Projet Wintel : application graphique en Java

RAPPEL : API JAVA 1.6 DISPONIBLE EN HTTP://DOWNLOAD.ORACLE.COM/JAVASE/6/DOCS/API

1. PRESENTATION DE L'APPLICATION GRAPHIQUE

L'application *Wintel* simule, de façon très simplifiée, le répertoire téléphonique d'un agenda électronique.

La fenêtre principale (*Wintel*) se compose d'une liste de correspondants (Figure 1) où chaque correspondant peut être sélectionné par simple clic dans la liste.

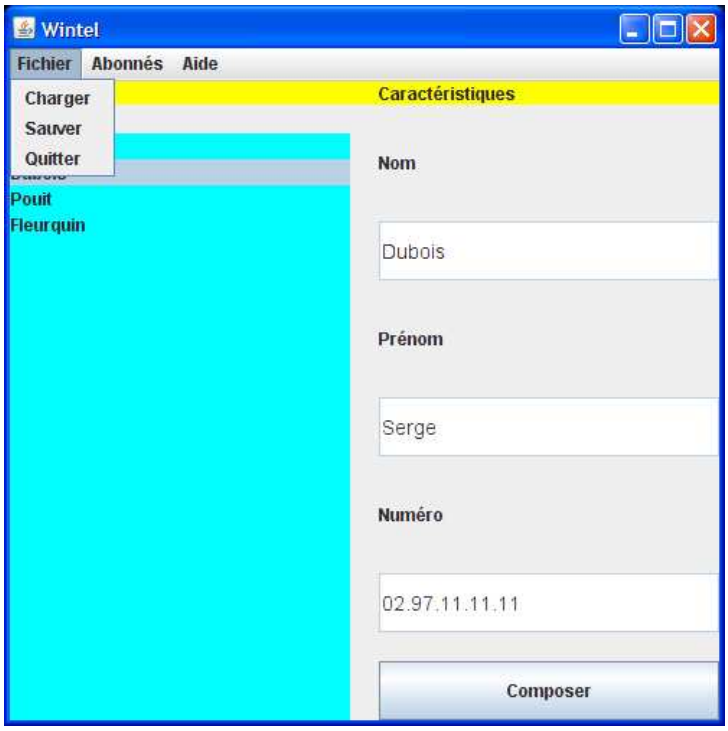


Figure 1 : la fenêtre principale

Figure 2 : la fenêtre « Ajouter un correspondant »

La modification (ajout, suppression, modification) de la liste des correspondants se fait par l'intermédiaire de trois autres fenêtres (dont une seule est représentée à la Figure 2). Les données sont enregistrées sur le disque. Elles sont chargées et sauvegardées par simple clic dans le menu « Fichier » de la fenêtre principale (voir Figure 1).

Toute l'application est basée sur la bibliothèque graphique *swing* de Java.

La Figure 3 montre le diagramme de classes de l'application. Notez l'existence de 2 paquetages : *ihm* et *datas* à respecter impérativement lors du codage.

