CEFOD BUSINESS SCHOOL

Département de Génie informatique

Licence 2

Mini Projet de fin d’année Licence 2

**Rapport Mini Projet 2 :**

**Gestion des Commandes**

Nom : **MBAÏRAM**

Prénoms : **BENJAMIN BEASSOUM**

Date**: 08/07/2024**

Ce rapport décrit la création d’une application qui crée une hiérarchie de classes pour mettre en relation les restaurants, les clients qui veulent commander de menu et les livreurs.

**N.B**: Le contenu de ce rapport est fait en prenant en compte les ajustements personnels ajoutés au projet initial.

**Objectifs** : La manipulation de la hiérarchie des classes en java

Ce projet se concentre sur les définitions des classes Personne, Client, Propriétaire, Livreur, Menu, Restaurant, Commande et TestAlloResto

**Outils et Technologies**

Langage de programmation : **Java**

Environnement de développement : **Intellij**

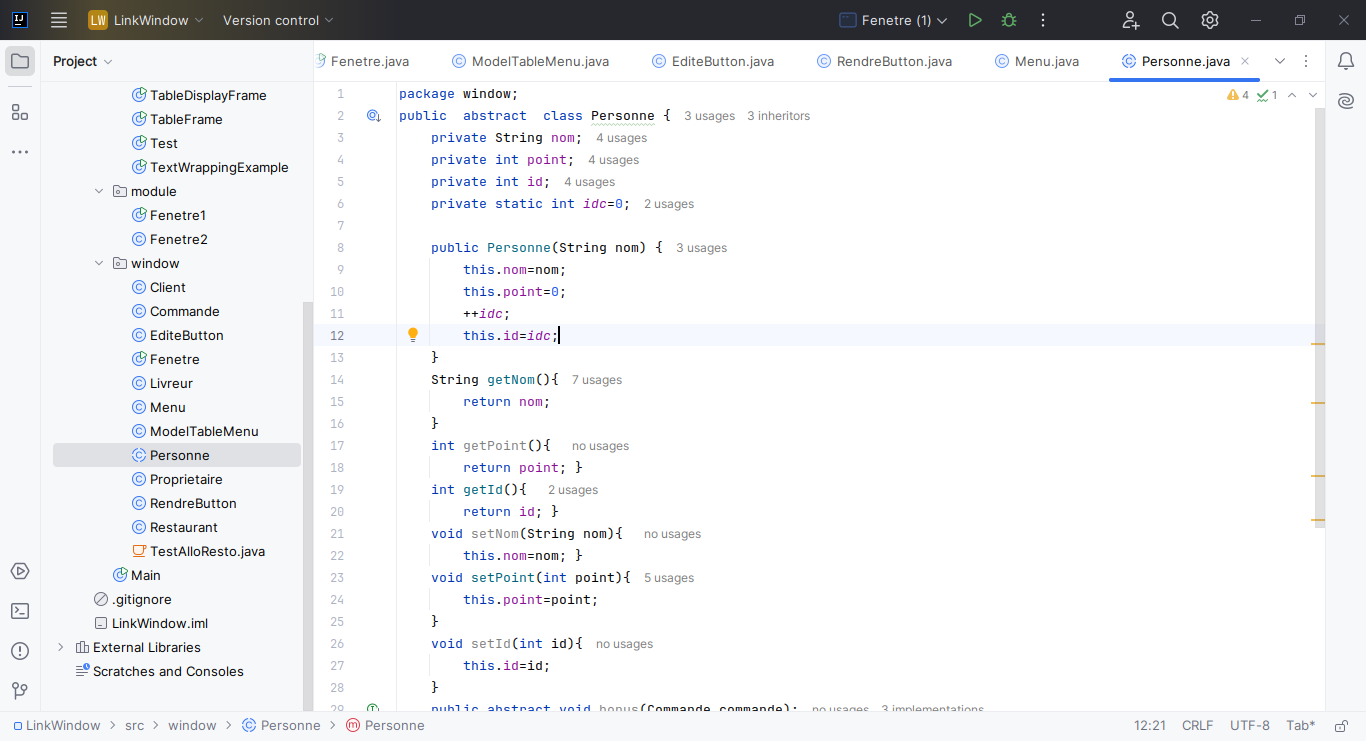
Package principaux utilisés : **javax.sing.\***, **java.util.\*** et **java.awt.\***

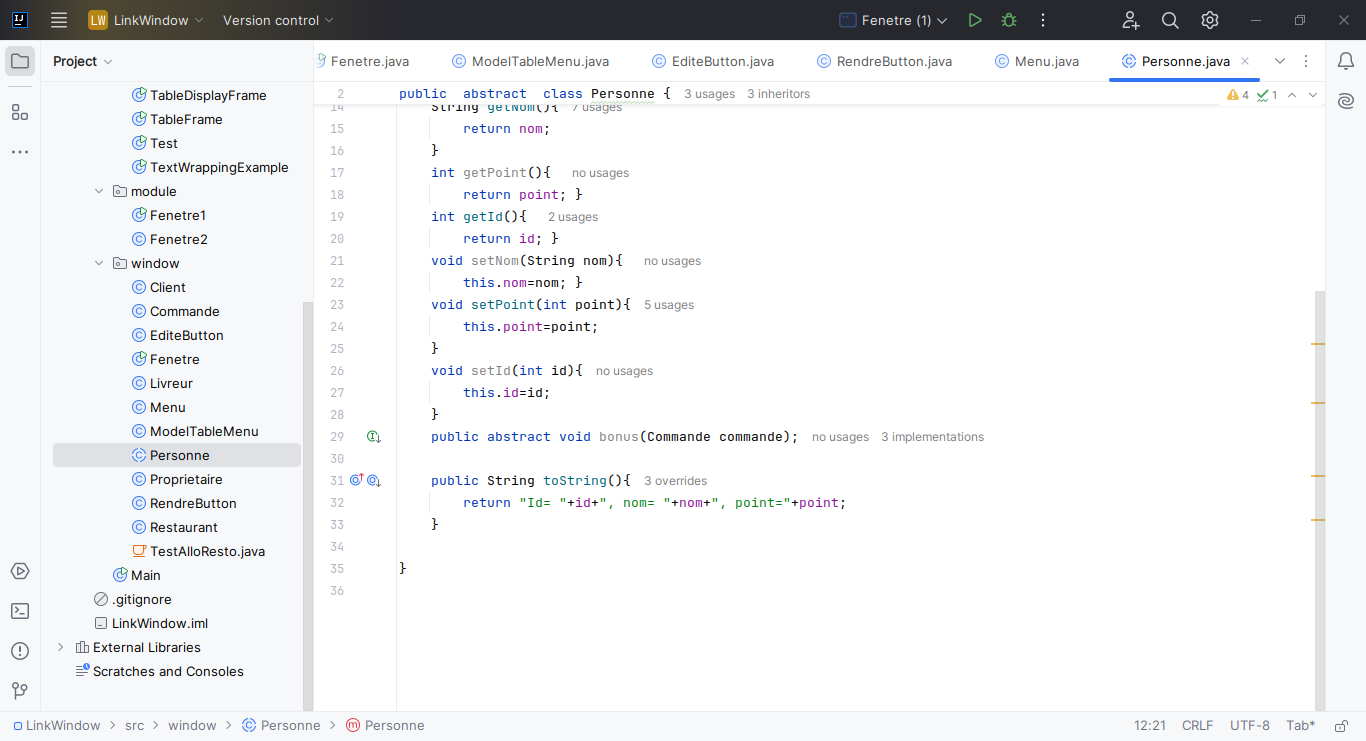
**Méthodologie d’implémentation :**

1. Classe abstraite **Personne** :

* Définition des attributs **nom**, **point** et **id** ;
* Définition de l’attribut **idc** (compteur de la classe personne) et initialisé à 0 ;
* Implémentation du constructeur avec une incrémentation de **idc** à chaque création d’une instance de personne ;
* Implémentation des assesseurs et des modificateurs des attributs **nom**, **point** et **id** ;
* Implémentation des méthodes : méthode abstraite bonus et méthode **toString**() ;

Aperçu code :

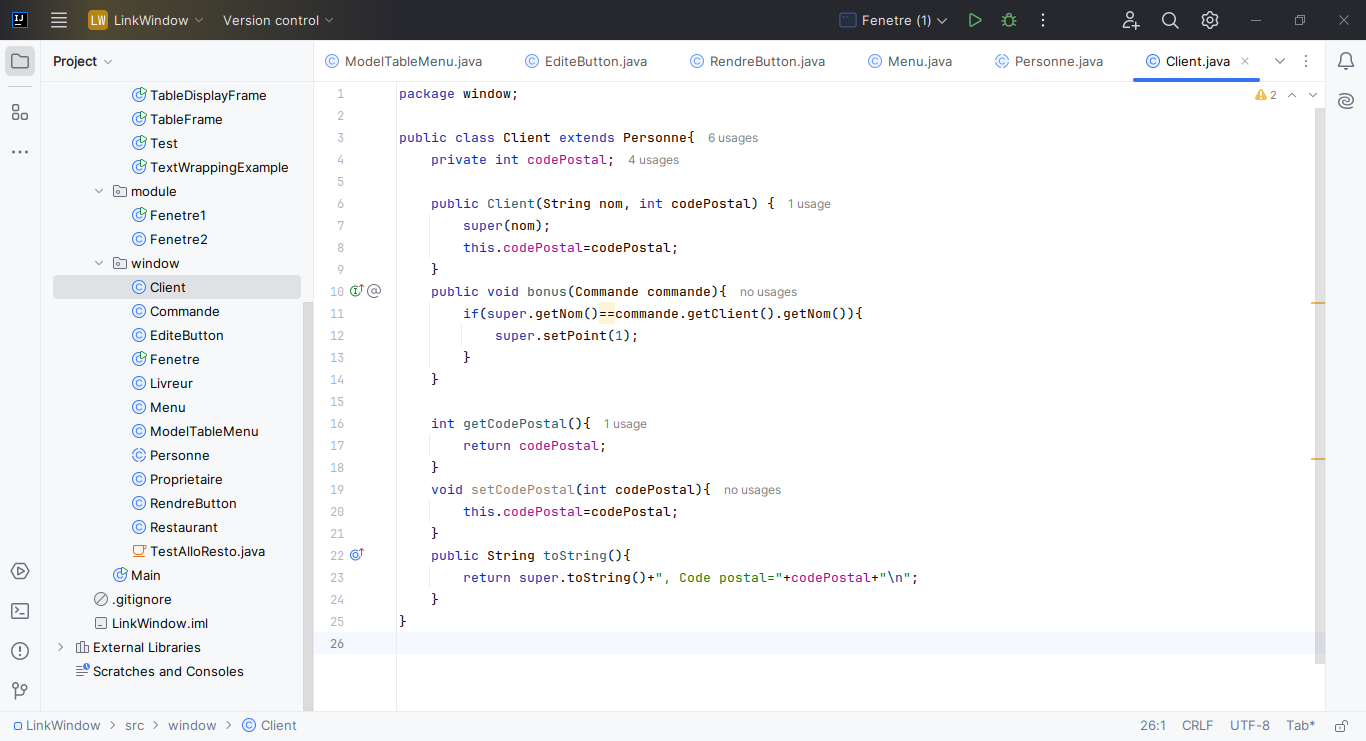




2. Classe **Client** qui hérite de la classe Personne :

* Définition de l’attribut **codePostal** ;
* Implémentation du constructeur de la classe Client avec les paramètres **nom** et **codePostal** ;
* Utilisation de **super**(nom) pour faire appel au constructeur de la classe Personne ;
* Implémentation de l’assesseur et du modificateur de l’attribut codePostal ;
* Implémentation des méthodes : méthode **bonus** () et méthode **toString** () ;

