**Projet - POO en java**

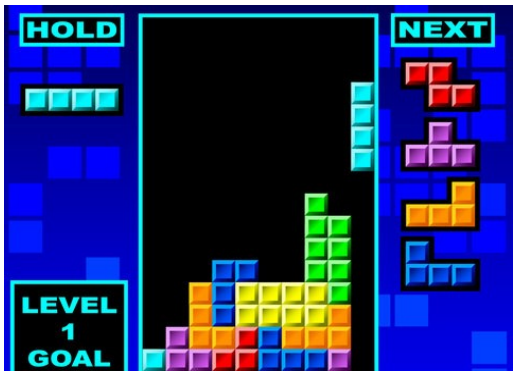
# *Jeux de plateaux et modularité*

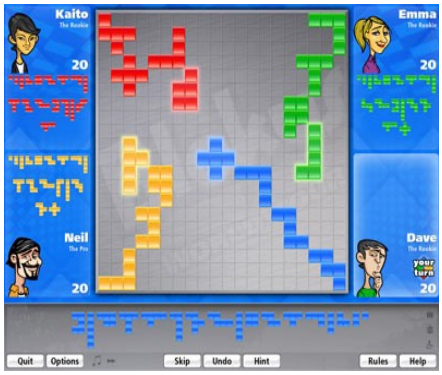
***Critères d'évaluation :***
***- Qualité de l'analyse et du code associé***
***- Respect du Modèle Vue Contrôleur Strict***
***- Modularité***
***- Extensions proposées***

# 1 Sujet

## *1.1 Tetris, Blokus et Puzzle*

Au cours de ce projet, on souhaite développer une petite bibliothèque de 3 jeux : le Tetris, le Blokus et un jeu de puzzle. Ces trois jeux exploite un plateau pour le placement de pièces. On propose ainsi de réaliser un module de gestion de plateaux, qui sera exploité pour les différentes applications.





Le travail sera ainsi réalisé en deux phases :

- développement du module de gestion de plateau (identification des besoins, analyse, développement)

- développement des différentes applications et de leurs extensions optionnelles.

## *1.2 Travail en binôme*

- Travail personnel entre les séances
- Évaluations individuelles
- Rapport par binôme (10 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)

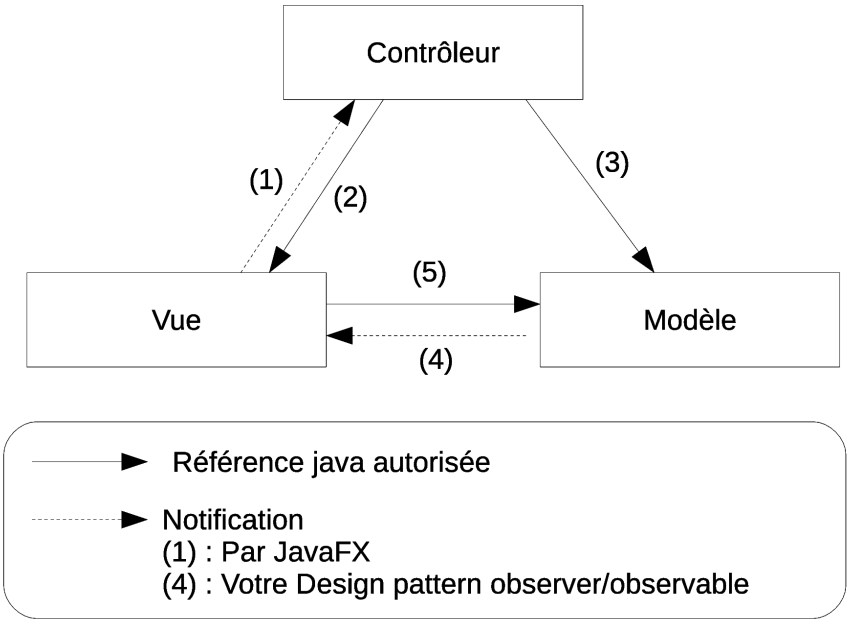
## *1.3 Travail à réaliser*

Développer une application graphique java la plus aboutie possible (utilisant JavaFX) des applications proposées de façon modulaire, en respectant le modèle MVC Strict. Vous êtes responsables de votre analyse, et pouvez proposer des fonctionnalités. Veillez à proposer ces fonctionnalités incrémentale ment, 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 modulaire plutôt qu'un algorithme central long et complexe** (par exemple, éviter de parcourir les valeurs de la grille dans la partie modèle mais distribuez le traitement : chaque

case gère ses propres contraintes).