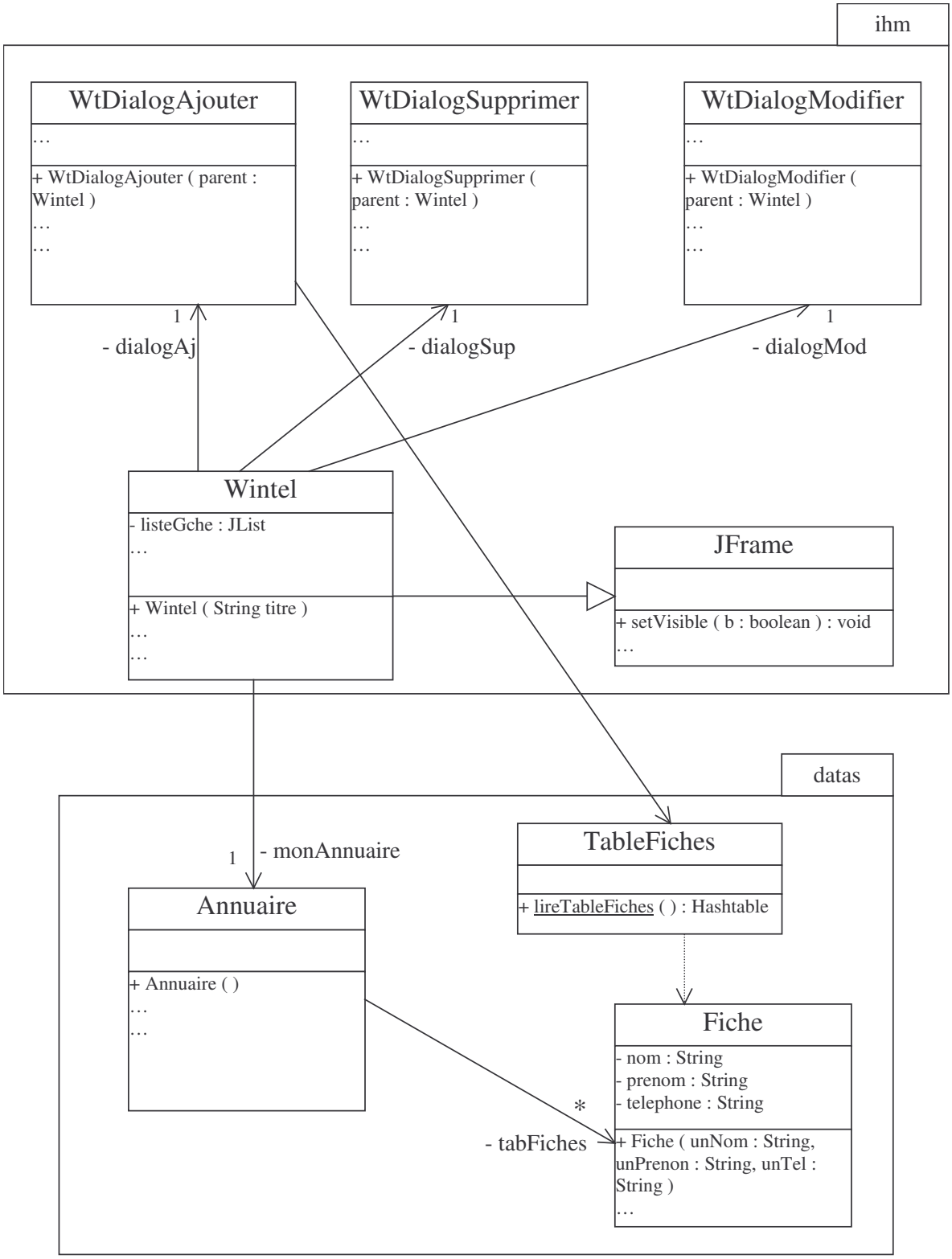


Figure 3 : diagramme de classes de l'application

La classe principale *Wintel* hérite de *JFrame* et contient :

- En attributs (privés !) : TOUS les éléments graphiques qui apparaissent dans la fenêtre principale (voir Figure 4).

- Une méthode privée *creerInterface ()* qui met en place TOUS les éléments graphiques dans la fenêtre principale.
- Une méthode privée *attacherReactions ()* qui attache aux composants graphiques adéquats des écouteurs d'événements utilisateurs.
- Un constructeur publique *Wintel()* qui aura essentiellement le code suivant :

```
public Wintel() {
     super ( "Wintel" );        // appel constructeur de JFrame
     this.creerInterface();     // mise en place du décor
     this.attacherReactions(); // écouteurs des événements utilisateurs
     this.init();               // création de l'annuaire (+ des 3 WtDialog*)
     this.setSize( 500, 500 ); // dimensionnement de la fenêtre
     this.setVisible( true );  // rendre la fenêtre visible
     this.setDefaultCloseOperation ( EXIT_ON_CLOSE );  // clic sur la croix
}
```

- Un lanceur (inclut dans la classe *Wintel*) qui crée simplement une instance de *Wintel*. La création de l'instance affichera forcément la fenêtre à l'écran.

2. MISE EN PLACE DU DECOR DE L'APPLICATION WINTEL

Ecrire dans la classe *Wintel* le constructeur (voir ci-dessus) et la méthode *creerInterface ()* qui met en place le décor décrit graphiquement à la Figure 4. Déclarer TOUS les composants graphiques en attributs privés.

TESTER régulièrement cette méthode pour visualiser l'aspect graphique de votre code à l'écran.

1. *Wintel* hérite de *JFrame*.
2. Le gestionnaire de placement de *Wintel* est un *BorderLayout* : *setLayout (new BorderLayout())*.
3. Au nord, on place un widget *JPanel* (simple panneau, « panelNord ») caractérisé par un gestionnaire de placement *GridLayout (1, 2)* de 1 ligne et 2 colonnes. A l'intérieur, 2 widgets *JLabel* (simple affichage d'un texte en lecture) pour les titres « Abonnés » et « Caractéristiques ».
4. Au centre, on place aussi un widget *JPanel* « panelCentre » caractérisé par un gestionnaire de placement *GridLayout (1, 2, 40, 0)* de 1 ligne, 2 colonnes et 40 pixels de distance entre les colonnes. A l'intérieur, colonne de gauche, on ajoute un widget *JList* et à droite un troisième *JPanel* « panelCaract » avec un gestionnaire de placement *GridLayout (0, 1)* particulier (en effet, on se serait attendu à un gestionnaire *GridLayout (7, 1)* mais ce n'est pas le cas, voir la Javadoc de *GridLayout*).
5. A l'intérieur de « panelCaract », on place successivement : 1 *JLabel* « Nom », un 1er *JTextField*, 1 *JLabel* « Prénom », un 2eme *JTextField*, 1 *JLabel* « Numéro », un 3eme *JTextField*, 1 *JButton* « Composer ».
6. Finalement, on place la barre de menu (widget *JMenuBar*) en procédant comme suit :
   - créer une barre de menu : *myBar = new JMenuBar();*
   - créer les objets *JMenu* : *menuFichier = new JMenu("Fichier"); menuAbonnes = new JMenu("Abonnés"); menuAide = new JMenu("Aide");*

- les ajouter à la barre de menus : *myBar.add(menuFichier); myBar.add(…);* etc.
- créer des objets *JMenuItem* : *itemCharger = new JMenuItem("Charger"); …("Sauver"); …("Quitter"); itemAjouter = new JMenuItem("Ajouter"); …("Modifier"); …("Supprimer"); itemAide = new JMenuItem("Aide");*
- les ajouter à l'objet *JMenu* correspondant : *menuFichier.add(itemQuitter); menuFichier.add(…); menuFichier.add(…); menuAbonnes.add(itemAjouter); menuAbonnes.add(…); menuAbonnes.add(…); menuAide. add(itemAide);*
- finalement, associer la barre de menu (*myBar*) à la fenêtre de l'application (Wintel) : *setJMenuBar(myBar);* (le placement et la forme de la barre de menu sont automatiques).