Aperçu code :



3. Classe Propriétaire qui hérite de la classe Personne :

* Implémentation du constructeur de la classe qui prend en paramètre une variable nom utilisée lors de l’appel du constructeur de la classe Personne ;
* Implémentation des méthodes : méthode **bonus** () et méthode **toString** () ;

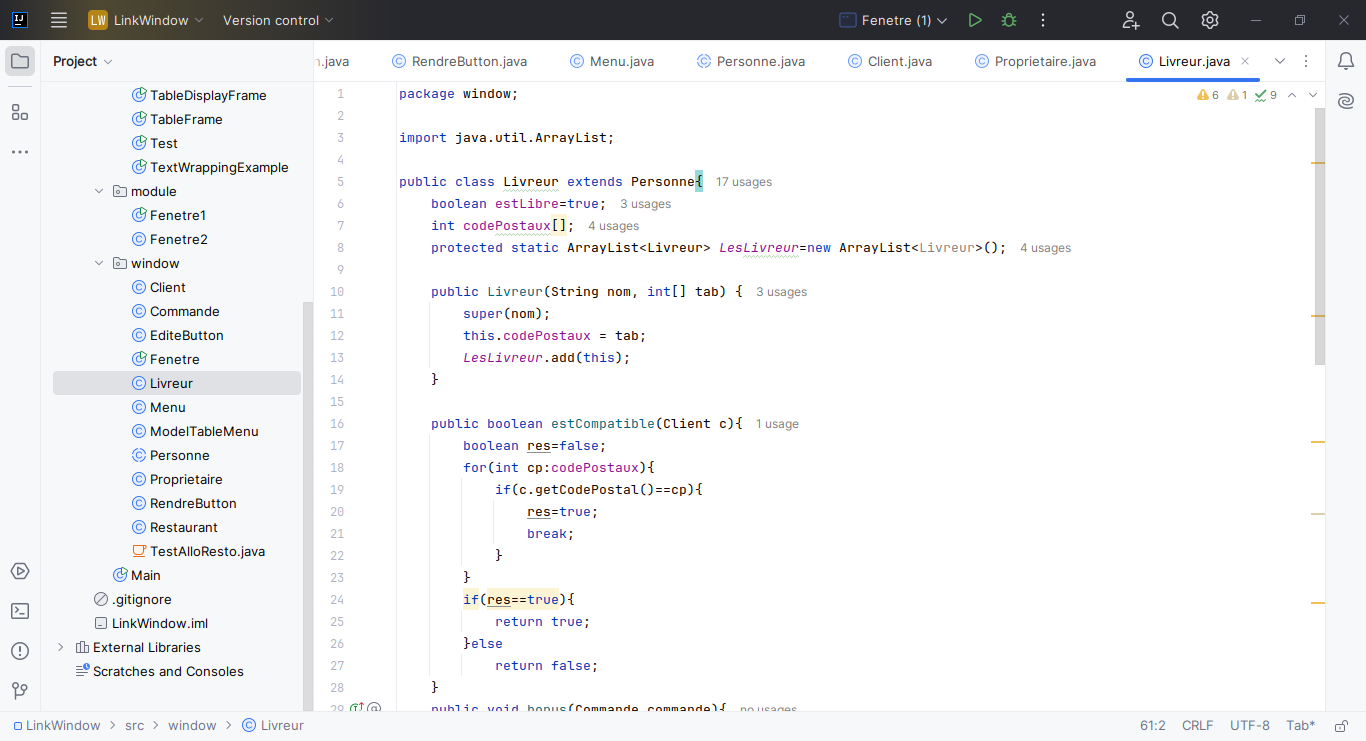
Aperçu code :

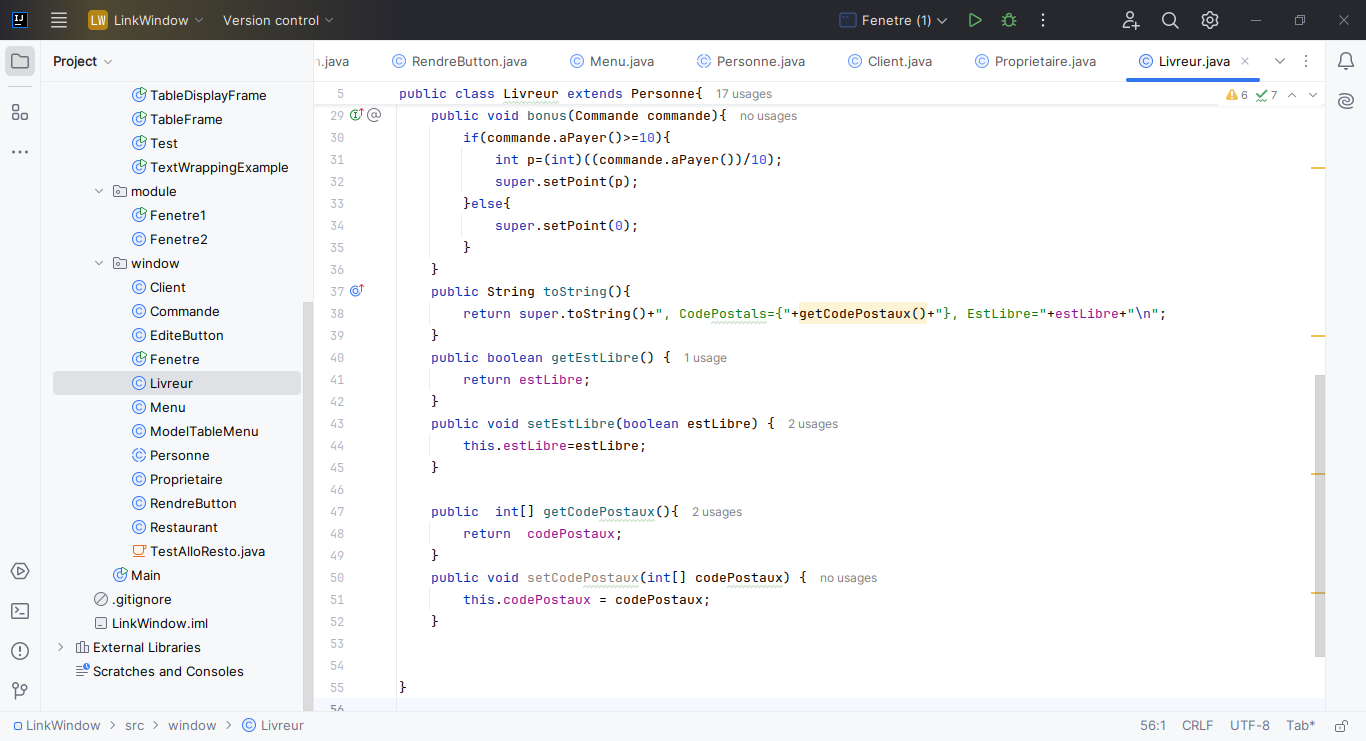


4. Classe **Livreur** qui hérite de la classe Personne :

* Définition des attributs **estLibre** de type booléen et **codePostaux** de type tableau ;
* Définition de l’attribut **Leslivreur** de type ArrayList qui stocke tous les livreurs créés **(ajout personnel)** ;
* Implémentation du constructeur de la classe qui prend en paramètre le nom et le tableau des code postaux ;
* Utilisation de **super**(nom) pour faire appel au constructeur de la classe Personne ;
* Implémentation des assesseurs et des modificateurs des attributs **estLibre** et **codePostaux** ;
* Implémentation des méthodes : méthode **estCompatible** (), méthode bonus et méthode **toString** ();

Aperçu code :

