DEPARTEMENT INFORMATIQUE
92 Bd Niels Bohr
69100 Villeurbanne

TP3 – MVC

R3.04 QUALITE DE DEVELOPPEMENT

I. INTRODUCTION

Dans ce TP vous allez découvrir le design pattern d'architecture le plus utilisé pour organiser l'IHM des applications logicielles et Web professionnelles : le design pattern MVC (modèle-vue-contrôleur). Il consiste à séparer dans un premier temps les différentes vues du modèle (comme déjà présenté dans les précédents TP), auquel s'ajoute l'intervention de classes de contrôle ayant pour mission de vérifier la cohérence des données et de mettre à jour le modèle. Seules ces classes "contrôleur" auront le droit d'accéder au modèle en écriture.

MVC s'articule autour de deux autres design patterns: le contrôleur est implémenté avec le DP stratégie, et le DP observateur gère les interactions entre les vues et le modèle.

II. PRESENTATION DU SUJET

Dans ce TP, nous vous demandons de traiter (affichage et modification) certaines données d'une promotion (fictive) d'étudiants. Nous nous intéresserons aux données suivantes d'un étudiant : numéro, nom, prénom, bac d'origine, département.

Le modèle sera constitué de deux classes: une classe Etudiant et une classe Promotion. Depuis le main, l'appel à la fonction loadFileCSV de la classe Promotion vous permettra de charger directement votre promotion qui est contenue dans le fichier "PromoBUT.csv", du répertoire "data".

Votre application se présentera sous la forme d'une fenêtre principale qui sera composée de quatre vues:

- une vue histogramme, qui affichera sous forme d'histogramme JFreeChart le nombre d'étudiants par type de bac.

- une vue camembert, qui affichera sous forme de diagramme camembert JFreeChart la proportion d'étudiants de chaque département (on n'affichera que les départements où il y a au moins une personne).

- une vue liste, qui affichera la liste des étudiants dans une JList, et qui possèdera un bouton supprimer pour effacer l'étudiant sélectionné de la liste.

- une vue formulaire, qui permet d'ajouter un étudiant, ou de le supprimer à partir de son numéro d'étudiant.

Adrien Peytavie – Eric Duchene

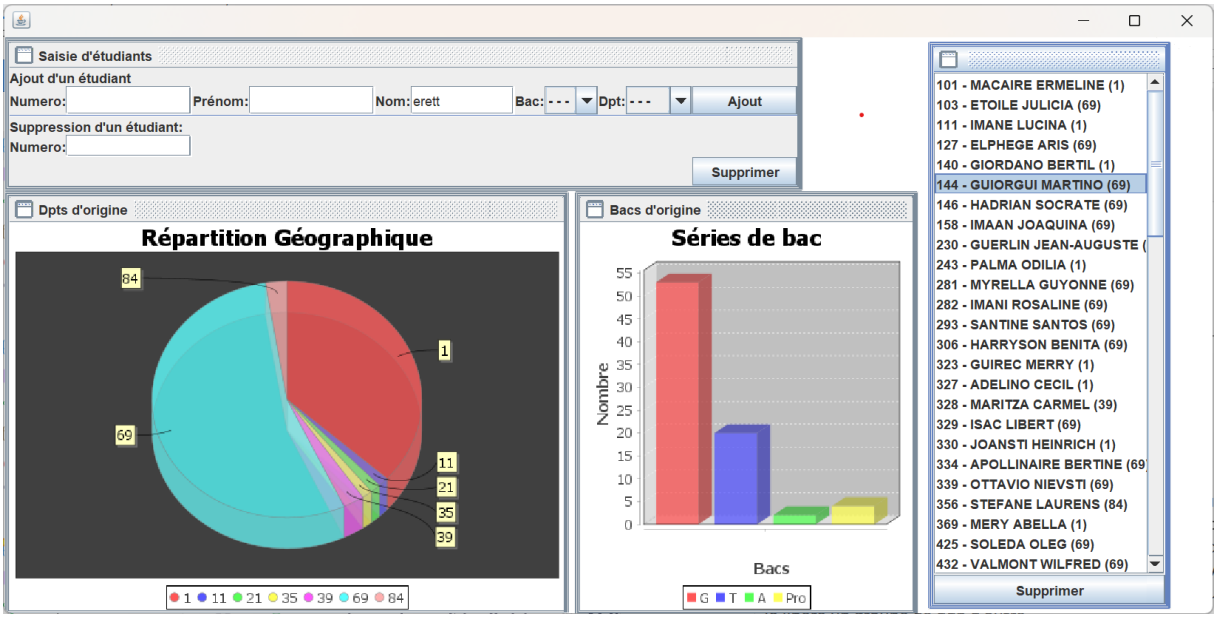
Toutes ces vues seront placées à l'intérieur du panneau principal (JDesktopFrame) de la fenêtre principale. Elles devront hériter de la classe JInternalFrame.

