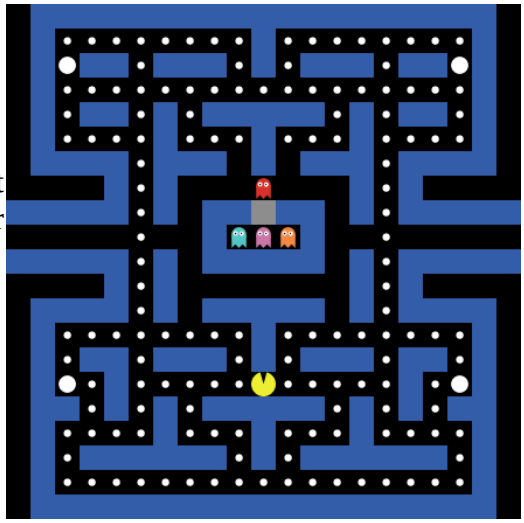
Projet - POO en java

Pac-Man

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

1 Sujet

1.1 Pac-Man



Description du jeu : https://fr.wikipedia.org/wiki/Pac-Man

Précisions concernant l'implémentation :

- Le plateau est représenté par une grille de cases du côté de la vue et une grille de cases du côté du modèle. Les murs sont matérialisés par des cases pleines, les couloirs par des cases vides.

- Les mouvements sont discrétisés : on avance case par case, pas d'autres positions intermédiaires (pixel, etc.)

- Les entités (Fantômes et PacMan) implémentent leur comportement dans une méthode ***run*** (appelée par une boucle dans la classe Jeu).

Soit une classe Pac-Man héritant de Entite définira choixDirection() afin de déterminer la direction que prendra le Pac-Man. *Concernant les Fantômes, prévoir une IA simple (définir une direction à suivre, changer aléatoirement de direction si l'entité rencontre un mur).*

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)

Note : les critères d'évaluation seront adaptés à la configuration actuelle (difficultés de communication à distance...)

1.3 Travail à réaliser

Développer une application graphique Java la plus aboutie possible (utilisant Swing) de l'application, 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 Étapes suggérées pour l'analyse et l'implémentation

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

2.2 Connexion MVC – Modèle presque vide

Commencez par lire et comprendre le code fourni (base : MVC + Swing) et vérifiez qu'il fonctionne sur votre machine (Build+Run). Il permet de valider un comportement MVC Strict opérationnel, avec modèle très simple. À cette étape, les processus métiers ne sont pas implémentés. Vous ferez évoluer ce code selon vos besoins pour ajouter ces traitements métiers (et l'affichage correspondant).

Remarque : Il est normal d'avoir une structure de grille côté modèle et une structure de grille côté vue (gérées par Swing), 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.

2.3 Écrire les traitements du modèle

Commencez par ajouter l'environnement : rajouter les classes nécessaires dans le *modèle*

- Murs
- Pac-Gommes

Ensuite, ajoutez l'affichage de cet environnement dans la *vue*, en affichant la bonne image selon l'état du modèle.

Ajoutez les processus métiers (dans le *modèle*) afin que la partie de Pacman soit jouable :

- Gestion des collisions (Pacman / Murs / Fantômes)
- Stratégie des fantômes (IA simple)
- Capacité de manger (pac-gommes, fantômes, …)
- Système de points
- Régles du jeu (mort, fin du niveau, …)
- etc.

2.4 Ajouter une ou plusieurs extensions suivant votre avancement

- Une extension que vous proposez
- IA Fantômes améliorée
- Éditeur/Générateur de Niveau
- Meilleurs scores
- Effet de *wraparound* (« boucle » sur les bords : on rentre par la gauche si on sort à droite)
- Jeu collaboratif sur un même plateau ou en réseau
- etc.

3 Évaluation

3.1 Présentation / Démo

Vous devrez présenter, en binôme, votre projet lors d'une soutenance :

- Quelques minutes de présentation (montrer le fonctionnement, préciser les choix de conception)
- Quelques minutes de questions **individuelles** (les deux binômes doivent avoir compris l'ensemble du code, et pouvoir répondre aux questions)

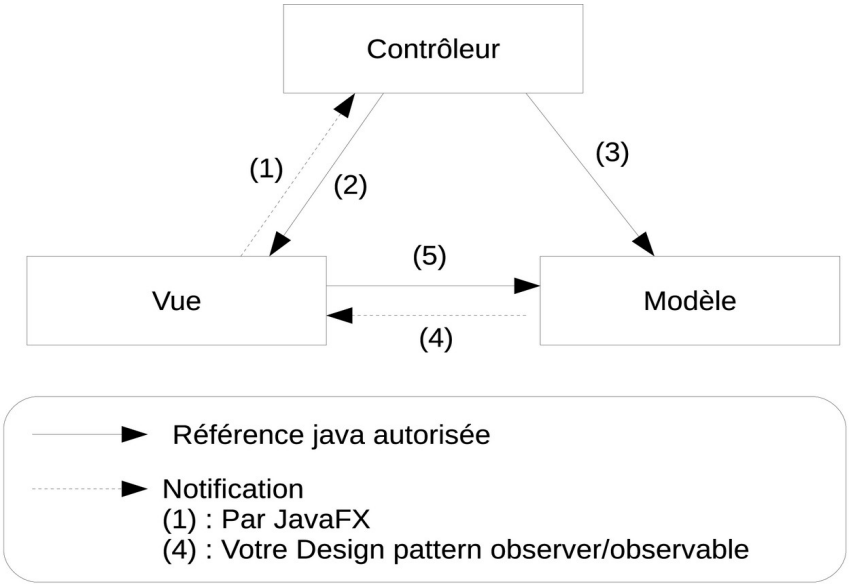
3.2 Rendu

Vous devrez rendre, sur Tomuss, une archive ***au format Zip*** contenant :
- le code source de votre projet : code Java, ressources (images, sons, …) + librairies éventuelles
- (optionnel) un Jar compilé de votre projet
- un rapport ***au format PDF***

Le rapport (8 pages maximum) doit contenir :
- la liste des fonctionnalités et extensions (proportion de temps associée à chacune d'elles)
- une documentation UML (au minimum un diagramme de classes)
- la justification de votre analyse (choix de conception)
- des copies d'écran d'une partie en cours (en particulier les fonctionnalités que vous souhaitez mettre en valeur)

4 Rappel de cours

4.1 MVC Strict



MVC Strict :
(1) Récupération de l'événement JavaFX par le controleur
(2) Répercutions locale directe sur la vue sans exploitation du modèle
(3) Déclenchement d'un traitement pour le modèle
(4) Notification du modèle pour déclencher une mise à jour graphique
(5) Consultation de la vue pour réaliser la mise à jour Graphique

Application Calculette :
(1) récupération clic sur bouton de calculette
(2) construction de l'expression dans la vue (1,2...9, (,))
(3) déclenchement calcul (=)
(4) Calcul terminé, notification de la vue
(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

4.2 Modèle

Données :

Grille, Cases (états), Entités (Pac-Man + Fantômes), Murs, Pac-Gommes, Super-Pac-Gommes, sablier, etc.

Processus :

Initialiser ; Réaliser Actions (déplacements) ; Tests de fin de partie ; etc.

4.3 Vue Plateau

Pour commencer la vue de votre projet, inspirez-vous du code fourni, puis faites évoluer ce code.

Les étudiants connaissant très bien Java et souhaitant approfondir en autonome peuvent proposer un code différent (par ex. avec JavaFX) mais ce n'est pas recommandé.