TP5 Interface graphique

Développement Logiciel (L2-S4)

Vendredi 21 février 2014

Le but de ce TP est de vous familiariser avec différents aspects des interfaces graphiques en Java.

Préambule Récupérez les sources à l'adresse suivante :
`http://www.perso.limsi.fr/sadoun/TP5.zip`
Prenez le temps de lire toutes les explications, de vous référer à vos polycopiés de cours et au wiki (`https://www.lri.fr/~anab/teaching/DevLog`).

Très important : Ce TP est noté. Avant de partir, vous serez noté par votre chargé de TP sur les critères suivants : Questions réussies (questions qui fonctionnent et qui respectent le sujet du TP) ; Utilisation des concepts du cours (visibilité des variables et méthodes, héritage, exceptions, Threads, entrées sorties ...) ; Clarté du code (Lisibilité, commentaires).

Vous avez également la possibilité de terminer le TP chez vous si ce n'est pas le cas durant la séance. Dans ce cas, vous devez envoyer les sources dans une archive NomPrenomTP4.zip à votre chargé de TP. Le titre du mail doit être [DevLog] TP5 de Nom Prénom

On se propose de créer une interface graphique permettant la création et la gestion des personnages créés lors du TP3. Pour cela vos pouvez vous basez sur vos sources et sur celles proposées à l'adresse : *https ://www.lri.fr/ anab/teaching/DevLog/*.

La fenêtre principale *GestionnaireUI* de notre application hérite d'une *JFrame*. Cette fenêtre permet la création, le chargement et la sauvegarde d'un gestionnaire grâce a un menu. Cette fenêtre permet aussi l'affichage de la liste de personnages du gestionnaire, et l'ajout d'un personnage au gestionnaire.

Un double-click sur un personnage du gestionnaire déclenche l'ouverture d'une nouvelle fenêtre *PersonnageUI* héritant de *JDialog*. Cette dernière fenêtre permet d'afficher une table des caractéristiques du personnage (type, nom, force, vitesse, arme ...). Elle permet aussi de filtrer les personnages selon leur type, force, arme, ...

Vous utiliserez principalement l'*API Swing*[1] lors de vos développements.

Exercice 1 : Création du gestionnaire

Pour commencer créer une classe *Gestionnaire.gui.GestionnaireUI* héritant de *JFrame*.

1. implémentez une méthode *init*() qui sera appelée par le constructeur de *GestionnaireUI* pour initialiser la *JFrame* (titre, taille, position centré au lancement, visibilité ...).
2. implémentez une méthode *initMenu*() qui sera appelée par le constructeur. Cette méthode crée un menu nommé *Gestionnaire* avec les sous-menus suivant :
   – *Nouveau* : permet la création d'un nouveau gestionnaire. Il remplace tout simplement l'ancien gestionnaire par un nouveau.

1. Vous trouverez des tutoriels très complets pour les composants Swing à cette adresse : http ://docs.oracle.com/javase/tutorial/uiswing/components/index.html

- *Charger* : permet le chargement d'un gestionnaire existant. Demande a l'utilisateur de choisir un fichier de type '*.save' grâce au composant *JFileChooser* avant de le charger grâce a la méthode *EntreesSorties.chargerFichier*(File f). Vous pouvez utiliser le fichier 'personnage.save' qui vous est fourni. Contentez-vous pour l'instant d'afficher le résultat d'appel de la méthode *toString*() sur ce gestionnaire pour tester le chargement.
- *Sauvegarder* : permet la sauvegarde du gestionnaire courant. Il demande a l'utilisateur de choisir un fichier pour sauvegarder le gestionnaire courant (*EntreesSorties. sauvegarderFichier* (Gestionnaire g, File f)). Veillez toutefois à surcharger la méthode **public void** *approveSelection*() de *JFileChooser* afin de demander une confirmation a l'utilisateur dans le cas où il sélectionne un fichier existant, afin de ne pas écraser le fichier sélectionné par mégarde. Une méthode simple pour demander une confirmation consiste a utiliser la méthode **public static int** *JOptionPane.showConfirmDialog*(. . .).

```
int result = JOptionPane.showConfirmDialog ( this , " The file exists , overwrite ? " ,"
Existing file " , JOptionPane.YES_NO_CANCEL_OPTION ) ;
if ( result == JOptionPane . YES_OPTION )  ...
else  ...
```

Exercice 2 : Affichage des personnages

Nous nous occuperons maintenant de l'affichage des personnages du gestionnaire. Pour cela, nous utilisons une *JList*. Afin de pouvoir mettre à jour le contenu de cette *JList*, nous utilisons une instance de *DefaultListModel*. Tous les changements de cette instance, sont répercutés sur la *Jlist* automatiquement.