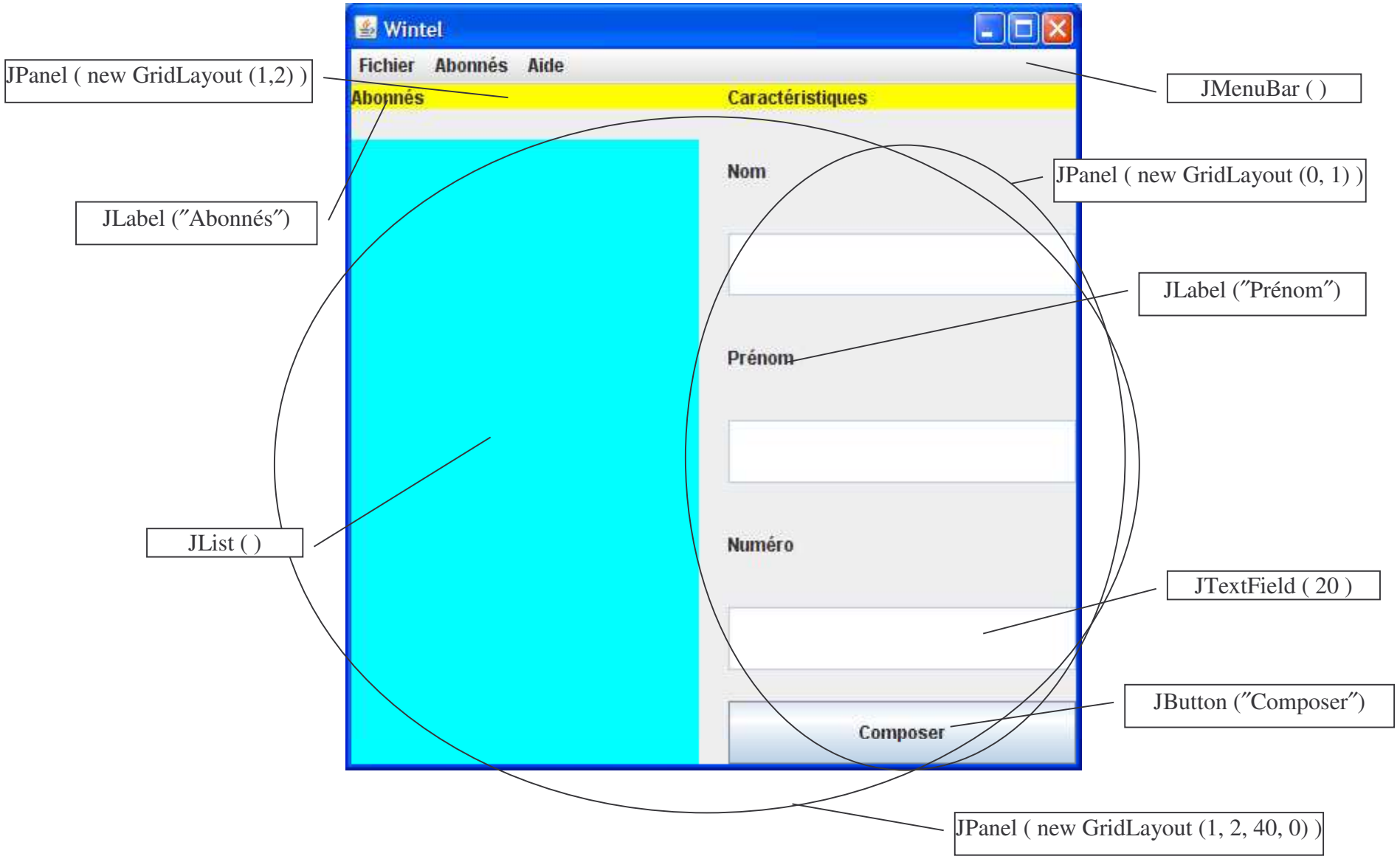


Figure 4 : les différents “widgets” et gestionnaires de placement mis en œuvre

3. MISE EN PLACE DE L'ANNUAIRE

Le stockage sur le disque (écriture et lecture) des coordonnées téléphoniques de chaque correspondant est géré par deux classes : *Fiche* et *Annuaire*. Ces classes ont déjà été développées lors d'une séance précédente (I/O des objets), elles appartiennent au paquetage *datas*. On se rappelle qu'une *Fiche* mémorise les caractéristiques d'un seul correspondant alors que l'annuaire mémorise un tableau (de type *java.util.Hashtable<K,V>*) d'objets *Fiche*. Les types génériques de ce tableau correspondent d'une part au type de la clé de recherche (K = type *String*) et, d'autre part, au type des objets stockés dans la table (V = type *Fiche*). La classe *Annuaire* offre tous les services nécessaires à l'ajout, la suppression, la recherche des fiches dans le tableau mais également l'écriture et la lecture de l'annuaire sur le disque (fichier « annuaire.out » par défaut).

L'annuaire dans Wintel

Un annuaire est représenté dans l'interface graphique à travers la *JList* qui affiche la liste de tous les noms des personnes de l'annuaire.

Le fonctionnement et l'utilisation de l'annuaire se font comme suit :

- lors du lancement de l'application, l'annuaire est créé mais il est vide. Pour afficher l'annuaire dans la *JList* à partir du fichier « annuaire.out », cliquer sur l'item « Charger » du menu « Fichier »,
- lorsque l'on clique sur un nom de la *JList*, les caractéristiques de la personne apparaissent dans les champs textuels de droite,
- une fois affiché dans l'interface graphique, l'annuaire peut être modifier de 3 façons : en ajoutant une nouvelle personne (item « Ajouter » du menu « Abonnés »), en modifiant une personne existante (item « Modifier » du menu « Abonnés »), en supprimant une personne existante (item « Supprimer » du menu « Abonnés »).
- pour sauvegarder l'annuaire mis à jour par suppression, modification ou ajout, dans le fichier « annuaire.out », cliquer sur l'item « Sauver » du menu « Fichier ».