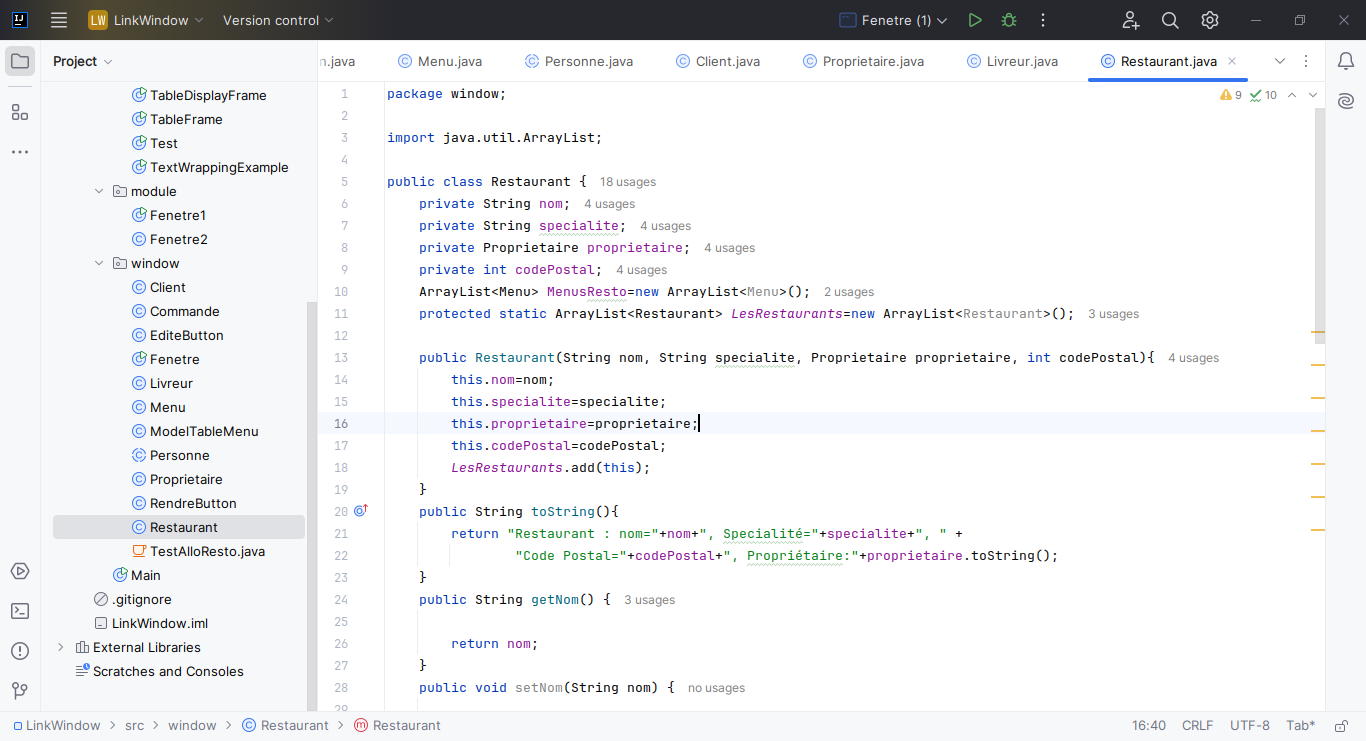


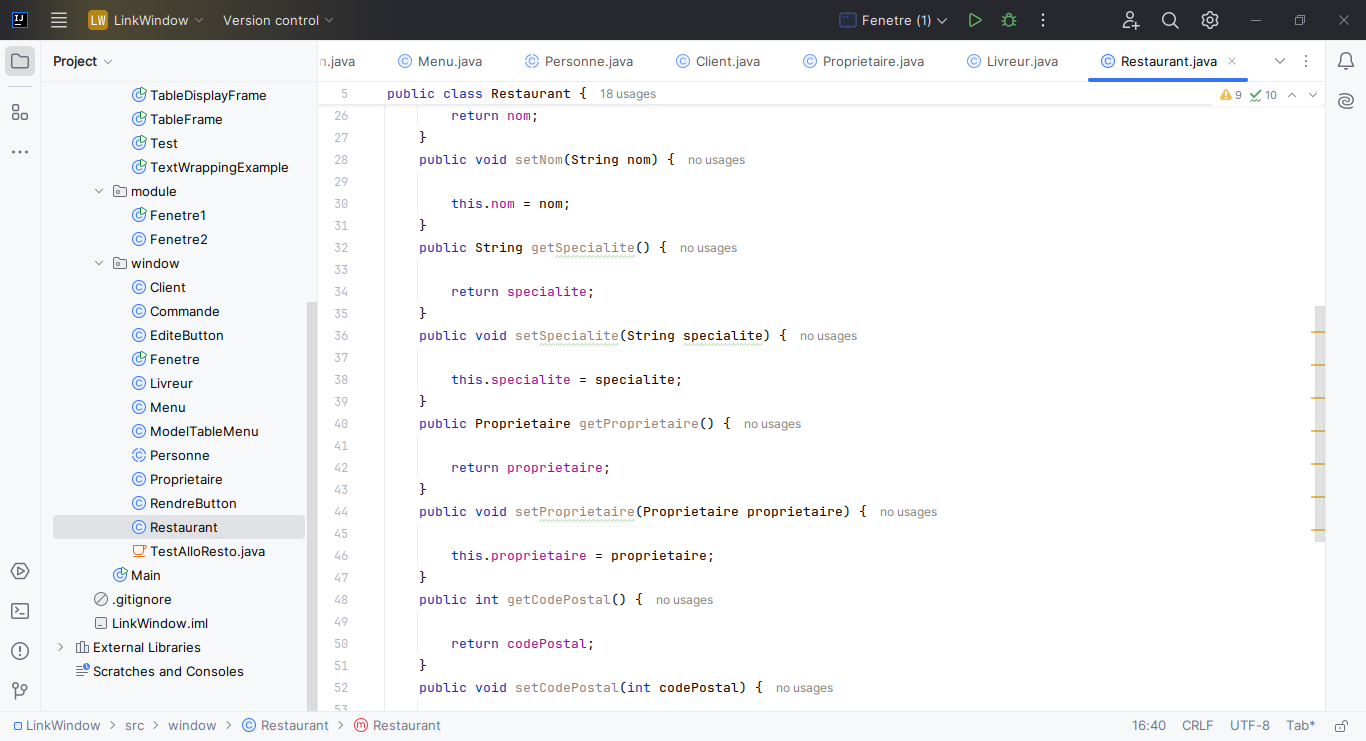


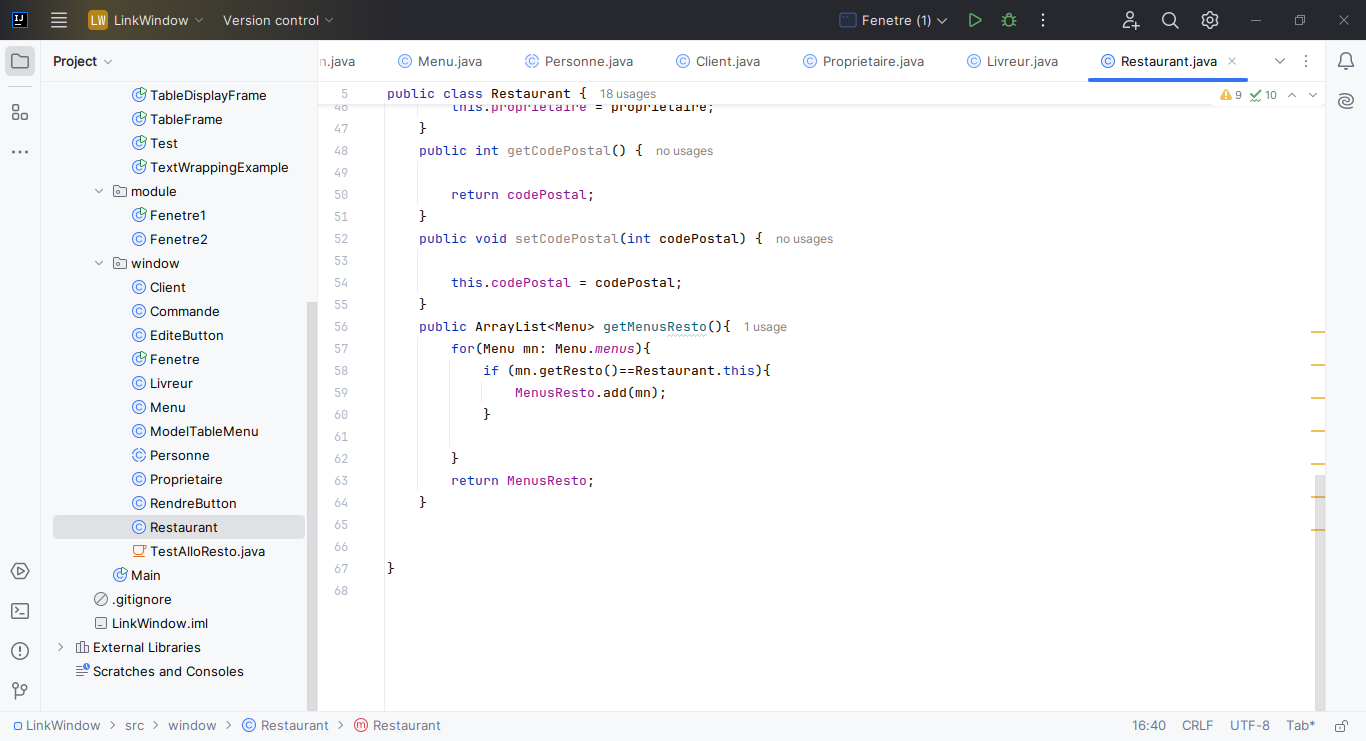
5. Classe **Restaurant** :

* Définition des attributs **nom** de type String, **spécialité** de type String, **propriétaire** de type Propriétaire et **codePostal** de type entier ;
* Définition de l’attribut **LesRestaurants** de type ArrayList qui stocke tous les Restaurants créés **(ajout personnel)** ;
* Implémentation du constructeur de la classe qui prend en paramètre tous les attributs de la classe ;
* Implémentation des assesseurs et des modificateurs des attributs de la classe ;
* Implémentation de la méthode **toString** () ;

Aperçu code :