# 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 concernant le plateau*

### Données :

Cases,
Pièces,
Grille,
etc.

### Processus :

Validation d'une position pour les cases : chaque case vérifie si elle est en conflit
Validation d'une positon pour une pièce,
Tester la position d'un pièce,
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.**

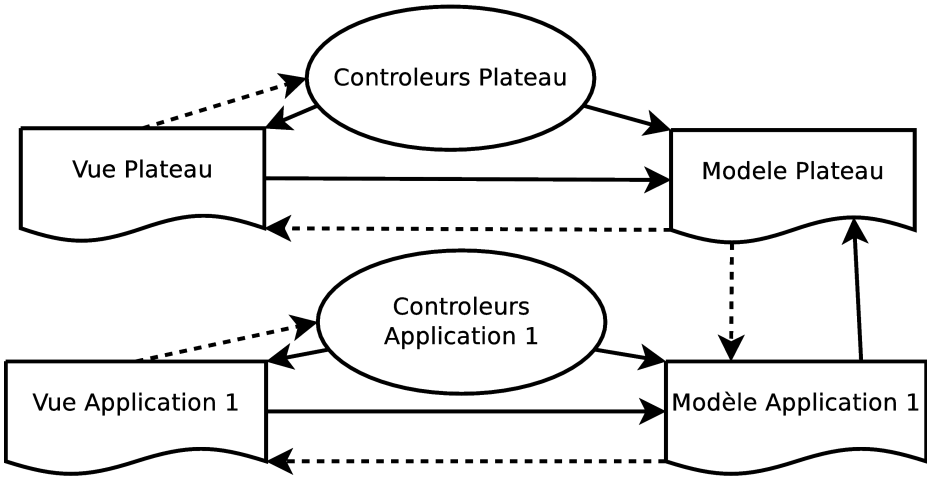
**<u>Ne pas utiliser FXML de JavaFX</u>, ce n'est pas la priorité pour le projet (à moins de connaître très bien**

**JavaFX, et de vouloir approfondir vos connaissances à ce sujet de façon autonome)**

### Exemple de rafraîchissements :

Afficher l'état du plateau (différentes cases à afficher, différentes pièces), visualiser les conflits, etc.

# 3 Intégration du plateau dans une application




# 4 Étapes suggérées pour l'analyse et de l'implémentation

## *4.1 [Plateau] Analyse sur papier : Modélisation Objet du problème (classes, périmètres des fonctionnalités, principaux traitements)*

## *4.2 [Plateau] Modèle – Vue*

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

- Contrôleur : Réceptionnez les saisies sur les cases
- Modèle : réaliser les traitements (interpréter les clics et saisies, mettre à jour les états des cases, etc.)
- Vue : mettre à jour l'affichage de la grille

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, c'est nécessaire pour garantir l'indépendance du modèle et de la vue. Les rôles des grilles sont différents, il n'y a pas de redondance.

## *4.3 [Plateau] Modèle*

Implémenter les processus métier (Sudoku) :
Suite à l'initialisation de la grille (pour commencer directement dans le code) et à son affichage, développez le traitement de test de validité des nouvelles cases renseignées, etc..

## *4.4 Ajouter les applications*

**Tetris, Blokus et le Puzzle.**

### *4.5 Ajouter une ou plusieurs extensions suivant votre avancement*

- **Une extension de vous proposez**
- [plateau + application] Sauvegarder plateaux, charger plateaux
- Meilleurs scores, chronomètre, etc.
- Aide pour l'utilisateur
- Résolution automatique
- [plateau] Plateaux triangulaire, 3D, etc.
- etc.