IN201 : 7 – GUI (chat)

Author : Christophe Garion <garion@isae.fr>
Public : SUPAERO 2A
Date :

Résumé

Le but de ce TP est de construire une application en utilisant le patron de conception MVC et d'utiliser quelques composants de Swing.

1 Objectifs

Les objectifs du TP sont les suivants :
- comprendre le MVC en construisant une application ;
- manipuler quelques composants simples de Swing.

2 Présentation du problème

On souhaite réaliser une interface graphique pour une application de *chat*[1]. On rappelle qu'un *chat* permet à des utilisateurs de s'envoyer des messages de façon informelle. L'application à développer devra posséder deux vues graphiques qui permettront d'envoyer une chaîne de caractères et qui afficheront sur une zone de texte les messages reçus par l'application ainsi que les noms d'utilisateurs correspondant à chaque message.

3 Définition du modèle

Le modèle sera une classe `Chat` qui stocke un ensemble d'entrées, modélisées par des instances de `EntreeChat`. Chaque entrée est composée de deux chaînes de caractères :
- le nom de l'auteur de l'entrée
- le texte de l'entrée

La classe `Chat` étend la classe `java.util.Observable` afin de pouvoir être observée par les vues.
Un diagramme de classe représentant `Chat` et `EntreeChat` vous est proposé sur la figure 1.

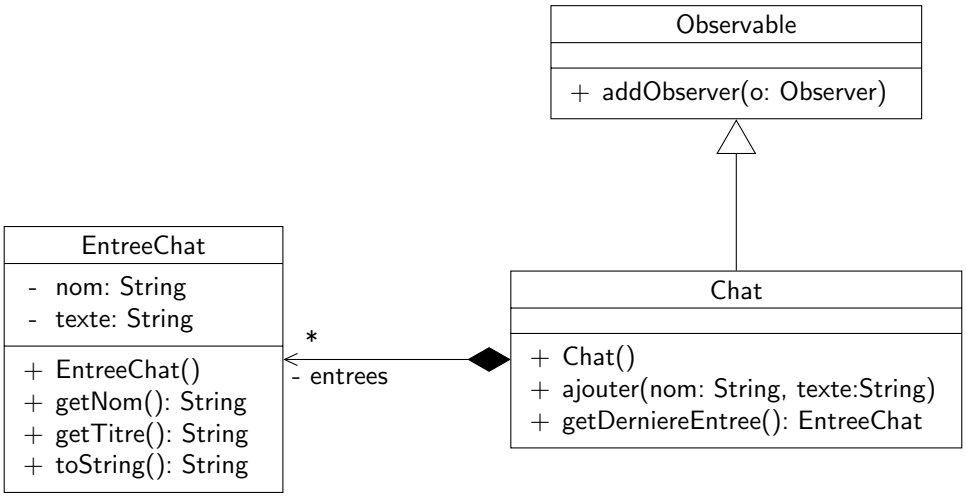

FIGURE 1 – Les classes `Chat` et `EntreeChat`

4 Définition des vues

Les vues seront des fenêtres graphiques composées des éléments suivants :
- un nom d'utilisateur ;

1. Causette en francais, clavardage en québécois...

- un bouton permettant d'envoyer le texte à l'application ;
- une zone de texte `JTextField` pour écrire le texte à envoyer ;
- une zone de texte pour écrire les messages provenant de l'application. Ce sera un objet de type `JTextArea`. Regarder la documentation javadoc de `JTextArea` pour trouver la méthode permettant d'ajouter du texte à la zone de texte.

Un exemple minimal d'IHM est proposé sur la figure 2.

