# TP4

L'objectif de ce TP est de mettre en pratique le modèle PAC. Afin de pleinement comprendre l'intérêt de cette approche, nous l'appliquons sur l'application d'album photo réalisée au 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 (modèle) : `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. Ouvrez la classe `Album` et observez les modifications de code marquées d'un commentaire `//PAC`. En particulier, notez qu'à chaque modification de l'album, ce dernier mentionne qu'il a été modifié *via* `setChanged` et notifie tous les objets qui l'observent *via* `notifyObservers`, en fournissant un des trois types de message suivants en paramètre :
   - `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.

   La notification entraîne l'appel de la méthode `update` de chaque objet qui observe l'album.
2. Ajoutez au *package* `presentation` une classe `Visionneuse` correspondant à la fenêtre principale de l'application.
3. L'`ImageView` centrale servant à afficher la photo courante du modèle doit être mise à jour chaque fois que la photo courante est changée ou redimensionnée.
   (a) Ajoutez au *package* `controle` une classe `ControlePhotoCentre` qui soit un observateur du modèle. Elle possède donc une référence vers l'`ImageView` centrale afin de la mettre à jour suivant les changements du modèle.
   (b) Dans la classe `Visionneuse`, créez une instance de `ControlPhotoCentre` et ajoutez-la à la liste des observateurs du modèle.
4. Le `Slider` servant à afficher le niveau de zoom de la photo courante doit être mis à jour chaque fois que la photo courante est changée. De plus, le modèle doit être mis à jour chaque fois que l'utilisateur modifie le niveau de zoom de la photo courante.
   (a) Ajoutez au *package* `controle` une classe `ControleSlider` qui soit non seulement un observateur du modèle, mais aussi un gestionnaire d'événements du `Slider`. Elle possède donc une référence vers le `Slider` afin de le mettre à jour suivant les changements du modèle, et vers le modèle afin de redimensionner la photo courante si l'utilisateur déplace le curseur.
   (b) Dans la classe `Visionneuse`, créez une instance de `ControleSlider` et ajoutez-la à la liste des observateurs du modèle, ainsi qu'à la liste des gestionnaires d'événements de la propriété `value` du `Slider`.

1. La classe `Album` a été modifiée et étend à présent la classe abstraite `java.util.Observable`.

5. La `ListView` servant à afficher la liste des photos 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 photo.
   (a) Ajoutez au *package* `controle` une classe `ControleListe` qui soit à la fois un observateur du modèle afin de mettre à jour la sélection de la `ListView` si la photo courante du modèle est changée ou d'ajouter le nom de la nouvelle photo si une nouvelle photo est ajoutée à l'album, et un gestionnaire d'événements de la `ListView` afin de modifier la photo courante du modèle pour qu'elle corresponde à sa sélection.
   (b) Dans la classe `Visionneuse`, créez une instance de `ControleListe` et ajoutez-la à la liste des observateurs du modèle, ainsi qu'à la liste des gestionnaires d'événements de la propriété `selectedItem` du modèle de sélection de la `ListView`.
6. Les `Button` 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) Ajoutez au *package* `controle` une classe `ControleBoutonBas` 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 la photo courante du modèle (aucune si l'index du bouton est différent de l'index de la photo courante, une bordure bleue de 4 pixels sinon), et un gestionnaire d'événements de son bouton afin de modifier la photo courante du modèle pour qu'elle corresponde à son index en cas de clic.
   (b) Dans la classe `Visionneuse`, créez une instance de `ControleBoutonBas` pour chaque bouton du bandeau bas et ajoutez-la à la liste des observateurs du modèle, ainsi qu'à la liste des gestionnaires d'événements du bouton. Notez que l'ajout d'un bouton dans le bandeau bas suite à l'ajout d'une nouvelle photo dans l'album n'est pour l'instant pas traité.
7. Les `Button` « précédent » et « suivant » 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 ces boutons.
   (a) Ajoutez au *package* `controle` une classe `ControleBoutonPrecedent/Suivant` qui soit à la fois un observateur du modèle afin de mettre à jour la disponibilité du bouton « précédent »/« suivant » en fonction de la photo courante du modèle (disponible si la photo courante n'est pas la première/dernière de l'album), et un gestionnaire d'événements du bouton « précédent »/« suivant » afin de modifier la photo courante du modèle en sélectionnant la photo d'index immédiatement inférieur/supérieur.
   (b) Dans la classe `Visionneuse`, créez une instance de `ControleBoutonPrecedent/Suivant` et ajoutez-la à la liste des observateurs du modèle, ainsi qu'à la liste des gestionnaires d'événements du bouton « précédent »/« suivant ».
8. L'item *Ajouter une photo* 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) Ajoutez au *package* `controle` une classe `ControleMenu` qui soit un gestionnaire d'événements de l'item *Ajouter une photo* du menu permettant d'ajouter une photo dans l'album.
   (b) Dans la classe `Visionneuse`, créez une instance de `ControleMenu` et ajoutez-la à la liste des gestionnaires d'événements de l'item *Ajouter une photo*. Comme mentionné précédemment, il est normal pour le moment que l'ajout d'une photo ne rajoute pas de bouton dans le bandeau bas.
9. Le `Pane` servant de bandeau bas doit réagir à l'ajout d'une nouvelle photo dans l'album.

(a) Ajoutez au *package* `controle` une classe `ControleBandeauBas` 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 `ControleBoutonBas` devra être mise en place pour le bouton ajouté.

(b) Dans la classe `Visionneuse`, créez une instance de `ControleBandeauBas` et ajoutez-la à la liste des observateurs du modèle.