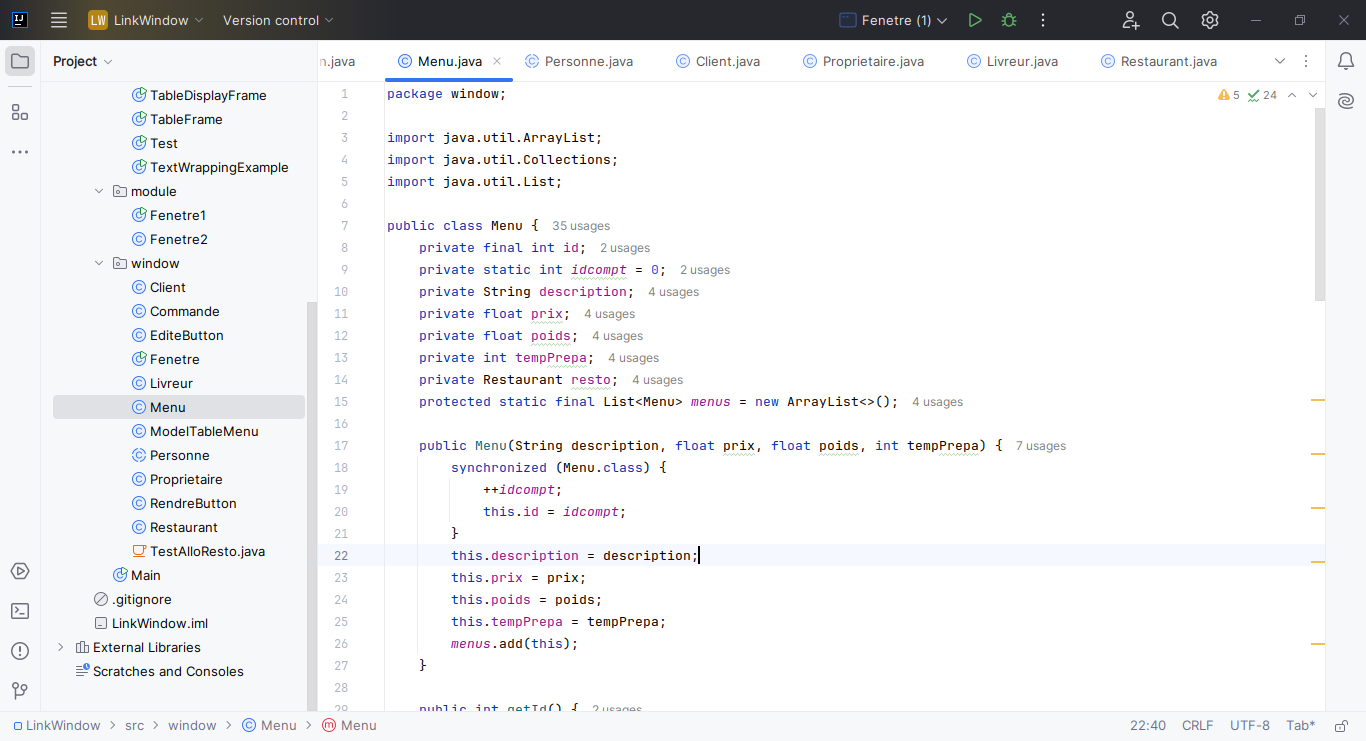




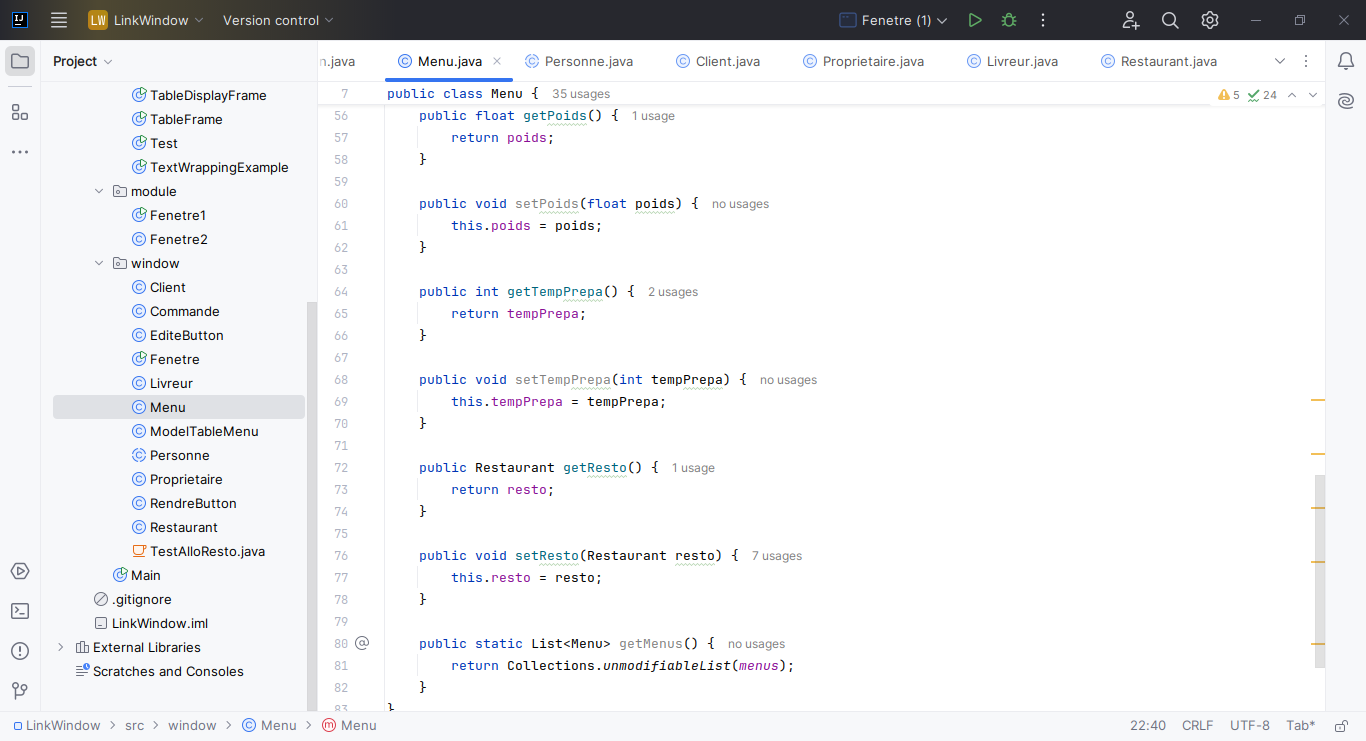
6. Classe **Menu** :

* Définition des attributs **description** de type String, **prix** de type float, **poids** de type float, **TempPrepa** de type int et **resto** de type Restaurant ;
* Définition de l’attribut menus de type ArrayList qui stocke tous les menus créés et les attributs **id** et **idcompt** qui aident à attribuer automatiquement à un menu un identifiant. Utile dans le choix des menus en interface graphique pour le client **(ajout personnel) ;**
* Implémentation du constructeur de la classe qui prend en paramètre tous les attributs de la classe sauf l’attribut resto, ajouté après création du Menu grâce au méthode **setResto**() de la classe;
* Implémentation des assesseurs et des modificateurs des attributs de la classe ;
* Implémentation de la méthode **toString** ();

Aperçu code :



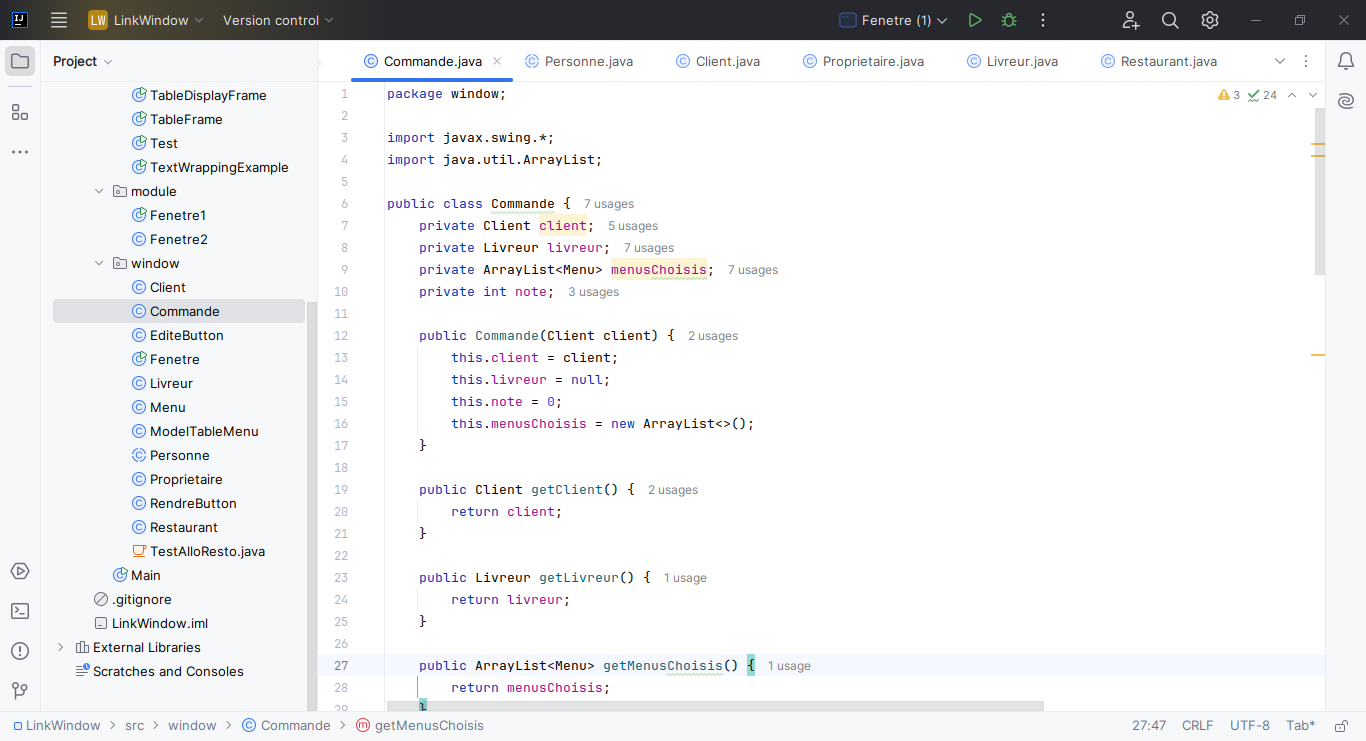




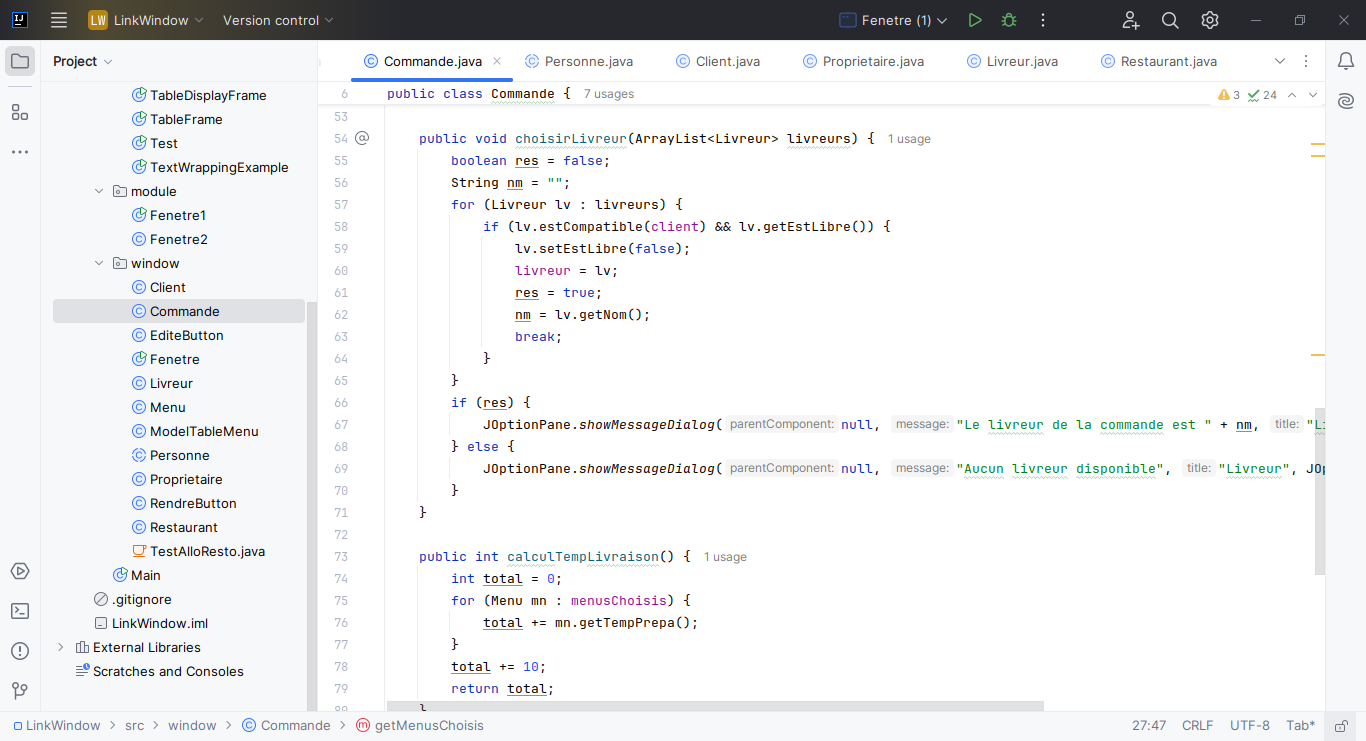
7. Classe **Commande** :