Travail à réaliser

1. Définir un attribut privé *monAnnuaire* de type *Annuaire* dans la classe *Wintel* et le créer dans la méthode *private void init()* de la classe *Wintel*.
2. Ecrire la méthode publique *public void ajouterAbonne (String nom, String prenom, String numTel)* dans la classe *Wintel*. Elle servira surtout dans une section ultérieure pour ajouter un nouveau contact dans l'annuaire à partir d'une boite de dialogue (voir la classe *WtDialogAjouter*).

   Cette méthode doit :

   2.1. Créer un objet *Fiche* à partir des arguments « nom », « prenom », « numTel »,

   2.2. Ajouter la fiche dans l'annuaire (voir les méthodes de la classe *Annuaire*),

   2.3. Ajouter la clé de la fiche (= le nom du correspondant) dans le widget *JList* (colonne de gauche) de *Wintel* (voir ci-après pour les détails techniques).

   Cette méthode devra être testée (à partir de l'objet *Wintel* créé dans le lanceur).

3. Ecrire les réactions aux différents clics (voir la section suivante pour les détails techniques) :

   3.1. Réaction au clic sur un nom de la liste d'abonnés : les caractéristiques de l'abonné sélectionné (nom, prénom, téléphone) doivent s'afficher dans les zones textuelles correspondantes (colonne de droite de *Wintel*).

   3.2. Réaction au clic sur l'item « Sauver » du menu « Fichier » : l'annuaire téléphonique est sauvegardé sur le disque (fichier « annuaire.out » par défaut).

   3.3. Réaction au clic sur l'item « Charger » du menu « Fichier » : lire l'annuaire téléphonique à partir du disque et afficher chacun des noms (= les clés de l'annuaire) dans le widget *JList*.

La réaction au clic sur un nom de la *JList*

Avant toute chose, une *JList* ne peut fonctionner correctement que si on lui associe explicitement une collection (un tableau) qui mémorise en permanence les éléments affichés dans la *JList*. C'est uniquement par l'intermédiaire de ce tableau (appelé « modèle de liste ») que les opérations d'ajout et de suppression des éléments sont réalisées : l'affichage des éléments dans la *JList* se synchronise automatiquement avec le contenu du tableau.

Dans notre cas, on donne à la *JList* un « modèle de liste » par défaut : *maListe.setModel(new DefaultListModel())*. Toutes les opérations possibles sur le tableau et donc la *JList* (ajout, suppression etc.) sont donc décrites dans la classe *DefaultListModel* (le type de collection associée à la liste).

Pour récupérer le tableau associé à la *JList* (et pour ensuite pouvoir modifier ce tableau et donc l'affichage de la *JList*), utiliser la méthode *getModel()* de la classe *JList* (ne pas oublier le transtypage : *DefaultListModel monTab = (DefaultListModel) maListe.getModel()*). En particulier pour la méthode *public void ajouterAbonne (String nom, String prenom, String numTel)* à réaliser ci-dessus, il faudra utiliser la méthode *void addElement (Object obj)* de l'objet *monTab* (type *DefaultListModel*) pour ajouter visuellement dans la *JList* le nom de la personne.

La réaction au clic souris sur un nom de la *JList* s'écrira dans une classe séparée *EcouteurListeGche*. Procéder de la manière suivante :

1. Mettre la *JList* à l'écoute de l'événement souris *MouseEvent* : *maListe.addMouseListener (new EcouteurListeGche(this))*. Le paramètre *this* est important et représente ici la référence sur l'objet *Wintel*.

2. Ecrire la classe *EcouteurListeGche* (paquetage *ihm*) qui doit définir la méthode qui sera appelée en réaction au clic souris dans la liste (le code de la classe est donné ci-dessous mais il faut le comprendre ! Attention, les accesseurs de la classe *Wintel* doivent avoir la même signature que ceux utilisés dans le code ci-dessous) :

```
class EcouteurListeGche extends MouseAdapter {
     // La référence sur Wintel
     private Wintel theWin;

     // Constructeur : lui donner la référence sur Wintel
     // sinon l'accès aux méthodes de Wintel est impossible.
     public EcouteurListeGche ( Wintel monWin ) {
        this.theWin = monWin;
     }
```

```
        // Méthode de REACTION au clic souris sur un élément de la JList
        public void mouseClicked ( MouseEvent e ) {
           String cle;
           // Accès à la JList
           JList maListe = theWin.getListeGche();  // getListeGche accesseur de Wintel
           // Récupération de l'endroit (index) où l'utilisateur a cliqué
           int index = maListe.locationToIndex(e.getPoint());

           // Récupération du tableau qui mémorise les éléments de la JList
           DefaultListModel tab = (DefaultListModel)( maListe.getModel());
           // L'index correspond précisément à la case du tableau contenant la donnée
           // sélectionnée par l'utilisateur avec la souris
           cle = (String)tab.get(index);
           // La clé (nom de la personne) permet de récupérer la fiche de la personne
           // dans l'annuaire
           Annuaire monA = theWin.getAnnuaire();   // getAnnuaire accesseur de Wintel
           Fiche laFiche = monA.consulter(cle);

           // Affichage des informations de la fiche dans les 3 champs textuels
           // de droite
           JTextField nom = theWin.getFieldNom();          // accesseur de Wintel
           nom.setText(laFiche.getNom());
           JTextField prenom = theWin.getFieldPrenom();    // accesseur de Wintel
           prenom.setText(laFiche.getPrenom());
           JTextField num = theWin.getFieldNumero();       // accesseur de Wintel
           num.setText(laFiche.getTelephone());
        }
    }
```

Réaction aux clics sur les items « Sauver » et « Charger » du menu « Fichier »

