IHM – TP3

L'objectif de ce troisième TP est de mettre en pratique le modèle PAC. Afin de pleinement comprendre l'intérêt de cette approche, nous l'appliquerons sur l'application d'album photo réalisée dans le dernier exercice du TP précédent. Contrairement à cette fois-là, les composants seront indépendants les uns des autres et l'ajout/suppression d'une partie de l'interface ne compromettra pas le reste de l'application.

Bien que l'interface finale sera en tout point identique, l'organisation du code sera relativement différente et répartie en trois packages :

abstraction regroupe les classes du noyau applicatif : `Photo` et `Album`[1] ;

presentation regroupe les classes liées à l'interface proprement dite ;

controle regroupe les classes permettant de faire le lien entre une partie de l'interface et le modèle.

1. Ouvrir la classe `Album` et observer les modifications de code marquées d'un commentaire `//PAC`. En particulier, noter qu'à chaque modification de l'album, ce dernier mentionne qu'il a été modifié *via* `setChanged` et notifie d'un changement tous les objets qui l'observent *via* `notifyObservers`. Cette notification entraîne l'appel de la méthode `update` de chaque objet qui observe l'album en fournissant un des trois types de message suivants :
   - `Album.CHANGEMENT_IMAGE_COURANTE` si l'image courante est changée
   - `Album.CHANGEMENT_TAILLE` si l'image courante est redimensionnée
   - `Album.NOUVELLE_IMAGE` si une nouvelle image est ajoutée en fin d'album
2. Ajouter au package `presentation` une classe `AlbumPhoto` qui soit la fenêtre principale de l'application.
3. Le `JLabel` central servant à afficher l'image courante du modèle doit être mis à jour chaque fois que l'image courante est changée ou redimensionnée.
   (a) Ajouter au package `control` une classe `ControlJLabel` qui soit un observateur du modèle. Elle possède donc une référence vers le `JLabel` central (lien d'association) afin de le mettre à jour suivant les changements du modèle.
   (b) Dans la classe `AlbumPhoto`, créer une instance de `ControlJLabel` et l'ajouter à la liste des observateurs du modèle.
4. Le `JSlider` servant à afficher le niveau de zoom de l'image courante doit être mis à jour chaque fois que l'image courante est changée. De plus, le modèle doit être mis à jour chaque fois que l'utilisateur modifie le niveau de zoom de l'image courante.
   (a) Ajouter au package `control` une classe `ControlJSlider` qui soit non seulement un observateur du modèle, mais aussi un écouteur du `JSlider`. Elle possède donc une référence vers le `JSlider` afin de le mettre à jour suivant les changements du modèle, et vers le modèle afin de redimensionner l'image courante si l'utilisateur déplace le curseur.
   (b) Dans la classe `AlbumPhoto`, créer une instance de `ControlJSlider` et l'ajouter à la liste des observateurs du modèle, ainsi qu'à la liste des écouteurs du `JSlider`.

1. La classe `Album` étend à présent la classe `Observable`.

5. La `JList` servant à afficher la liste des images à partir de leur nom doit être mise à jour chaque fois que le modèle change. De plus, le modèle doit être mis à jour chaque fois que l'utilisateur sélectionne une nouvelle image.
   (a) Ajouter au package `control` une classe `ControlJList` qui soit à la fois un observateur du modèle afin de mettre à jour la sélection de la `JList` si l'image courante du modèle est changée ou d'ajouter le nom de la nouvelle image si une nouvelle image est ajoutée à l'album, et un écouteur de la `JList` afin de modifier l'image courante du modèle pour qu'elle corresponde à sa sélection.
   (b) Dans la classe `AlbumPhoto`, créer une instance de `ControlJList` et l'ajouter à la liste des observateurs du modèle, ainsi qu'à la liste des écouteurs de la `JList`.
6. Les `JButton` du bandeau bas doivent être mis à jour chaque fois que le modèle change. De plus, le modèle doit être mis à jour chaque fois que l'utilisateur clique sur un des boutons.
   (a) Ajouter au package `controle` une classe `ControlJButtonBas` qui servira de contrôleur pour chaque bouton du bandeau bas. Cette classe est à la fois un observateur du modèle afin de mettre à jour la bordure de son bouton en fonction de l'image courante du modèle (`null` si l'index du bouton est différent de l'index de l'image courante, un bord bleu sinon), et est un écouteur de son `JButton` afin de modifier l'image courante du modèle pour qu'elle corresponde à son index en cas de clic.
   (b) Dans la classe `AlbumPhoto`, créer une instance de `ControlJButtonBas` pour chaque bouton du bandeau bas et les ajouter à la liste des observateurs du modèle, ainsi qu'à la liste des écouteurs de leur `JButton`. Noter que l'ajout d'un bouton dans le bandeau bas suite à l'ajout d'une nouvelle image dans l'album n'est pour l'instant pas traité.
7. Les `JButton` «précedente» et «suivante» doivent être mis à jour chaque fois que le modèle change. De plus, le modèle doit mis à jour chaque fois que l'utilisateur clique sur ces boutons.
   (a) Ajouter au package `controle` une classe `ControlJButtonPrecedente`/`ControlJButtonSuivante` qui soit à la fois un observateur du modèle afin de mettre à jour la disponibilité du bouton «précédente»/«suivante» en fonction de l'image courante du modèle (disponible si l'image courante n'est pas la première/dernière de l'album), et un écouteur du `JButton` «précédente»/«suivante» afin de modifier l'image courante du modèle en sélectionnant la photo d'index immédiatement inférieur/supérieure.
   (b) Dans la classe `AlbumPhoto`, créer une instance de `ControlJButtonPrecedente`/`ControlJButtonSuivante` et l'ajouter à la liste des observateurs du modèle, ainsi qu'à la liste des écouteurs du `JButton` «précédente»/«suivante».
8. L'item «Ajouter une image» du menu n'a pas à observer le modèle car son aspect est insensible aux modifications de l'album. En revanche, il doit pouvoir modifier l'album en y ajoutant une photo.
   (a) Ajouter au package `controle` une classe `ControlJMenu` qui soit un écouteur de l'item «Ajouter une image» du menu permettant d'ajouter une photo dans l'album.
   (b) Dans la classe `AlbumPhoto`, créer une instance de `ControlJMenu` et l'ajouter à la liste des écouteurs de l'item «Ajouter une image». Noter qu'il est normal pour le moment que l'ajout d'une photo ne rajoute pas de bouton dans le bandeau bas.
9. Le `JPanel` servant de bandeau bas doit réagir à l'ajout d'une nouvelle photo dans l'album.
   (a) Ajouter au package `controle` une classe `ControlJPanelBas` qui soit un observateur du modèle afin d'ajouter un nouveau bouton au bandeau bas lorsqu'une photo est ajoutée dans l'album. Attention, une nouvelle instance de `ControlJButtonBas` devra être mise en place pour le bouton ajouté.
   (b) Dans la classe `AlbumPhoto`, créer une instance de `ControlJPanelBas` et l'ajouter à la liste des observateurs du modèle.