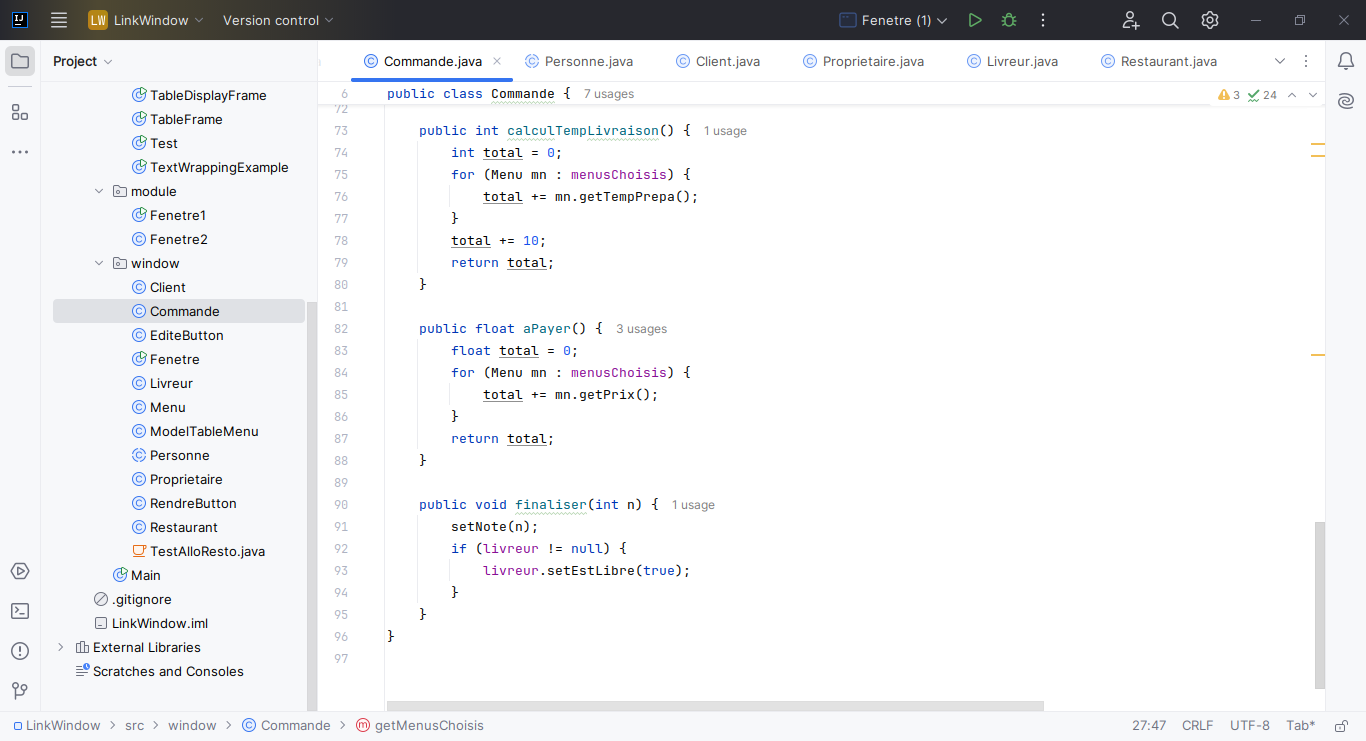
* Définition des attributs **client** de type Client, **livreur** de type Livreur, **menusChoisis** de type ArrayList, **note** de type int ;
* Implémentation de constructeur de la classe qui prend en paramètre l’attribut client ;
* Implémentation des méthodes : la méthode **ajouterNewMenu**() qui aide à ajouter un menu dans le panier du client, la méthode **supprimerMenu**() qui aide à supprimer un menu dans le panier du client, la méthode **choisirLivreur** qui aide à choisir un aide pour la commande, la méthode **calculTempLivraison**() qui aide à calculer le temps nécessaire à la livraison de la commande, la méthode **aPayer**() qui aide à calculer le prix de la livraison, la méthode **finaliser**() qui aide à noter la commande et méthode **toString()** ;

Aperçu code :









8. Classe **TestAlloResto** :

Cette classe définie à la base pour effectuer les tests sur la création des différents objets de nos classes ci-haut a été modifié afin que tout le processus sensé se dérouler dans la console se déroule en interface graphique avec un latitude laissé à l’utilisateur d’entrer ces propres choix.

Cette classe se présente maintenant comme ci :

* Héritage de la classe **JFrame** de **javax.swing.\* ;**
* Définition de tous les attributs des composants graphiques ;
* Utilisation des modèles de positionnement des composants graphiques : **FlowLayout**, **GridLayout**, **BorderLayout** et **GridBagConstraints** ;
* Utilisation des composants graphique :

- **JLabel** pour afficher du texte ;

- **JButton** pour créer des buttons ;

- **JTextfield**, **JTextPane** et **JTextarea** pour laisser le choix à utilisateur à entrer du texte ;

- **JPanel** pour créer des panneaux ;

- **CardLayout** pour gérer les composants graphiques à l’écran ;

- **JscrollPane** qui aide à dérouler le contenu dans un panneau ;

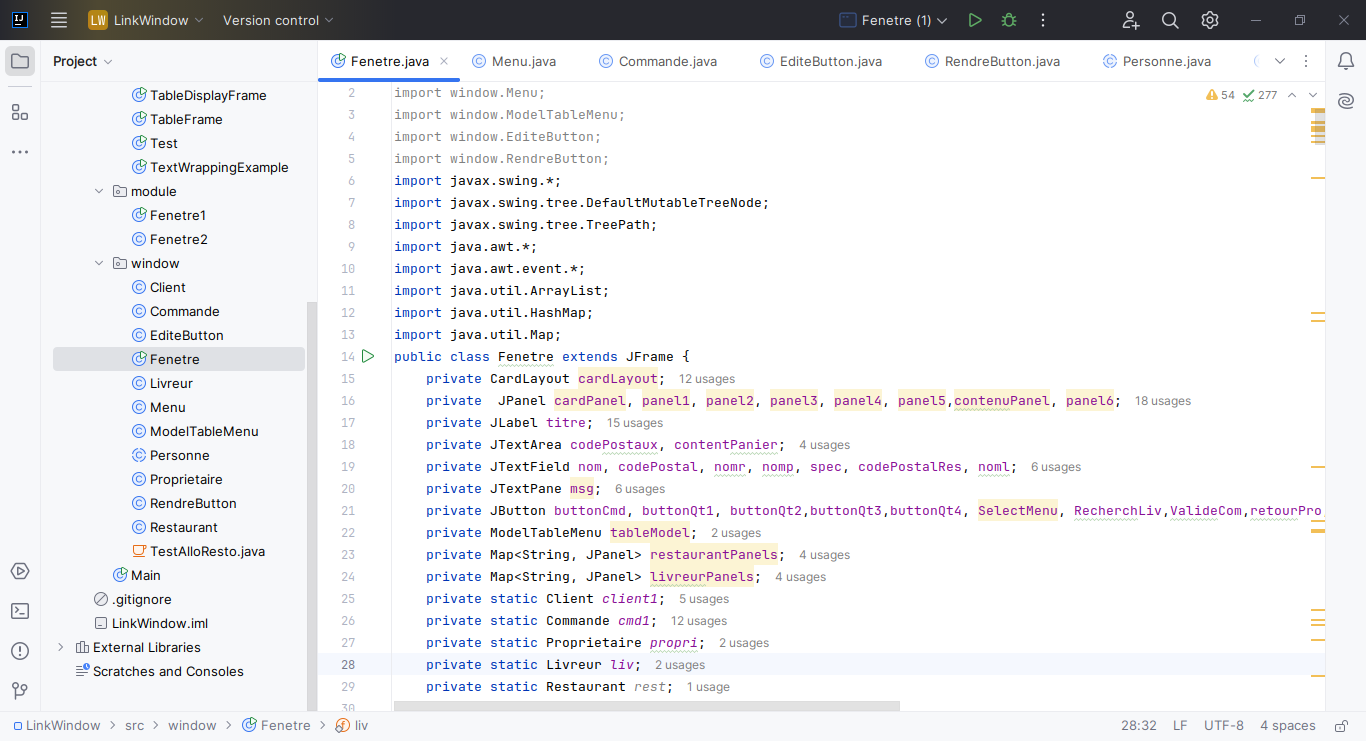
* Implémentation de modèle personnel de listes déroulantes avec la création de deux fichiers : EditeButton.java et RendreButton.java important dans l’affichage de tous les restaurants disponibles dans le site.

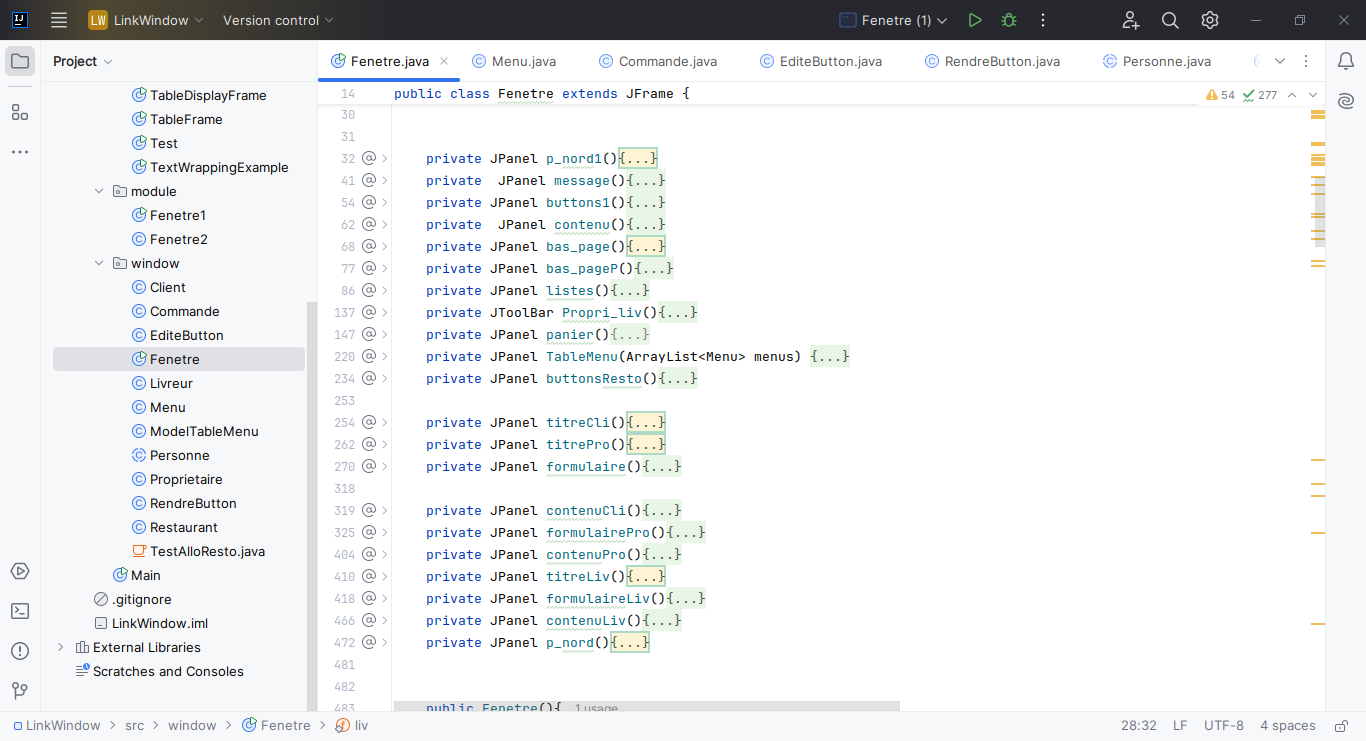
- RendreButton aide à rendre chaque restaurant un button qui peut actionner pour afficher les menus du restaurant et autres ;

