Cahier de conception pour le carnet d’adresses

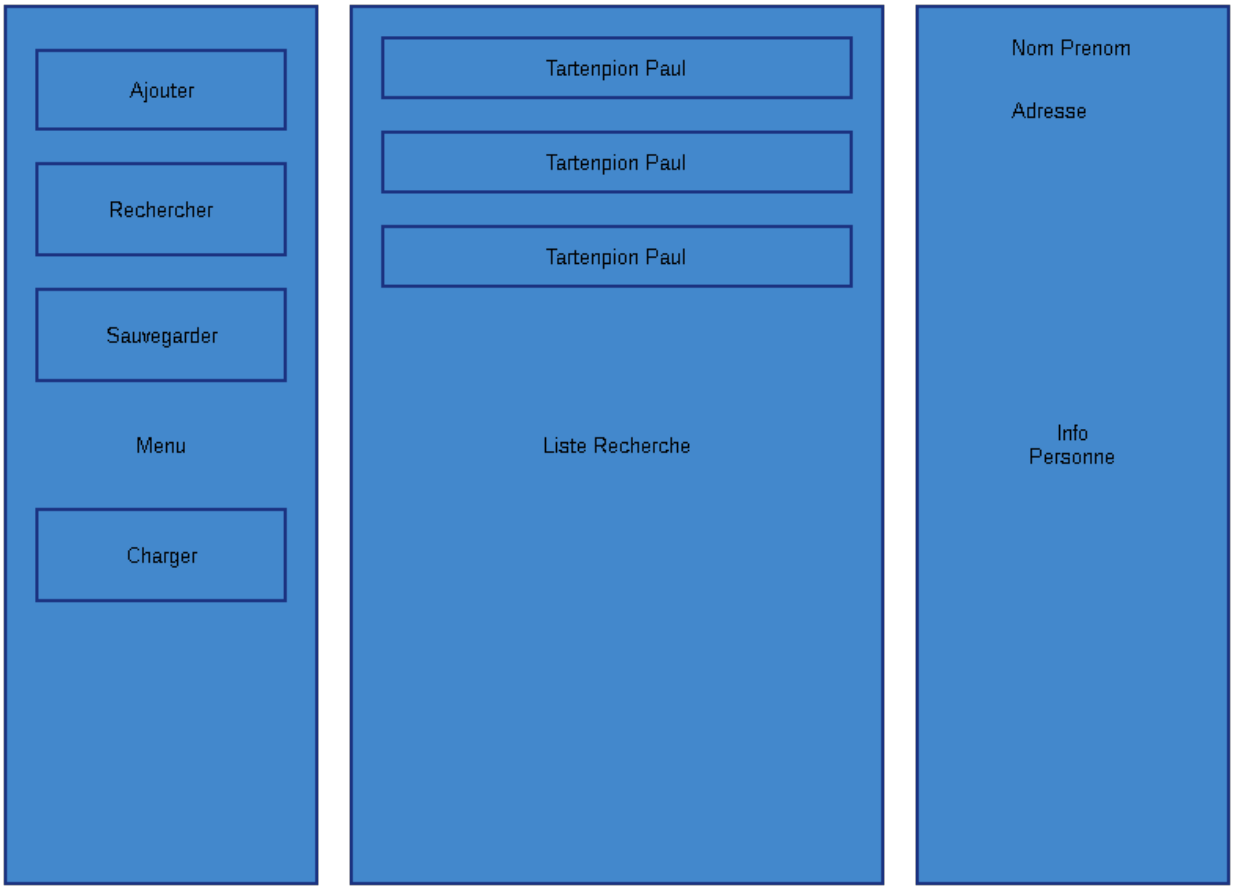
Auteur : Bastien Loubet, Michael Meunier et Antoine Rey

Historique des modifications :
V1 : Création du documents.
V2 : Ajout de la maquette de l’application.

Table des matières

1 Vue d’ensemble de l’application

1.1 Maquette de l’écran principal



Notre application se sépare visuellement en trois colonnes, qui correspondent à des objets java.swing.

1.2 Fonctionnalités implémentées

Le menu de la colonne de gauche qui propose nos quatre actions principales :

1. Ajouter un contact
2. Effectuer une recherche par nom
3. Sauvegarder le carnet
4. Charger le carnet

De plus l’interface propose deux fonctionnalités supplémentaires :

1. L’affichage des résultats de la recherche sous forme de boutons dans la colonne du centre

2. L’affichage des informations de la personne sélectionnée dans la liste de recherche. Ces informations s’affichent dans la colonne de droite.

2 Conception générale

2.1 Objectif

Élaborer un exemple complet de programme regroupant un maximum de pattern connus pour implémenter une conception SOLID.

2.2 Diagramme de classe complet

Le diagramme de classe complet de notre application suit. Les sections suivantes détaillent l’implémentation de toutes les fonctionnalités.

