Projet - POO en Java

Simulation d'un potager

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

1 Sujet

1.1 A Végétable Garden

Ce projet consiste à simuler un potager en vue de dessus.

Les fonctionnalités de base à développer pour le projet sont les suivantes :

- l'utilisateur doit pouvoir planter/récolter différents légumes sur les cases du potager

- l'environnement doit être simulé (hydrométrie, température, ensoleillement)

- Des variétés doivent être simulées (2 à plusieurs) : conditions de croissance à définir

- La vitesse de la simulation doit pouvoir être paramétrée (vitesse accélérée pour la démo)

Précisions concernant l'implémentation :

- Le plateau est représenté par une grille de cases pour la vue, et par une grille de cases pour le modèle.

- L'utilisateur peut cliquer sur les différentes cases afin de réaliser des actions (planter, cueillir, etc.)

1.2 Travail en binôme obligatoire (ID binôme à renseigner sur Tomuss)

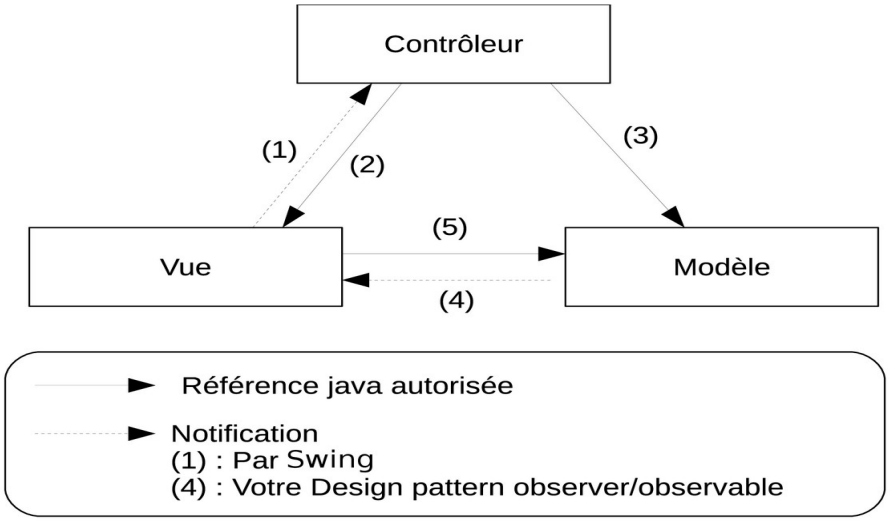
- Travail personnel entre les séances ;
- Évaluations individuelles ;
- Rapport par binôme : 6 pages au maximum, listes de fonctionnalités et extensions (indiquer la proportion de temps associée à chacune d'elles), copies d'écran, 1 diagramme UML de votre choix.
- Démonstration lors de la dernière séance encadrée (présence des deux binômes obligatoire).

1.3 Travail à réaliser

Développer une application graphique Java la plus aboutie possible (utilisant Swing) de l'application, en respectant le modèle MVC Strict. Vous êtes responsables de votre analyse, et pouvez proposer des fonctionnalités supplémentaires. Veillez à proposer ces fonctionnalités incrémentalement, afin d'avoir une démonstration opérationnelle le jour de la soutenance. **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 :
(1) Récupération de l'événement Swing 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 tel que dans l'application *Calculette* (utilisation de classes anonymes ou de classes internes).

2.2 Modèle

Données :

Cases du potager, météo, gestion du pas de simulation, etc.

Processus :

- Initialiser : création d'un potager vide ou à défricher
- Simulation : exécution d'un pas de simulation, planter/récolter sur une case
- Etc.

2.3 Vue Plateau

Pour commencer la vue de votre projet, inspirez-vous du code fourni (disponible sur Moodle), qui respecte le MVC Strict, puis faites évoluer ce code.

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 Vue/Contrôleur – Modèle presque vide

Utilisez le code fourni comme base (MVC + Swing), et faites-le évoluer (changer le type de cases graphiques suivant vos besoins, changer le modèle, etc.).

Commencez par lire et comprendre le code fourni, les relations entre les différentes classes, la gestion des événements (MVC). Vérifiez également que le code fonctionne sur votre machine (Build+Run puis Package).

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

Météo, croissance, gestion de la vitesse de simulation, etc.

3.4 Ajouter une ou plusieurs extensions suivant votre avancement

- Une extension que vous proposez vous-même ;

- Adaptation des icones suivant la croissance (taille des sprites)

- Simulation avancée (espèces compatibles, nuisibles, qualité du sol, etc.)

- Sauvegarde et meilleurs scores ;

- Niveau « scrollables » (plus grands que l'écran) ;

- Gestion de la transparence (canal alpha, dessiner le composant case), afin de visualiser le taux d'humidité du sol autour des légumes sur la case

- Simulation économique (achat/vente graines, légumes, engrais, outils, traitements, etc.)

4 Évaluation

4.1 Présentation / Démo

Vous devrez présenter, en binôme, votre projet lors d'une soutenance (4 à 7 minutes)

- Quelques minutes de démo (montrer le fonctionnement, préciser les choix de conception)

- Choisissez une partie de votre analyse pour l'expliquer / justifier (1-2 min)

- Quelques minutes de questions **individuelles** (les deux binômes doivent avoir compris **l'ensemble** du code, et pouvoir répondre aux questions). Informez votre encadrant de la répartition des tâches (50/50, 60/40, etc.)

4.2 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
- un Jar compilé de votre projet (vérifier qu'il fonctionne depuis n'importe quelle machine et charge correctement les ressources)
- un rapport ***au format PDF***

Merci de rendre une archive « propre » (rapport à la racine, un dossier pour le projet, pas d'éléments inutiles tels que les fichiers objets).

- [Attention!!] un test antiplagiat (spécifique aux codes informatiques) est exécuté sur l’ensemble des projets. Si le code est plagié, les deux groupes sont pénalisés indifféremment. Nous vous encourageons à échanger vos idées, mais pas de recopie de code.