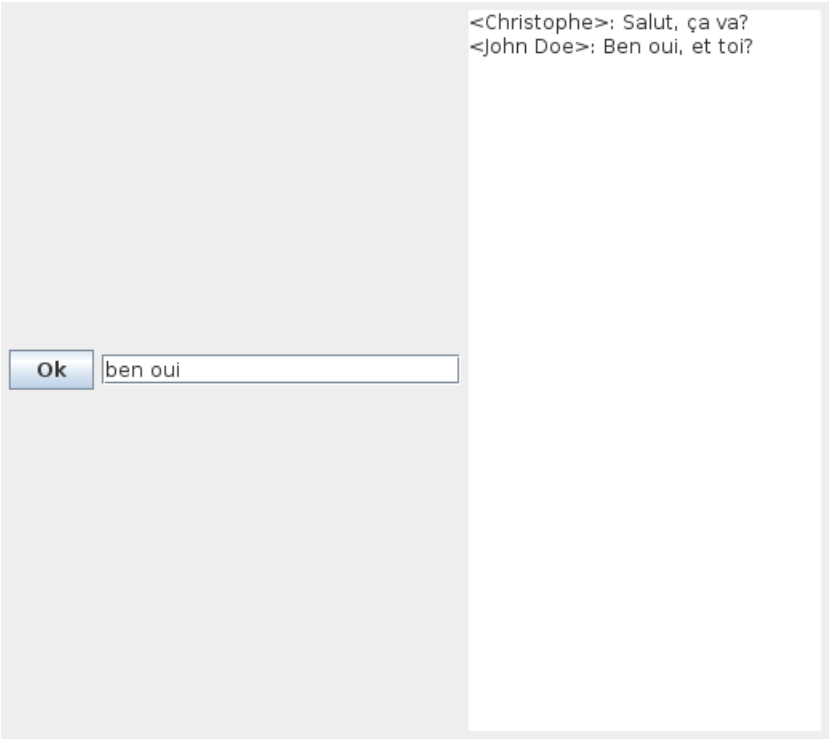

FIGURE 2 – Une vue possible pour le *chat*

5 Conception de l'application

Proposer un diagramme de classes simples modélisant l'application. On s'appliquera à bien choisir la façon dont la vue et le modèle vont communiquer.

6 Implantation

Pour réaliser le TP, vous allez devoir créer un projet Java sous Eclipse en utilisant votre dépôt Subversion. Si vous avez configuré votre dépôt pour qu'il soit disponible dans la vue *SVN Repository Exploring* d'Eclipse, vous créerez votre projet en faisant un *checkout* **à partir du répertoire TP7** de votre dépôt en cliquant droit dessus. N'oubliez pas de configurer votre *build path* pour pouvoir utiliser les éventuelles archives JAR placées dans le répertoire `lib`.

Il faut maintenant implanter les vues et les contrôleurs associés. La classe `Chat` vous est fournie dans l'archive disponible sous le répertoire `lib` de votre dépôt. Vous devez utiliser les classes et interfaces fournies par le JDK implantant le patron de conception Observateur, i.e. `java.util.Observable` et `java.util.Observer`. Cette classe et cette interface fonctionnent à peu près comme celles que nous avons développées, sauf que l'on peut passer des paramètres aux méthodes correspondant à `miseAJour`, `avertir` etc. Référez-vous à la javadoc de ces classes pour plus de détails.

1. un premier observateur pour `Chat` vous est fourni dans l'archive. Il s'agit de la classe `ChatFileLogger` qui écrit les conversations du Chat dans un fichier texte (cf. javadoc). Écrire un programme utilisant cette classe et vérifier que les entrées ajoutées dans le *chat* sont bien écrites dans le fichier choisi.
2. écrire la classe `VueChat` (on pourra utiliser d'autres *layout managers* que celui vu en cours si nécessaire). Pour cela :
   (a) quels sont les paramètres du constructeur de `VueChat` ? Implanter le constructeur en conséquence, en prévoyant que la classe est une sous classe de `JFrame` (titre de la fenêtre à mettre en place etc).
   (b) la méthode `initComponents` créant les composants et les plaçant via un `FlowLayout` vous est donnée. Modifier le constructeur de `VueChat` afin de l'appeler. Modifier le programme créé à la question 1 en créant deux vues sur le *chat* avant d'ajouter des entrées sur le *chat*. Que se passe-t-il ?
   (c) faire en sorte que `VueChat` soit un observateur de `Chat`. Que se passe-t-il lorsque l'on exécute le programme précédent ?
3. implanter le contrôleur permettant de lier la vue et le modèle. Créer un programme de test créant un *chat* et deux vues sur le *chat* et vérifier que tout fonctionne.
4. utiliser un autre *layout manager* pour placer le bouton et la zone de texte en dessus de la fenêtre de *chat*. On pourra consulter [2] pour trouver le *layout manager* à choisir.
5. (facultatif) ajouter un autre bouton et une zone de texte pour pouvoir changer le nom de l'utilisateur. On pourra utiliser une classe interne ou une classe anonyme (cf. slides et votre vacataire).
6. (facultatif) utiliser l'extension WindowBuilder d'Eclipse (cf. page Logiciels du site web pour l'installer sur votre machine personnelle) pour construire une nouvelle vue pour votre application. Le manuel utilisateur de WindowBuiler est disponible sur [1] et un petit tutoriel vous est proposé sur le site du cours.
7. (facultatif) tester l'application avec un *framework* de test unitaire comme Google Fest [3].

Références

[1] Google. *WindowBuilder User Guide*. 2013. URL : https://developers.google.com/java-dev-tools/wbpro/.

[2] Oracle. *Creating a GUI With JFC/Swing*. 2013. URL : http://docs.oracle.com/javase/tutorial/uiswing/.

[3] The Fest Developers. *Google Fest – Fixtures for Easy Software Testing*. 2013. URL : http://code.google.com/p/fest/.

License CC BY-NC-SA 3.0