# CNAM

# M-NFA035 Algorithmique et programmation : bibliothèques et patterns

*Dixième partie*

## Travail à effectuer

### Rappels

Quelques éléments importants avant de commencer le fil rouge :

- à terme, votre programme comportera une interface graphique ; cette interface ne vous permettra pas de lire au clavier ni d'écrire à l'écran ; donc, ne pas mettre d'ordre de lecture ou d'écriture dans vos classes ; si vous souhaitez valider vos classes, prévoyez alors une petite bibliothèque temporaire pour les entrées-sorties.
- Dans vos classes, mettez systématiquement vos champs private ou, à la limite, protected ; vous pourrez ensuite donner des accès à vos variables d'instances via des getters et des setters.

### Thème de la semaine

Cette semaine est consacrée à commencer à développer le fenêtre principale et essentiellement au développement de la barre de menu.

Etant donné que la barre de menu sera différente selon que l'utilisateur est un administrateur ou un client, nous allons faire jouer le rôle de fabrique à la classe Client avec une spécialisation dans la classe héritière : la classe Administrateur.

### *Exercice 1 : Fabrique de barres de menu*

1. Dans la classe « Client », ajoutez une nouvelle méthode « newMenuBar » dont le résultat est de type « ClientMenuBar », et retournant une nouvelle instance de la classe « ClientMenuBar » que nous allons créer par la suite.
2. Dans la classe « Administrateur », spécialisez cette méthode pour qu'elle retourne une nouvelle instance de la classe « AdministrateurMenuBar » qui sera, elle aussi, créée par la suite.

### *Exercice 2 : Barre de menu Client*

Nous allons définir « ClientMenuBar », la barre de menu pour un client, avec trois menus : Fichier, Catalogue et Mon compte.

Cette classe sera dérivée dans l'exercice suivant en « AdministrateurMenuBar ». Dans la classe fille, les menus pourront comporter plus d'items et de nouveaux menus pourront s'ajouter à ceux définis dans « ClientMenuBar ». Pour éviter de dupliquer du code, nous allons utiliser un patron de méthode :

- le constructeur de « ClientMenuBar » va ajouter les trois menus de base en utilisant des méthodes qui seront spécialisées au besoin dans la classe « AdministrateurMenuBar » ;
- le constructeur d'« AdministrateurMenuBar », ajoutera les menus spécifiques.

Pour chaque menu, nous créerons une classe étendant la classe JMenu. Pour cet exercice, le suivant et ceux à venir, pensez à définir un mnémonique et un raccourci clavier

pour chaque menu et item de menu.

1. Créez un package ui.menu dans lequel nous mettrons toutes les classes concernant les menus.
2. Créez une classe « MenuFichierClient » pour le menu « Fichier » étendant « JMenu ». Ce menu comportera deux items :
   - Enregistrer pour enregistrer les données dans le fichier par défaut
   - Quitter pour quitter l'application.

   Pour la suite, conservez les références vers ces deux items dans des variables d'instance.

   Si pour l'item Quitter vous pouvez utiliser l'action prévue à cet effet précédemment, il vous faudra écrire celle de l'item enregistrer. Vous pouvez créez une sous-classe de votre menu étendant « AbstractAction ». L'action à effectuer consiste simplement à appeler la méthode « sauver » de « GestionnaireSauvegarde ». En cas d'exception levée, signalez à l'utilisateur que l'enregistrement n'a pas pu avoir lieu par l'intermédiaire d'une fenêtre de dialogue créée à l'aide « JOptionPane.showMessageDialog ».
3. Créez de la même manière deux classes héritières de JMenu : la première pour consulter le catalogue, la seconde pour gérer le compte utilisateur.

   Pour l'instant, ces deux menus resteront vides.
4. Créez maintenant la classe « ClientMenuBar ». Vous y définirez 3 méthode protected, une pour chaque menu créant et retournant une nouvelle instance du menu concerné. Enfin vous ajouterez un constructeur ajoutant les trois menus à l'aide des méthodes que vous venez d'écrire. Prenez soin de conserver une variable d'instance référençant chacun des menus.

### *Exercice 3 : Barre de menu Administrateur*

Cette classe nommée « AdministrateurMenuBar » étendra la classe « ClientMenuBar ». Il n'y aura pas de modification à apporter aux menus Fichier et Mon Compte. Par contre, le menu Catalogue doit permettre en plus de manipuler le catalogue (ajout, modification et suppression de produits). Il faudra aussi ajouter la possibilité de manipuler les bons de commande du magasin et les utilisateurs de l'application.

Pour l'instant, les nouveaux menus resteront vides.

1. Créez deux classes étendant JMenu, une pour les utilisateurs et l'autre pour la gestion des commandes.
2. Créez une extension du menu catalogue en prévision. Pour l'instant, nous n'ajouterons rien de plus.
3. Créez la classe « AdministrateurMenuBar » étendant « ClientMenuBar ». Y spécialiser la méthode de création du menu catalogue pour générer, non plus une instance du menu destiné aux clients, mais une instance de celui destiné à un administrateur. Définissez le constructeur pour ajouter les deux nouveaux menus.

### *Exercice 4 : Activation et désactivation dans le menu Fichier*

Nous avons programmé le gestionnaire de fichiers dans une étape antérieure. Cette classe conserve dans un booléen « modifie » le fait que des données ont été ajoutées, supprimées ou modifiées dans la structure de données.