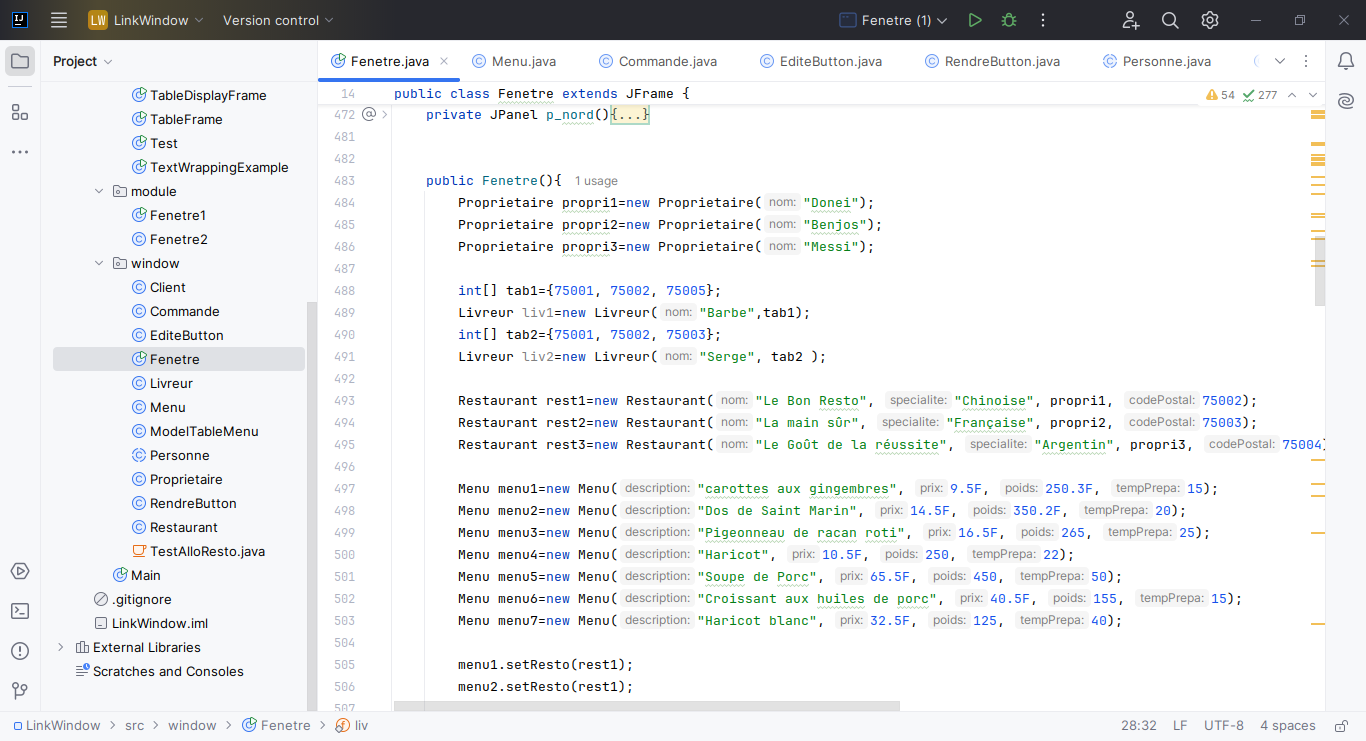
- EditeButton aide à éditer chaque button ;

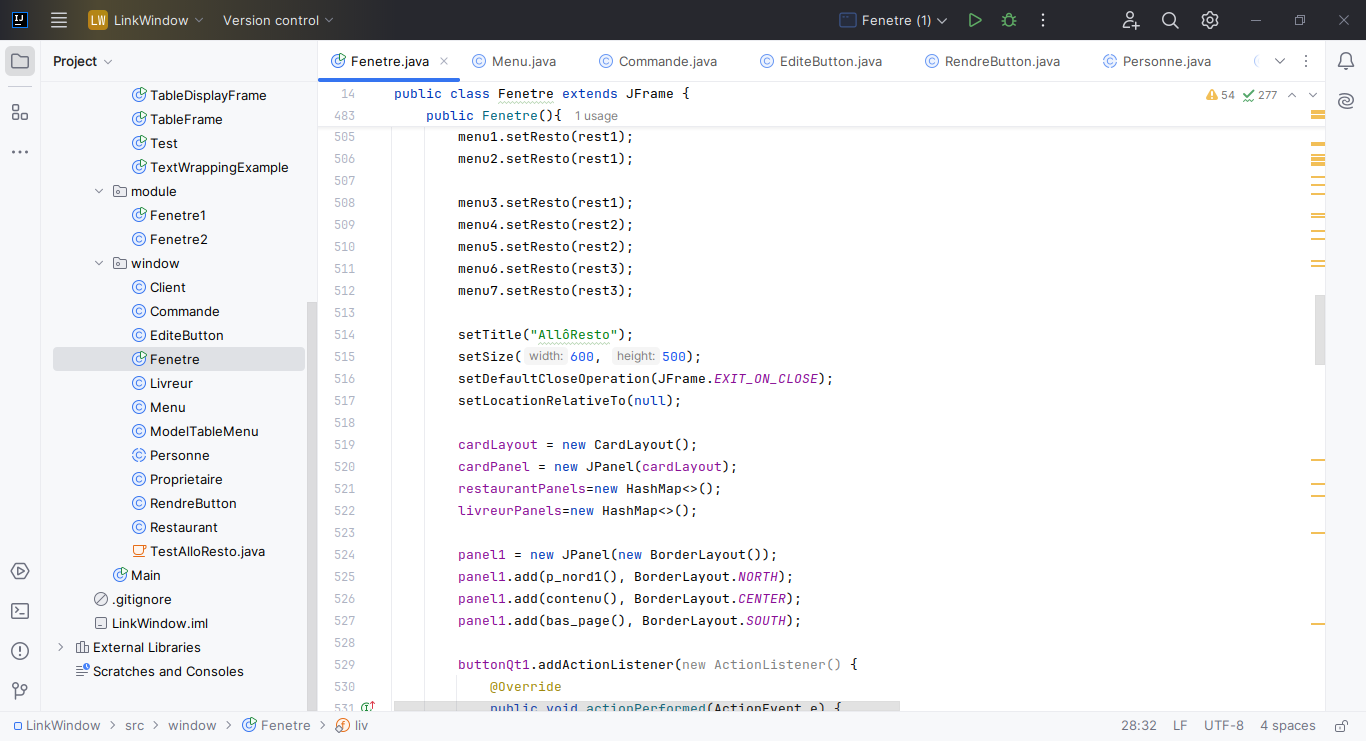
* Implémentation de modèle personnel de table pour afficher les menus de chaque restaurant dans un tableau de menu. La création du fichier ModelTableMenu aide à créer ce modèle en utilisation le package **AbstractTableModel** ;
* Implémentation du composant cardLayout afin de permuter entre différents panneaux en restant dans la mếme fenêtre ;
* Implémentation du constructeur de la classe avec la création des différents objets de type Livreur, Client, Propriétaire, Restaurant, Menu et Commande ;
* Implémentation de la méthode **main**() qui aide à créer une instance de la classe et affiche la fenêtre avec tous ces composants ;

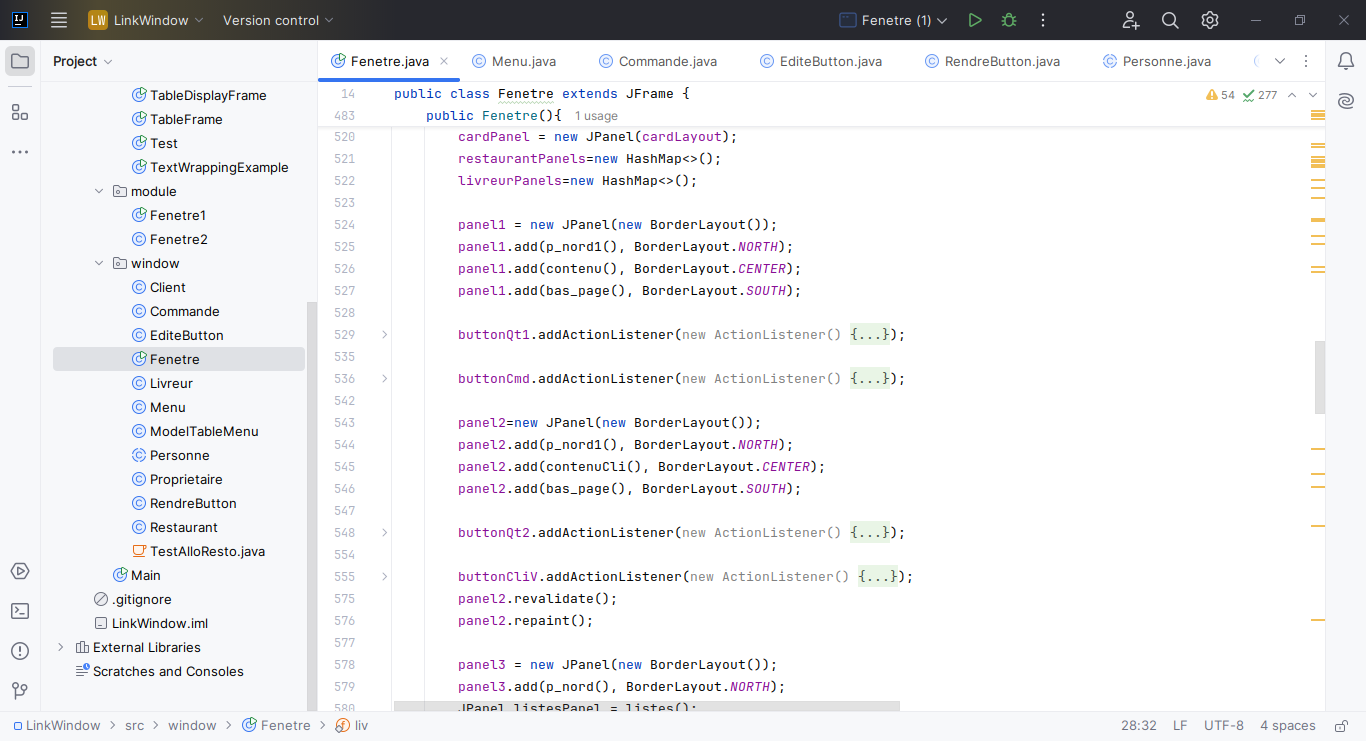
Aperçu du code (Certaines parties du code ont été masquées) :