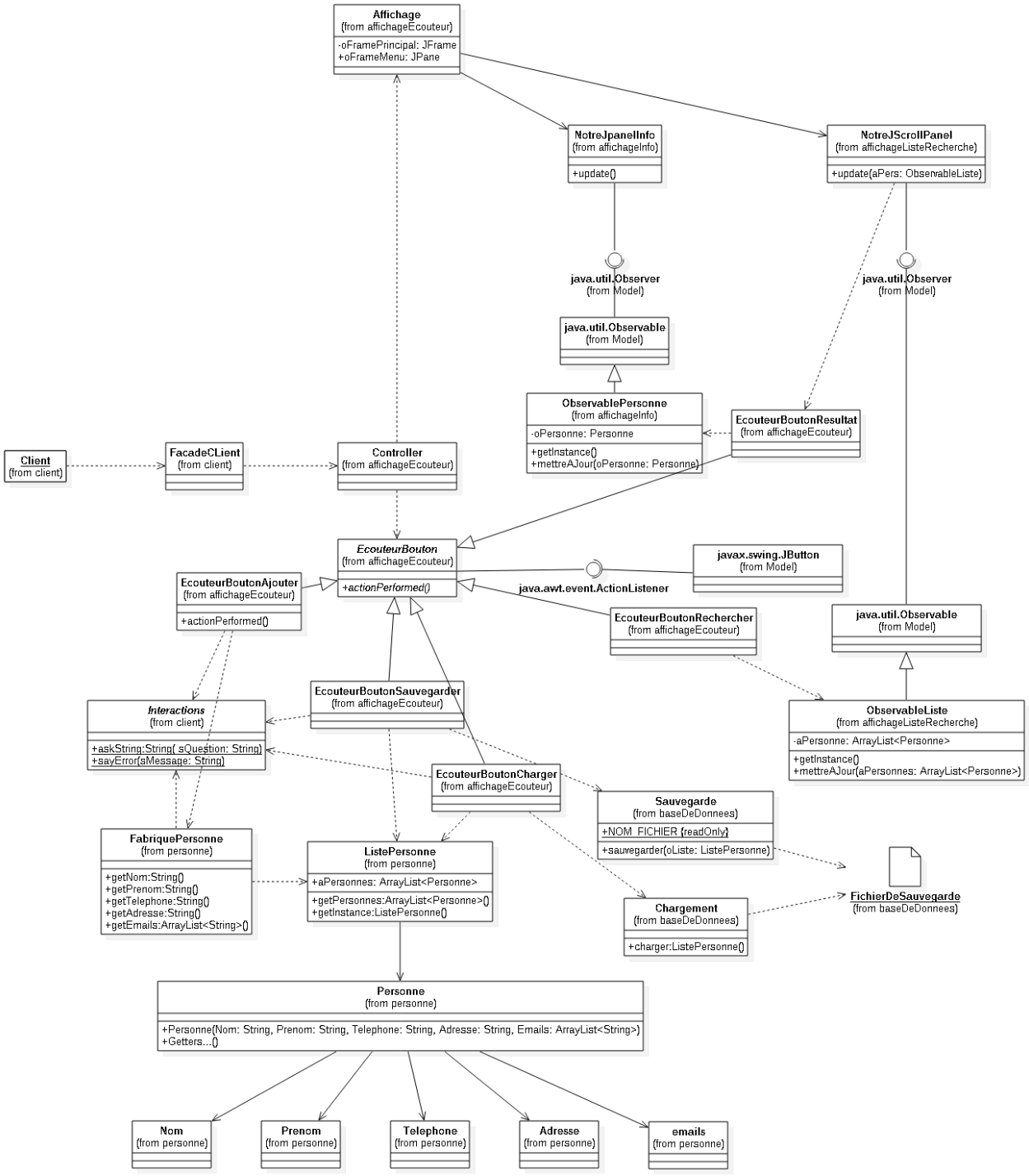
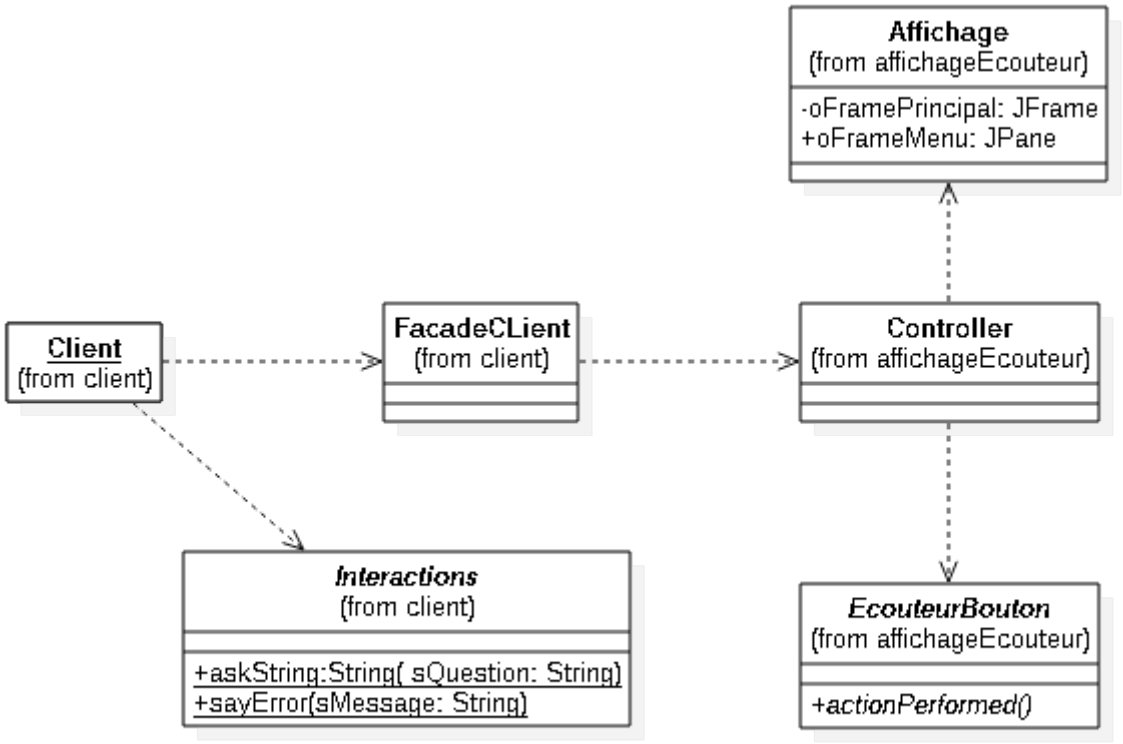