L'objectif de cet exercice est d'activer l'item « Enregistrer » du menu « Fichier » que si des données ont été modifiées depuis le lancement de l'application ou la dernière sauvegarde, et de n'activer l'item « Quitter » que si aucune modification n'a été apportée depuis le dernier enregistrement.

Nous allons construire un patron « observateur/observé » pour gérer ceci. L'observé est le gestionnaire de sauvegarde. L'observateur est le menu fichier. Il faudra signaler au menu fichier chaque fois qu'une modification a été apportée à la structure de donnée et chaque fois qu'il y aura eu un enregistrement dans le fichier.

1. Dans un premier temps, nous définissons ce que nous attendons d'un observateur pour qu'il puisse recevoir ce genre d'événements par une interface « FichierListener » avec deux méthodes (« fichierEnregistre » et « modificationSDApportee ») sans paramètre et sans résultat, la première destinée à signaler que le fichier a été enregistré, la seconde pour signaler que la structure de données a été modifiée.
   - Cette interface sera encapsulée dans la classe GestionnaireSauvegarde.
   - De plus, elle devra étendre l'interface « EventListener » prédéfinie dans l'API Java pour permettre d'utiliser la classe « EventListenerList », elle aussi prédéfinie dans l'API Java, qui permet de gérer des listes d'observateurs de manière sécurisée.
2. Dans un deuxième temps, dans la classe « GestionnaireSauvegarde », assurez-vous que la valeur du booléen « modifie » n'est modifiée que dans la méthode private « setModifie ».

   C'est cette seule méthode qui déclenchera les deux événements que nous avons définis. Si le booléen « modifie » est bien private, les autres classes ne peuvent agir sur cette variable d'instance que par le biais de la méthode marquer. Dans la classe « GestionnaireSauvegarde », normalement, seules les les méthodes suivantes : `sauver(File fichier)` et `restituer(File fichier)` doivent remettre « modifie » à false. Assurez-vous qu'elles ne le font que par un appel à la méthode « setSauvegarder ».
3. Dans un troisième temps, nous allons ajouter une liste d'observateurs à la classe « GestionnaireSauvegarde ». Pour ce faire, déclarez une variable d'instance « eventListenerList » de type « EventListenerList ». Initialisez-là avec une nouvelle instance de la classe « EventListenerList » soit dans la ligne de déclaration, soit dans le constructeur de « GestionnaireSauvegarde ».
4. Ensuite, nous allons permettre aux observateurs de s'enregistrer auprès de l'observé. Pour réaliser cela, il suffit d'ajouter une méthode « addFichierListener » prenant en paramètre « listener », un « FichierListener », et se contentant d'appeler la méthode « add » de « eventListenerList » avec en premier paramètre l'expression `FichierListener.class` et, en second paramètre, `listener`.
5. Puis, ajoutez les deux méthodes qui suivent en fin de « GestionnaireSauvegarde ».

```java
/**
 * Permet de signaler à tous les observateurs que le fichier à été
 * enregistré.
 */
public void fireFichierEnregistre() {
	// récupération de la liste des observateurs.
	FichierListener[] tab = eventListenerList
			.getListeners(FichierListener.class);
	// envoi de l'évènement à chacun des observateurs
	for (FichierListener fl : tab)
		fl.fichierEnregistre();
}

/**
 * Permet de signaler à tous les observateurs que la structure de données
 * a été modifiée.
 */
public void fireModificationSDApportee() {
	// récupération de la liste des observateurs.
	FichierListener[] tab = eventListenerList
			.getListeners(FichierListener.class);
```

```
        // envoi de l'évènement à chacun des observateurs
        for (FichierListener fl : tab)
              fl.modificationSDApportee();
}
```

6. Pour en finir avec « GestionnaireSauvegarde », dans la méthode « setModifie », faites appel à la seconde des deux méthodes ci-dessus quand le booléen vaut true, et faites appel à la première dans le cas contraire.
7. Enfin, modifiez la classe du menu fichier pour qu'elle implémente « FichierListener ». Dans la méthode « fichierEnregistre » signalant que le fichier a été enregistré, l'item Enregistrer du menu sera désactivé alors que Quitter sera activé. Réciproquement, la méthode « modificationSDApportee » fera l'inverse. Il ne reste plus qu'à appeler « fichierEnregistre » dans le constructeur de la classe pour initialiser l'état d'activation des deux items de menu.

### *Exercice 5 : Ajout de la barre de menu*

Dans le constructeur de « MainFrame », ajoutez la bonne barre de menu en récupérant l'utilisateur par le getter approprié de « Systeme » et en appelant dessus la méthode « newBarreMenu ».

Démarrez votre application et, si ce n'est pas déjà fait, créez un client à partir de la fenêtre d'identification. Puis testez les barres de menu avec l'administrateur et le client que vous venez de créer.

Pour l'administrateur vous devez obtenir 5 menus : « Fichier », « Catalogue », « Mon compte », « Utilisateurs » et « Commandes ». Tous les menus doivent apparaître mais seul « Fichier » doit contenir deux items : « Enregistrer » et « Quitter ». Seul « Quitter » doit être actif. Pour le client, Les menus « Utilisateurs » et « Commandes » ne doivent pas apparaître.