Dans les 2 cas, la réaction s'écrira dans une classe séparée *EcouteurItemSauver* et *EcouteurItemCharger*. Chacun des items doit être à l'écoute de l'événement *ActionEvent* : *itemSauver.addActionListener (new EcouteurItemSauver(this))* et *itemCharger.addActionListener (new EcouteurItemCharger(this)).*

De façon similaire à ce qui a été fait ci-dessus pour la liste, écrire le code des classes *EcouteurItemSauver* et *EcouteurItemCharger* :

- Elles doivent implémenter l'interface *ActionListener*.
- Elles doivent avoir un constructeur qui prend en paramètre la référence sur *Wintel*.
- Elles doivent donner le code de la méthode de réaction *public void actionPerformed (ActionEvent e)* :
  - Réaction au clic sur *ItemSauver* : simplement appeler la méthode *sauver()* de l'objet de type *Annuaire* (objet obtenu avec l'accesseur *getAnnuaire()* de la classe *Wintel*).

- Réaction au clic sur *ItemCharger* : simplement appeler la méthode publique *void chargerEtAfficherAnnuaire()* de la classe *Wintel*. Cette méthode de la classe *Wintel* appelle d'abord la méthode statique *charger()* de la classe *Annuaire* pour obtenir un nouvel objet *Annuaire* créé à partir du fichier. Ensuite, il faut forcer l'affichage de l'annuaire dans la *JList* par remplissage du tableau de type *DefaultListModel* avec les noms (c'est-à-dire les clés) de l'annuaire.

4. LES BOITES DE DIALOGUE

En réaction au clic de l'utilisateur sur les items « Ajouter », « Modifier », « Supprimer » du menu « Abonnés », il faut afficher la fenêtre de dialogue correspondante pour permettre l'interaction.

Chaque fenêtre de dialogue (Ajouter, Modifier, Supprimer) est un objet instance d'une classe *WtDialog** qui hérite de la classe *JDialog* de l'API. Une fenêtre de dialogue s'affichera toujours devant la fenêtre principale (ici *Wintel*) qui est appelée fenêtre parente. Il ne vous reste plus qu'à coder les 3 classes : *WtDialogAjouter*, *WtDialogSupprimer* et *WtDialogModifier* qui appartiennent toutes les 3 au paquetage *ihm*. L'instanciation des 3 fenêtres *WtDialog** se fait dans la méthode *init()* de la classe *Wintel*.

La classe WtDialogAjouter

Ecrire le code de la classe *WtDialogAjouter* dans un premier temps. Elle hérite de *JDialog*. L'aspect graphique de cette fenêtre doit ressembler à celui de la Figure 5.

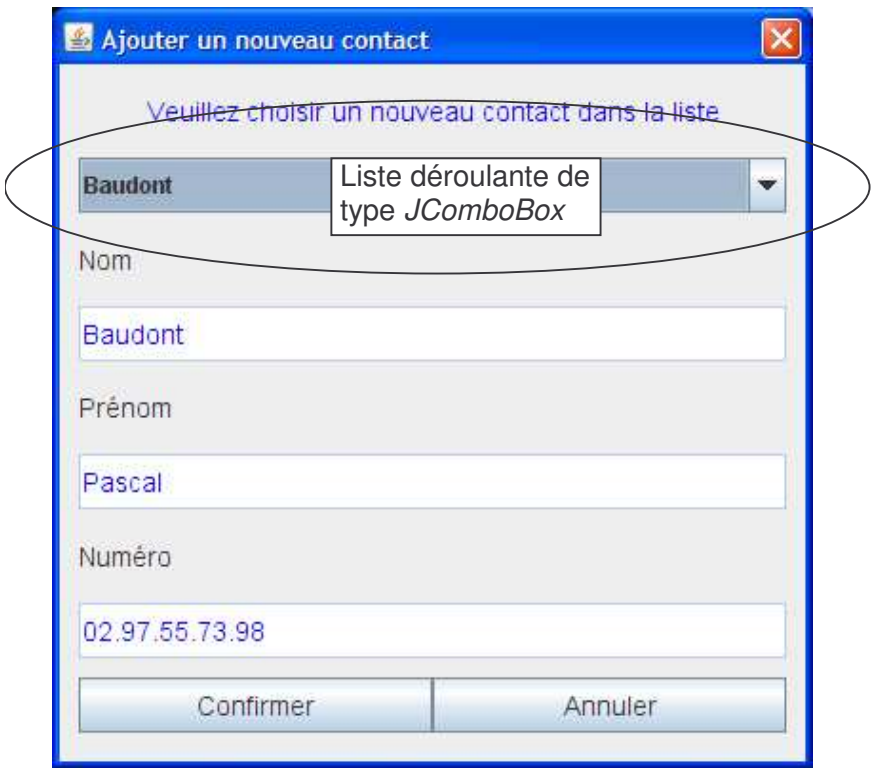


Figure 5 : la boite de dialogue WtDialogAjouter qui hérite de JDialog

Le principe est le suivant : le nouveau contact à ajouter dans l'annuaire est à sélectionner dans une liste déroulante de type *JComboBox*. Cette liste déroulante présente une liste de noms issus de la lecture d'un fichier binaire *table.bin* fourni (téléchargeable sur Moodle : AP4, semaine 19). En cliquant sur un nom, les 3 caractéristiques de la personne (nom, prénom, numéro de téléphone) s'affichent dans les 3 champs en dessous. Le clic sur le bouton « Confirmer » a pour effet d'ajouter cette nouvelle personne dans l'annuaire. Le clic sur le bouton « Annuler » rend simplement la fenêtre invisible sans ajout dans l'annuaire.