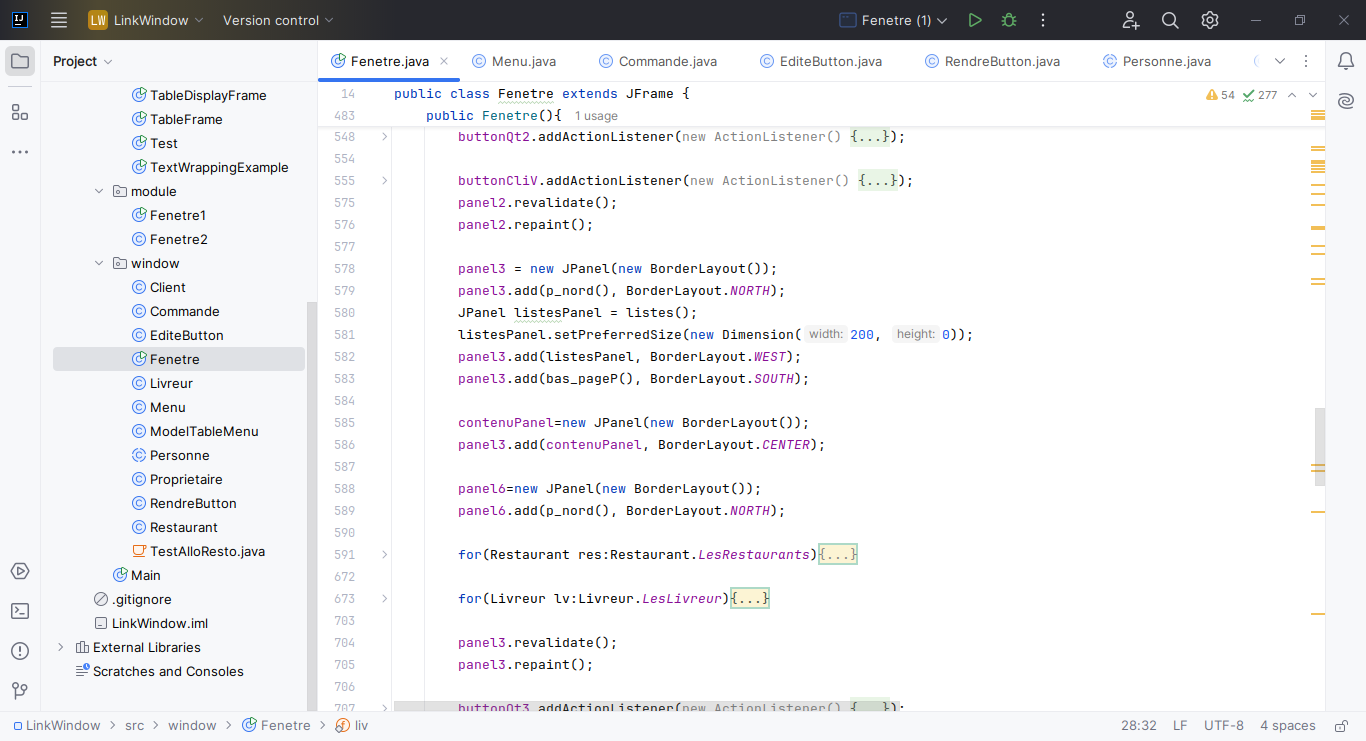


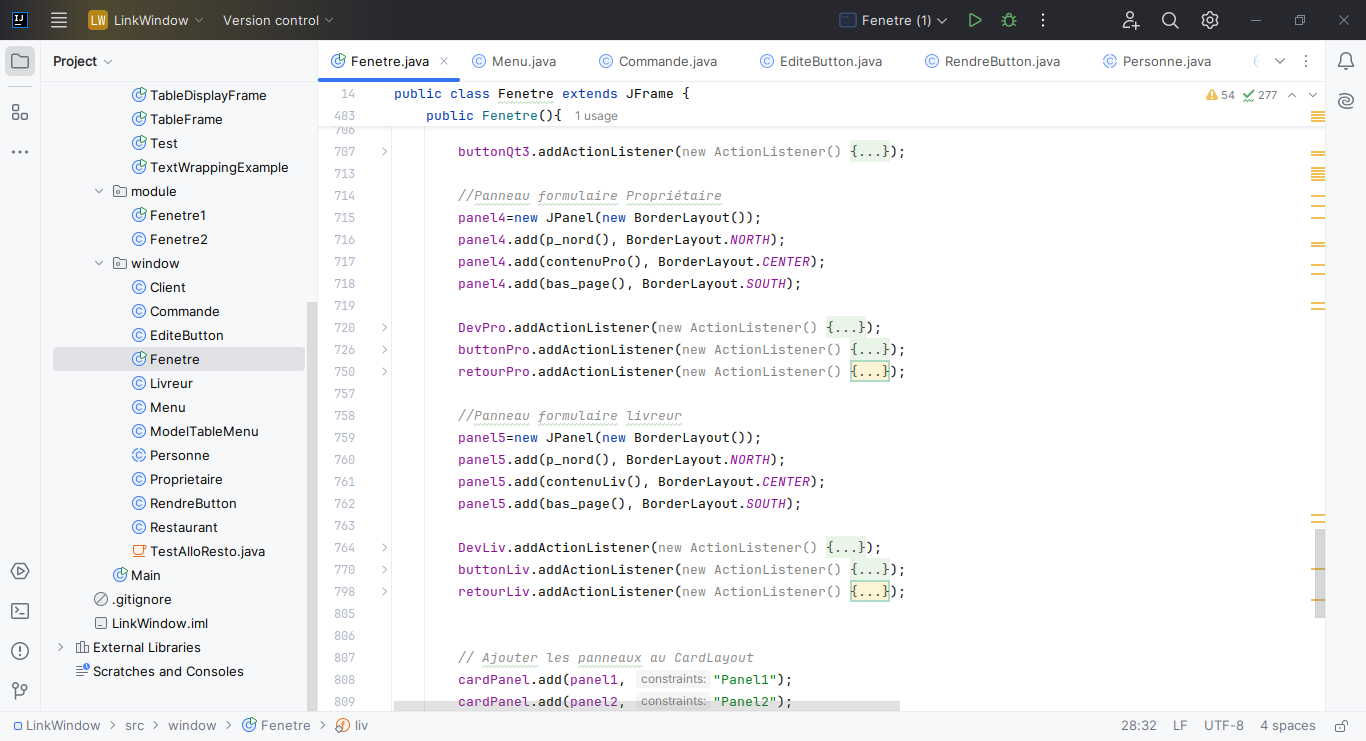


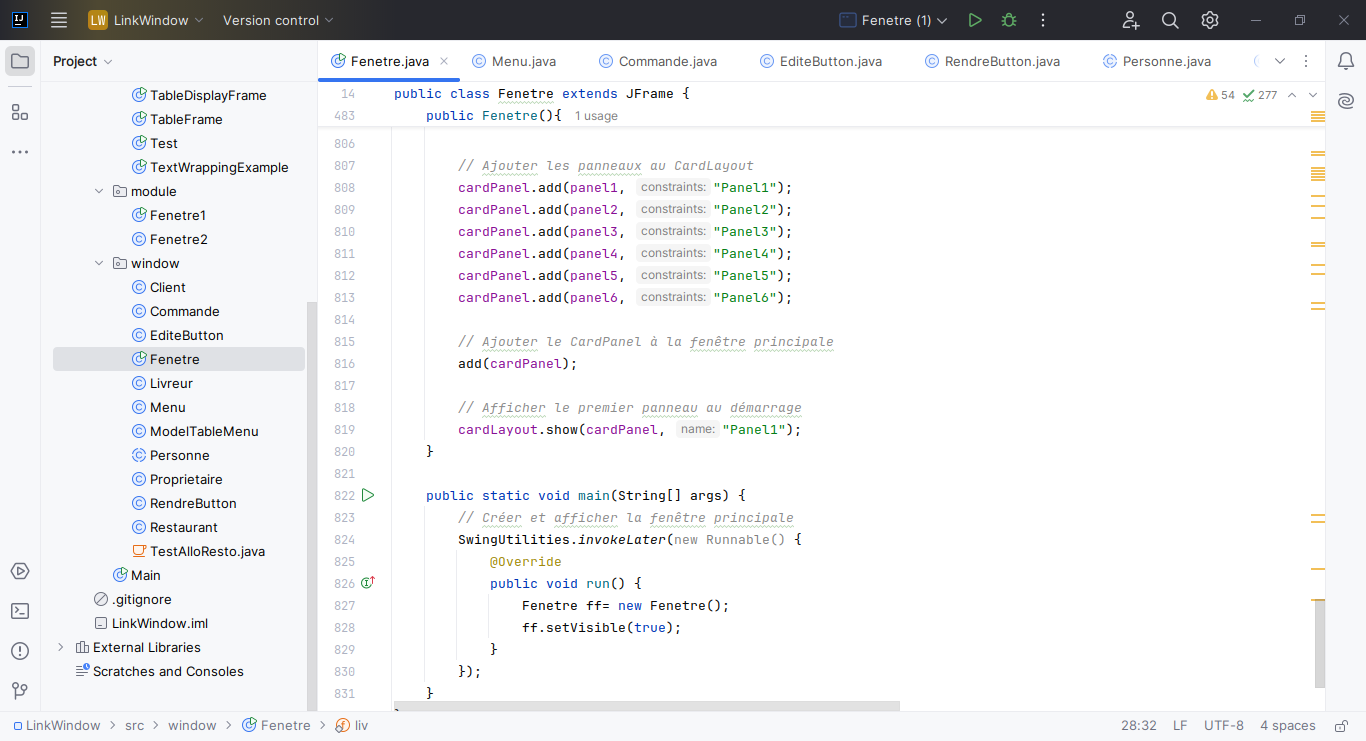












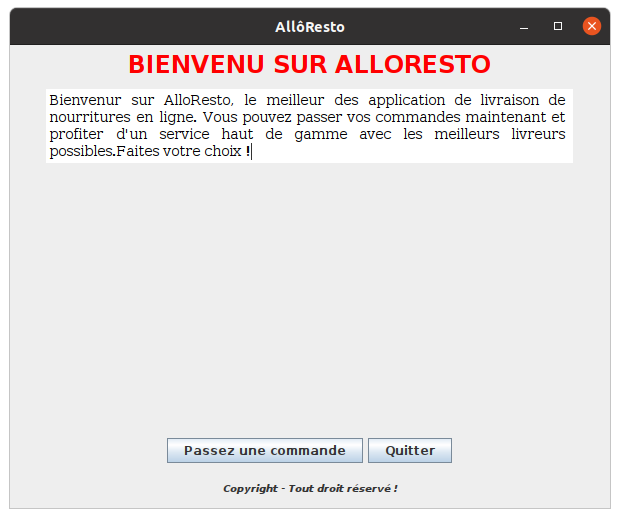
NB : les codes sources des fichiers EditeButton, RendreButton et ModelTableMenu sont ci-joint au rapport.