Fonctionnalités attendues:

- l'ajout/suppression d'un étudiant via la vue formulaire doit mettre à jour automatiquement les trois autres vues et le modèle. Par ailleurs, votre programme devra vérifier que les données rentrées dans le formulaire sont valides (cela sera un des rôles du contrôleur). Si ce n'est pas le cas, la vue affichera un message d'erreur.

- la suppression d'un étudiant via la vue liste doit mettre à jour cette même vue, le modèle, ainsi que les vues histogramme et camembert. Le contrôleur de cette vue vérifiera que l'étudiant sélectionné appartient bien à la promotion.

Votre enseignant vous fera une démonstration de ce qui est attendu. Nous la complétons avec la copie d'écran suivante :



III. DIAGRAMME DE CLASSES ET DEMARRAGE DU PROJET

Avant de commencer tout codage, la construction du diagramme de classes est cruciale pour un tel projet où de nombreuses classes sont interconnectées. Avec votre enseignant, vous allez le construire en respectant les règles du design pattern MVC. Vous découvrirez notamment avec lui les points suivants:

- le DP stratégie pour les contrôleurs. Le DP contrôleur permet aux vue d'accéder au modèle pour se rafraîchir, mais ne leur permet par l'accès en écriture. C'est un des rôles du contrôleur.

- l'utilisation de classes internes/anonymes pour les vues (notamment pour les écouteurs des boutons des vues liste et formulaire).

- le DP observateur entre les vues et le modèle: les vues vont être "observateur" du modèle qui sera "observable". Des interfaces *observateur* et *observable* devront être créées.

- les relations entre les classes M, V et C, et l'intérêt de séparer le code en trois types de classes.

IV. IMPLEMENTATION DE MVC

Pour simplifier le TP, et éviter de perdre trop de temps avec l'interface, nous vous fournissons un squelette de l'application contenant une partie de la Vue, mais qui affiche des données constantes et fictives. Celui-ci est disponible sur Moodle.

On construira le projet progressivement, en commençant par la fenêtre VueListe, puis VueFormulaire, puis VueCamembert et enfin VueHistogramme.

4.1 Mettre en place le modèle : la classe Etudiant et la classe Promotion qui se construit à partir de la lecture du fichier.

4.2 Compléter les classes VueListe et sa classe interne EcouteurSuppr2 qui écoute le bouton de suppression de VueListe. EcouteurSuppr2 peut être remplacée par une classe anonyme.

4.3 Mettre en place le Design Pattern Observer : Promotion hérite de java.util.Observable (ou de votre propre classe Observable) et VueListe implémente java.util.Observer (ou votre propre classe Observateur).

4.4 Mettre en place le contrôleur ControleurSuppressionListe.

4.5 Faites vérifier le bon fonctionnement de la fenêtre VueListe à votre enseignant.

4.6 Reprendre la démarche pour fenêtre VueFormulaire et faites vérifier son bon fonctionnement.

4.7 Mettre à jour les fenêtres VueCambembert et VueHistogramme pour que leur affichage corresponde au données du modèle et qu'elles se mettent à jour automatiquement. Faites vérifier leur bon fonctionnement.

V. AMELIORATION DE L'APPLICATION

5.1 On désire maintenant pouvoir sélectionner plusieurs étudiants dans la liste et les supprimer simultanément par l'appui sur le bouton « supprimer ». Modifier votre architecture en conséquence.

5.2 On désire également pouvoir modifier un étudiant par un double clic dans la liste. Modifier votre architecture en conséquence.