Etape1 : construire la classe *TableFiches* (paquetage *datas*, voir Figure 3). Cette classe définit une seule méthode statique *public static Hashtable<String, Fiche> lireTableFiches ()* qui a pour rôle la lecture du fichier *table.bin*. Ce fichier est constitué d'une suite d'objets de type *Fiche*. La méthode renvoie l'ensemble des fiches contenues dans le fichier sous forme d'un tableau de type *Hashtable<String, Fiche>* (où la clé de type *String* est le nom de la personne). Tester cette classe avant de poursuivre.

Etape2 : déclarer les attributs de la classe *WtDialogAjouter*, à savoir : 1 *JLabel* (pour le texte « Veuillez… »), 1 *JComboBox* (pour la liste déroulante), 3 *JLabel* (pour les 3 titres « Nom », « Prénom », « Numéro »), 3 *JTextField* (pour les 3 champs « Nom », « Prénom », « Numéro »), 2 *JButton* (Confirmer, Annuler) et le tableau des fiches (appelé *table*) de type *Hashtable<String, Fiche>*.

Etape3 : écrire le constructeur *public WtDialogAjouter (Wintel theWin)*. Ce dernier contient le code suivant :

```
super ( theWin, "Ajouter un nouveau contact", true );// appel constructeur JDialog
this.table = TableFiches.lireTableFiches(); // lecture des fiches disponibles
this.creerInterface ();                     // mise en place du décor (voir Figure 5)
this.attacherReactions();                   // écouteurs sur les boutons et JComboBox
this.setSize ( 400, 400 );
this.setVisible ( false );                  // invisible à la création
```

Etape4 : écrire la méthode *private void creerInterface()* qui met en place le décor selon la disposition de la Figure 5. Le choix est libre au niveau du (des) gestionnaire(s) de placement. Concernant la liste déroulante, il faudra la remplir avec les noms des nouveaux contacts (cf. fichier *table.bin*). Pour cela, utiliser (voir API Java !) le constructeur *JComboBox(Vector<String> items)* où le paramètre *items* contiendra la liste des noms des nouveaux contacts. Ces noms sont mémorisés dans le tableau des fiches (attribut *table*) et sont donc disponibles.

Etape5 : écrire la méthode *private void attacherReactions()* qui permettra à l'utilisateur d'interagir avec la souris.

Le clic sur un nom de la liste déroulante doit faire apparaître les coordonnées de la personne dans les 3 *JTextField*. Pour cela, attacher l'écouteur *ItemListener* (voir la méthode *void addItemListener(ItemListener aListener)* de la classe *JComboBox*) à l'objet *JComboBox* (liste déroulante). La réaction est à écrire dans une classe qui implémente *ItemListener*. Cette réaction consiste d'abord à récupérer le nom de la personne sélectionnée dans la liste (voir la méthode *Object getSelectedItem()* de la classe *JComboBox*). Avec ce nom, on récupère ensuite la fiche de la personne (le nom = la clé) que l'on affiche dans les *JTextField*.

Le clic sur le bouton « Confirmer » doit ajouter la personne sélectionnée dans l'annuaire (attribut *monAnnuaire* de la classe *Wintel*). Attacher d'abord l'écouteur *ActionListener* (méthode *void addActionListener(ActionListener l)* de la classe *JButton*). Ensuite, récupérer le contenu des 3 champs textuels (*JTextField*) pour ensuite appeler la méthode *public void ajouterAbonne(String nom, String prenom, String numTel)* de la classe *Wintel*. Pour accéder à cette méthode, il faut récupérer la référence sur l'objet *Wintel* grâce à la méthode *getOwner()* héritée de *JDialog*. A noter que *void ajouterAbonne(…)* a déjà été écrite et testée en section 3. La réaction au clic sur le bouton « Annuler » rend simplement la fenêtre *WtDialogAjouter* invisible (*setVisible(false)*).

Etape6 : déclarer 1 attribut *dialogAj* dans la classe *Wintel* de type *WtDialogAjouter*. Lors de la construction de l'objet *Wintel*, on appelle la méthode *init()* qui construit également maintenant l'objet *dialogAj*. La construction de l'objet *dialogAj* entraîne la mise en place de la fenêtre de dialogue mais elle reste invisible tant que l'utilisateur ne clique pas sur le menu « Ajouter ».

Attention : pour ajouter un nouvel abonné dans l'annuaire il faudra gérer les exceptions lancées par la méthode *ajouter(…)* de la classe *Annuaire*.

Etape7 : écrire la réaction au clic sur l'item « Ajouter » du menu « Abonnés ». Pour cela, il suffit de rendre la fenêtre de dialogue *dialogAj* visible. De cette façon, la fenêtre de dialogue pour ajouter un correspondant téléphonique apparaît à l'écran et l'utilisateur peut interagir. L'événement à capturer lors du clic est de type *ActionEvent*. De manière identique aux réactions au clic sur les items « Sauver » et « Charger » du menu « Fichier » (voir section 3 ci-dessus), il faut attacher un écouteur à l'item « Ajouter » du menu « Abonnés » : *itemAjouter.addActionListener (new EcouteurItemAjouter(this))*. La classe de réaction *EcouteurItemAjouter* (identique dans son principe à *EcouteurItemSauver* ou *EcouteurItemCharger*) peut être classe séparée ou classe interne à la classe *Wintel* (dans ce cas, passer *this* en paramètre n'est plus nécessaire).

Les classes WtDialogSupprimer et WtDialogModifier

En vous inspirant du code de la classe *WtDialogAjouter* à vous d'écrire complètement le code des classes *WtDialogSupprimer* et *WtDialogModifier*. L'aspect graphique est donné à la Figure 6.

