantes du jeu : autorisation de croisements de lignes, taille maximale/minimale pour les lignes, connecter plus de deux symboles, version 3D, etc.

- etc.