TD 7 : Swing, Observer

Objectifs pédagogiques : bases de swing, listeners.

7.1 Scores

Il s'agit dans cet exercice de créer un exemple minimaliste de type *HelloWorld* dans le cadre des interfaces Swing. On implantera dans une classe `ScoreFrame` l'interface graphique suivante, où le bouton + incrémente le score et le bouton − le décrémente.

Figure 4: Cette vue représente une fenêtre principale.

7.1.1 Gestion de la fenêtre : ScoreFrame

⇒ **Quels sont les traitements à réaliser pour créer et afficher une fenêtre ?**

⇒ **Donnez le code pour la définition d'une classe `ScoreFrame`** dont le constructeur crée une fenêtre au contenu vide, mais ayant un titre, une taille, et qui se ferme quand on clique sur la croix dans l'angle.

⇒ Donnez le code d'un `main` qui affiche cette fenêtre.

7.1.2 Gestion du contenu : ScorePanel

On se donne la classe `Score`, portant un entier et les opérations `incremente()`, `decremente()`, `getScore() : int`, `setScore(int)` et un constructeur.

Score.java

```
package pobj.scoreswing;

public class Score {
  private int val;
  public Score(int v){ val = v; }
  public void incremente() { val++; }
  public void decremente() { val--; }
  public int getScore() { return val; }
  public void setScore(int v) { val = v; }
}
```

Cette classe représente un modèle par rapport à la vue que l'on est en train de construire. On choisit ici de séparer le code correspondant au contenu de la fenêtre de la fenêtre elle-même (`ScoreFrame`).

⇒ **Créez une classe `ScorePanel`**, dérivant de `JPanel`. Son constructeur prend en argument une instance de `Score` dont elle affiche l'état.

⇒ **Mettez à jour le code de `ScoreFrame`** pour que son contenu soit une instance de `ScorePanel`.

⇒ **De quelle(s) classe(s) de composants `Swing` a-t-on besoin pour le contenu du `ScorePanel` ?**

⇒ **Quel gestionnaire de géométrie proposez-vous pour agencer ces composants ?**

⇒ **Faites un diagramme** montrant la structure du `ScorePanel`.

⇒ Donnez le code du constructeur de `ScorePanel` qui réalise cette mise en place de l'aspect graphique. On supposera que le `Score`, le `JTextField` et les deux `JButton` sont stockés comme attributs du `ScorePanel`. On positionnera le texte pour refléter le score courant.

7.1.3 Gestion des événements sur les boutons

⇒ Quel(s) événement(s) doit-on gérer pour que le clic sur le bouton "+" incrémente le score et le clic sur le bouton "−" le décrémente?

⇒ Donnez le code de traitement à effectuer, pour chaque bouton.

⇒ **Quelle interface doit-on implémenter, ce qui correspond à quelle(s) méthode(s)?**

⇒ **Discutez les propositions de solution suivantes** :

- Une solution dans laquelle c'est la classe `ScorePanel` elle-même qui est déclarée `actionListener` des boutons
- Une solution dans laquelle la classe `ScorePanel` contient une classe interne `BoutonListener` qui réalise les actions associées aux boutons
- Une solution dans laquelle on crée deux classes externes `BoutonListenerPlus`, `BoutonListenerMoins` qui sont les listeners respectifs des boutons
- Une solution dans laquelle on crée à la volée des classes anonymes de `Listener` attachées à chaque bouton

⇒ **Donnez un code complet pour la solution que vous préférez**.

7.1.4 Réutilisation de code

⇒ **Comment réaliser une fenêtre qui affiche deux scores (équipe A et équipe B) et leurs boutons + et − respectifs ?**

7.1.5 Gestion des événements sur le modèle

On suppose à présent que le score peut potentiellement évoluer à cause d'autres facteurs externes (par exemple le score décroit toutes les 5 secondes).

On souhaite que l'interface graphique reflète toujours l'état du modèle.

⇒ **Quel *design pattern* (DP) permet de traiter ce problème ?**

⇒ **Quels rôles occupent le `ScorePanel` et le `Score` dans ce DP ?**

⇒ **Implémentez ce fonctionnement**, en appui sur la classe `Observable` et l'interface `Observer` de `java.util`.