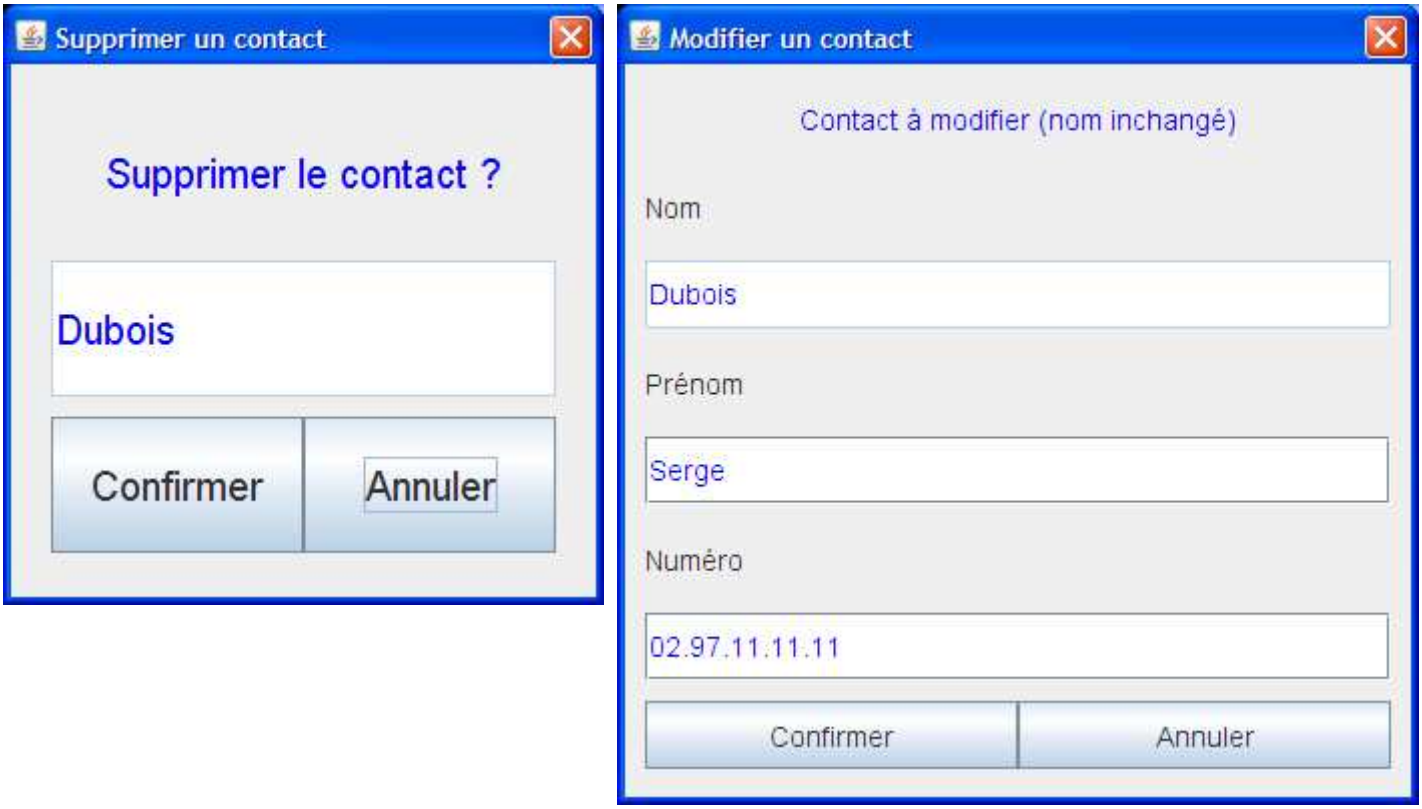


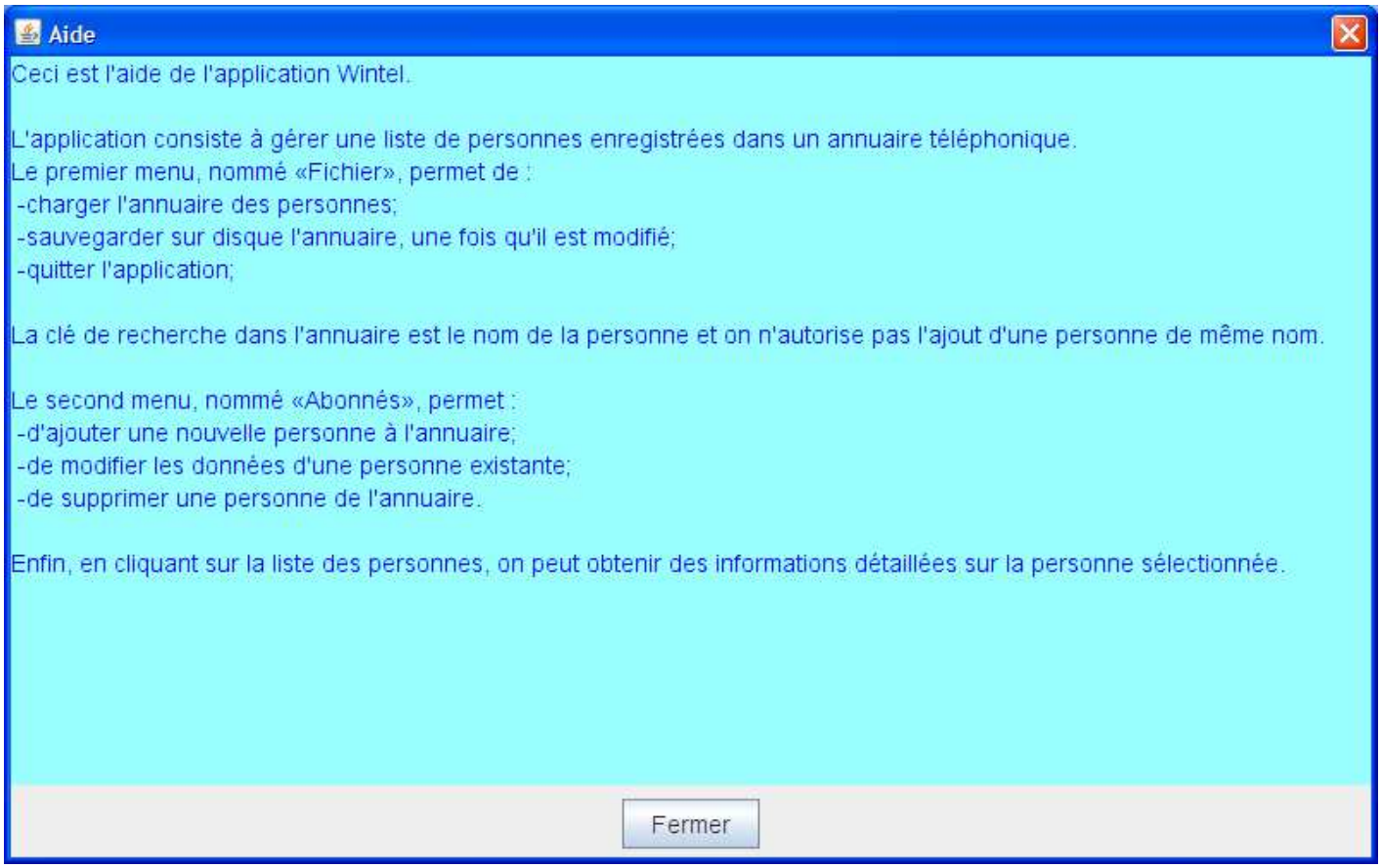
Figure 6 : les boites de dialogue WtDialogSupprimer et WtDialogModifier

On notera que pour supprimer un contact, seul son nom (clé de l'annuaire) est nécessaire. Pour la modification d'un contact, seuls les champs « Prénom » et « Numéro » sont modifiables.

5. L'AFFICHAGE DE L'AIDE

Ecrivez vous-même la réaction au clic sur l'item « Aide » du menu « Aide » de la barre de menu. La réaction doit simplement faire apparaître une boite de dialogue textuelle (avec un bouton « Fermer ») où une explication du fonctionnement de l'application doit être présentée. Cette boite de dialogue est une classe (appelée *DialogText* par exemple) qui hérite également de *JDialog* de l'API Java.

Un exemple d'aspect visuel est donné ci-dessous :



6. TRAVAIL SUPPLEMENTAIRE EVENTUEL

Ce travail supplémentaire NE donnera lieu à AUCUN bonus. Il est à réaliser uniquement si les fonctionnalités précédentes sont terminées et correctement codées et documentées. Il s'agit juste de proposer quelques idées supplémentaires à développer pour les + rapides.

Voici une liste (non exhaustive) d'ajouts/transformations possibles :

- Ecrire la réaction au clic sur le bouton « Composer ».

- Lors de la fermeture de *Wintel*, détecter une modification de l'annuaire sans sauvegarde et proposer la sauvegarde avant la fermeture définitive.
- Permettre une recherche dans la *JList* par entrée de texte dans un *JTextField* (nom du correspondant).
- Rajouter le champ « adresse » pour un contact téléphonique.
- Proposer un codage conforme au modèle MVC vu en cours.

7. MODALITES PRATIQUES