1. implémentez une méthode *initListPersonnages*() appelée par le constructeur. Cette méthode crée une *JList* puis la rajoute au contenu de *GestionnaireUI*. Pensez au cas où la liste des personnages serait trop grandes (*JScrollPanes*).
2. implémentez ensuite la méthode *initPersonnageIntoList*() qui réinitialise le contenu de la *JList* avant d'y ajouter tous les personnages du gestionnaire. A ce stade, vous devriez pouvoir charger le fichier 'personnage.save' et voir les différents personnages s'afficher.
3. rajouter maintenant un bouton 'ajouter personnage' permettant de charger un personnage depuis un fichier '.csv' et de l'ajouter au gestionnaire. Vous pouvez utiliser un *BoxLayout* pour forcer l'alignement vertical de la *JList* et du bouton à ajouter personnage. L'utilisateur doit pouvoir fournir le nom du personnage, puis le fichier a charger. Vous pouvez pour cela créer un *JDialog* demandant à l'utilisateur ces informations. Notez que si vous déclarez ce *JDialog* en tant qu'*Inner Class*, vous pouvez accéder simplement aux champs de *GestionnaireUI*.

   Notez aussi que la modification seule de l'instance de *DefaultListModel* ne suffit pas à mettre a jour le gestionnaire ...

   Pour vous faciliter la tâche et vous permettre de vous familiariser avec les *Layout managers*, nous vous proposons de créer un premier conteneur *JPanel* dont le layout manager est un *GridLayout*(2,2) (deux lignes, deux colonnes). Ce conteneur *JPanel* contient un *JLabel* "nom personnage", un *JTextField*, un *JLabel* "Fichier :" et enfin un *JButton* déclenchant l'ouverture d'un *JFileChooser*. Ce conteneur *JPanel* ainsi qu'un bouton Valider sont ensuite alignés sur le conteneur principal du JDialog en utilisant un *BoxLayout* vertical.
4. Vérifiez que l'utilisateur n'a pas rentré un nom de personnage vide, et a bien choisi un fichier. Sinon, affichez un message d'erreur (*JOptionPane.showMessageDialog*).
5. Ajouter un *Listener* qui permet de déclencher l'affichage du résultat de l'appel de *toString*() sur une instance de personnage lorsque l'utilisateur double-clique sur ce personnage.
6. Rajouter un petit logo (un fichier 'monster.png' vous est fourni) au dessus de la *JList*. Pour cela, créez une classe héritant de *JComponent* puis veillez à surcharger la méthode

paintComponent(Graphics g) afin de dessiner l'image. Veillez à ce que cette image reste centrée dans le cas où la fenêtre est redimensionnée.

Exercice 3

Nous allons maintenant passer a l'affichage des caractéristiques d'un personnage. Pour cela, nous implémenterons une classe *CarateristiqueUI* héritant de *JDialog*, affichant une *JTable* de caractéristiques. Cette fenêtre est ouverte lorsque l'utilisateur double-clique sur un personnage dans *GestionnaireUI*.

1. créez la classe *Gestionnaire.gui.CarateristiqueUI* ainsi qu'une méthode *init*() appelée depuis le constructeur pour initialiser la fenêtre.
2. implémentez une méthode *initListCaractéristique*() appelée depuis le constructeur permettant de créer une *JTable* et de l'ajouter au contenu de la fenêtre. Puis implémentez une méthode *initPersonnageIntoTable*() pour initialiser les données de la *JTable* en y ajoutant les informations concernant les caractéristiques du personnage (autant de ligne que de caractéristiques : type, nom ...).
3. Sur chaque ligne ajouter un colonne *sélectionner* qui s'affiche comme une *checkbox*. Elle permettra de spécifier les caractéristiques devant être mise à jour.
4. Faites en sorte de mettre a jour les caractéristiques du personnage lorsque la valeur de la colonne *sélectionner* est cochée. Vous pouvez utiliser pour cela un *TableModelListener*.
5. Ajouter un *Checkbox* à la fenêtre *CarateristiqueUI* permettant de cacher toutes les caractéristiques du personnage déjà sélectionner (*TableRowSorter* et *setRowFilter*).

Exercice 4 : Recherche dans la base

Reprenez l'énoncé de l'exercice 5 (ci-dessous) du TP 3 et adaptez le au mode graphique.

Il s'agit de créer une méthode permettant de chercher certains personnages dans la base en utilisant un système de dialogue. Un tel système peut fonctionner de la manière suivante :

Quel type de personnage souhaitez vous (Hero ou Monstre) ?
> Hero
Quel critère voulez vous utiliser (âge, nom arme) ?
> nom
Entrez un nom (ou une partie du nom).
> John
Voici les personnages correspondant à votre recherche :
- John Snow age : 24 points_de_vie : 40 force : 80 arme : épée
- John Wayne age : 55 points_de_vie : 0 force : 50 arme :pistolet