**Résultats :**

Le projet développé est une aide à effectuer des commandes auprès des restaurants. Cette application met en relation des Personnes (Livreur, Client, Propriétaire), des Restaurants, des Menus de restaurant et des Commandes :

1. Compilation du projet :

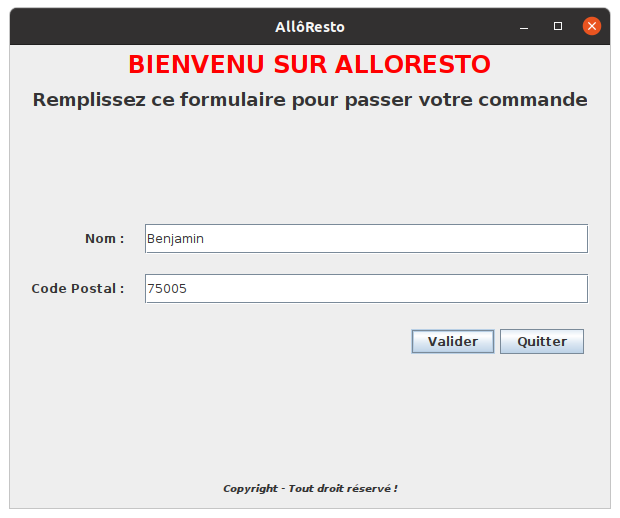
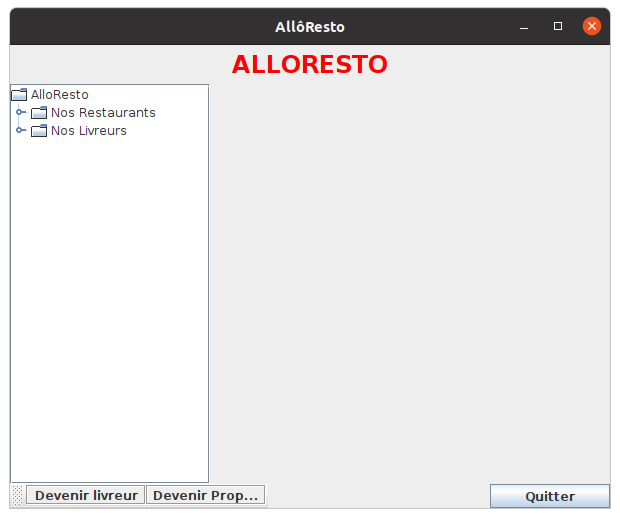
* Aperçu première page :



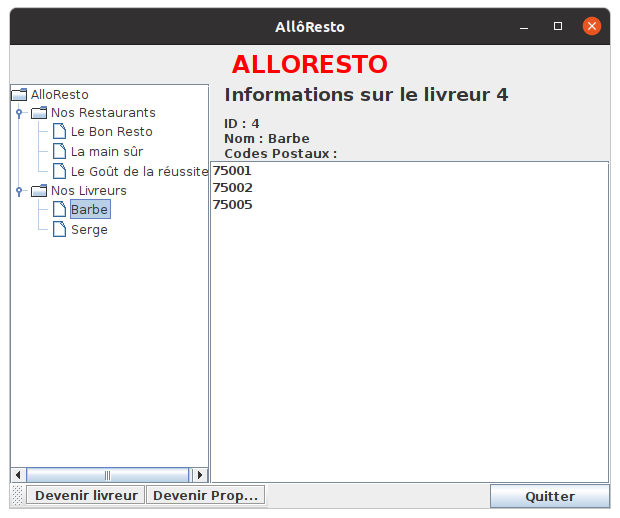
En cliquant sur « Quitter » la fenêtre se ferme et le programme s’arrête.

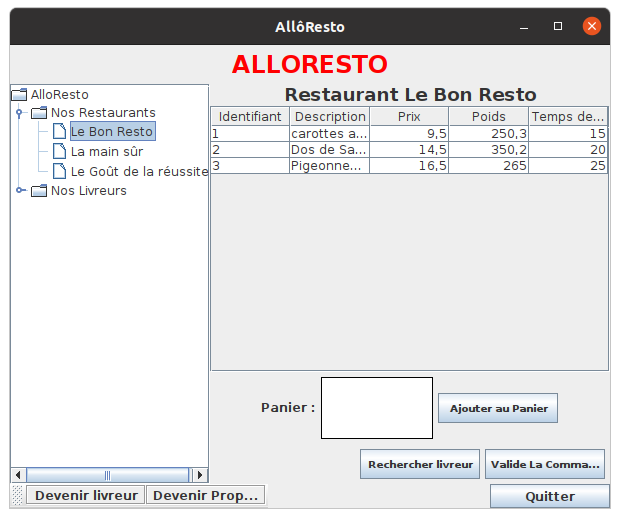
* Aperçu après clic sur le bouton Passez une commande :

A l’affichage de ce nouveau panneau, lorsqu’on clique sur « valider » un nouveau panneau est affiché :

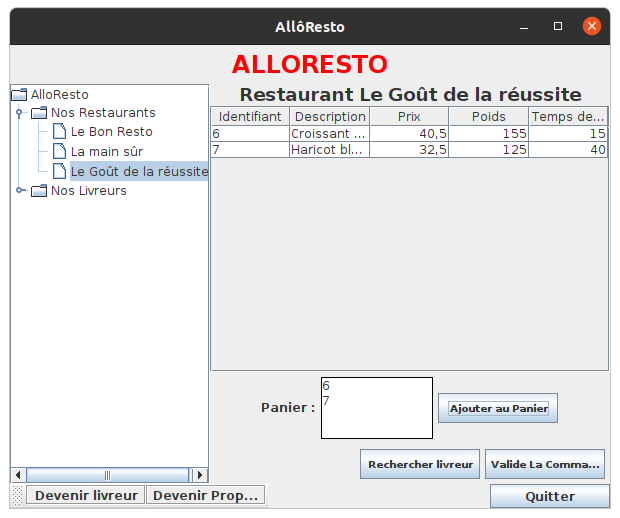


* Aperçu sur la liste des Restaurants et des Livreurs lorsqu’on clique sur Nos Restaurants ou Nos Livreurs :

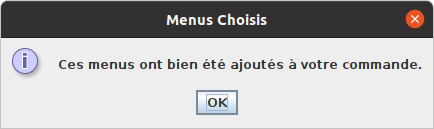




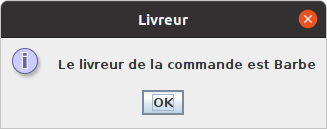
* Aperçu lorsque le Client passe une commande :
* Remplissage du panier par le client :



* Après avoir entré les id des menus choisis et cliquez sur « Ajouter au Panier » :

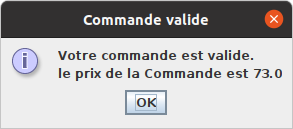


* Après avoir cliqué sur « Rechercher livreur » :

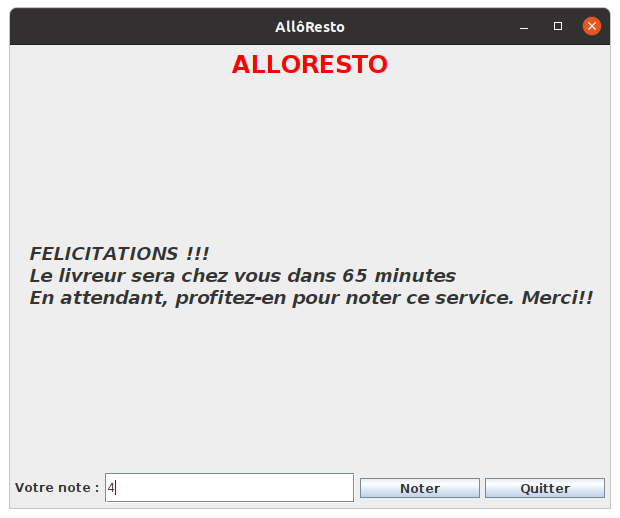


**NB** : le livreur est Barbe parce que les codes postaux de Barbe et le client se correspondent.

* **A**près avoir cliqué sur « Valide Commande » :

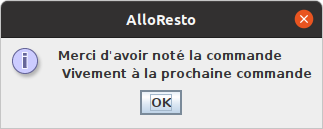


* Aperçu lorsque la commande s’est bien passée :
* Après avoir cliqué sur « Ok » de la dernière boite de dialogue :

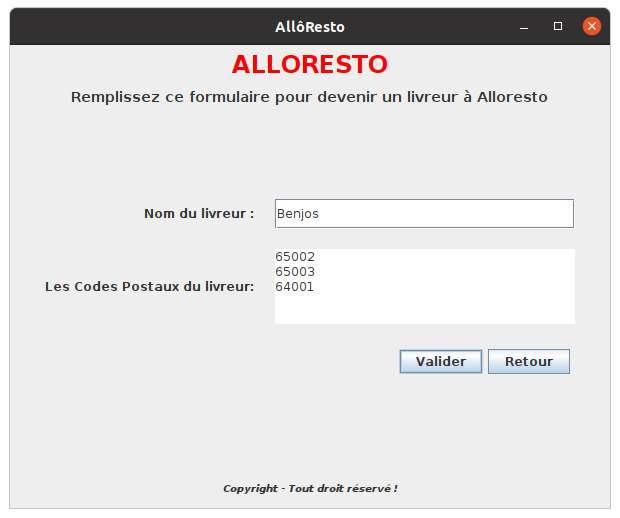


**NB** : Le temps de livraison est calculé directement et retourné dans le dernier panneau. Ici le temps de Livraison est de 65 minutes.

* Une possibilité est offerte au Client de noter la commande et le service assuré par la plateforme. Après remplissage du dernier champ par la note, le client clique « noter » et on a :



* Une clique sur le bouton « Quitter » peut fermer la fenêtre.
* Aperçu lorsque le client clique sur « Devenir Propriétaire » ou « Devenir Livreur » :
* Livreur :



* Propriétaire :



1. Analyse des résultats :

Les résultats obtenus satisfont bien aux attentes et objectifs de ce programme. Mais tout de même les ajustements personnels ajoutés au programme ont causé quelques problèmes et demandent encore une certaine maintenance.

Les erreurs non corrigées sont :

* L’ajout au panier des id des menus ne se fait que sur le dernier Restaurant de la liste des Restaurants ;
* La sélection d’un livreur se fait seulement sur le dernier Restaurant de la liste des Restaurants ;
* Nous notons également un problème de dispositions du code source de la classe principale non résolu ;

**Conclusion :**

Ce projet qui est axé sur la manipulation de la hiérarchie des classes en java a été un projet long et complexe et passionnant pour nous. Nous avons beaucoup de nouveaux grâce à ce projet. Nous notons entre autres :

* L’approfondissement des notions de redéfinition et d’héritage ;
* Création des modèles personnels de table en java ;
* Maîtrise de quelques composants graphiques en java tels que : JPanel, JLabel, JTextField, JTextArea et autres.
* Maîtrise des méthodes de positionnement des composants graphiques tels que le FLowLayout, le BorderLayout et le GridLayout ;

**NB : Ci-joint à ce rapport on a le fichier .zip du code source, un fichier exécutable de l’application**