Ce mini-projet se déroule sur 8 X 1h30 (8 séances minimum sur 4 semaines) et fera l'objet d'une note de contrôle continu (partagée à 50% avec la note du contrôle de connaissance). Il est à réaliser en binôme et sera rendu sur moodle (*DUT INFO AP4*) le vendredi 1er juin à 18h au plus tard (fermeture de l'espace de rendu à 18h exactement). Chaque jour de retard donne lieu à une pénalité de 2 points sur la note finale.

L'utilisation de Eclipse est interdit. Un maximum de *javaDoc* doit apparaître dans vos classes java développées.

L'ordre dans lequel doit être développé l'application est le suivant :

- Semaine 18 : mise en place du décor (section 2).
- Semaine 19 : mise en place de l'annuaire avec réactions (section 3).
- Semaine 20 : construction de la classe *WtDialogAjouter* (section 4).
- Semaine 21 (et 22 car semaine de recadrage) : construction des classes *WtDialogSupprimer*, *WtDialogModifier* et *DialogText* (section 5).

La notation tiendra compte :

- de l'état d'avancement du projet (nombre de fonctionnalités réalisées),
- de la conformité du code par rapport au cahier des charges,
- de la qualité des commentaires *javaDoc*, de la clarté de codage (organisation claire de chacune des classes) et du respect des règles de style en Java.

Le rendu doit être une archive *.zip* ou *.tgz* (pas de *.rar* SVP) aux noms du binôme et doit contenir :

- Un fichier *readme.txt* qui explique l'état d'avancement du projet et les éventuelles fonctionnalités supplémentaires codées.
- Tous les fichiers sources (**.java*) de votre projet rangés dans 2 répertoires séparés correspondants aux 2 paquetages : un répertoire */ihm* et un répertoire */datas*. Aucun fichier *.class* n'est demandé : ils seront générés lors de l'évaluation par compilation.
- Dans un troisième répertoire */doc*, on doit trouver toute la documentation *html* de vos classes extraites à l'aide de la commande *javadoc*. Le point d'entrée généré s'appellera *index.html*.