Illustration 1: Le diagramme de classe complet

3 Initialisation

3.1 Diagramme de classe



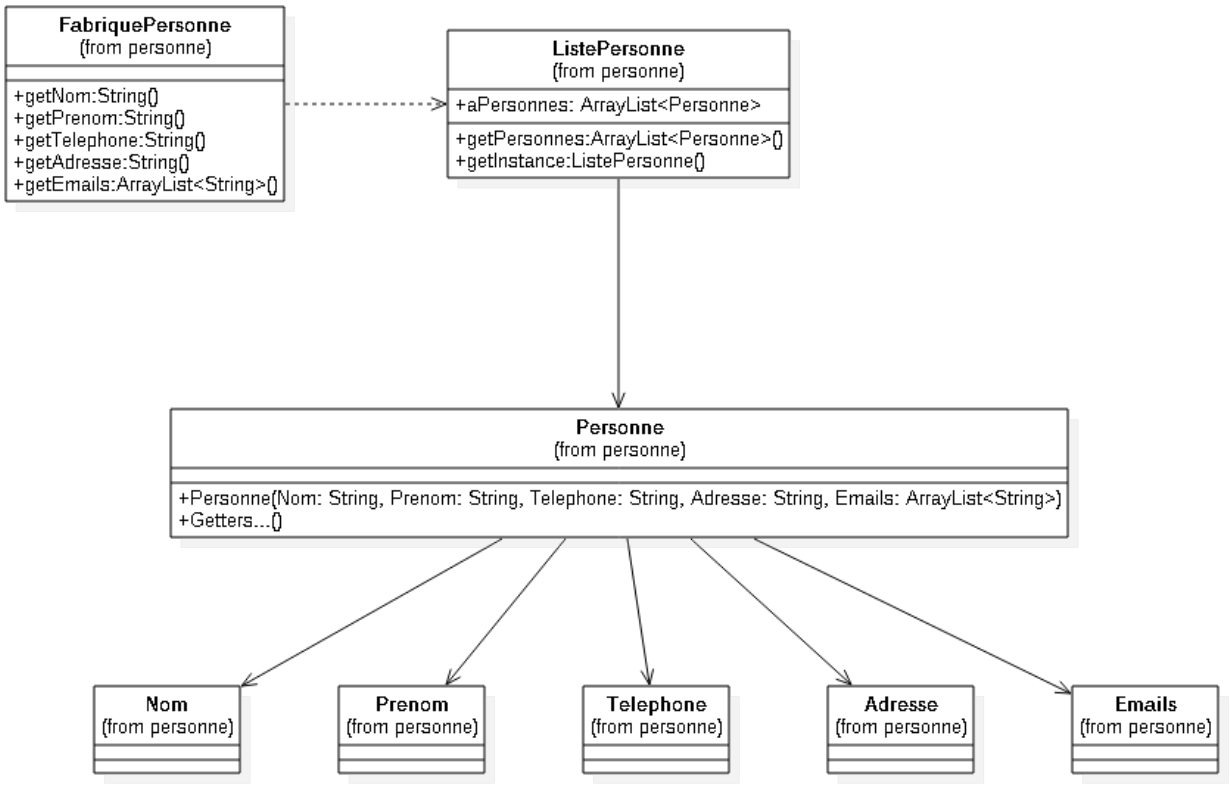
3.2 Responsabilité

Les objets ont les responsabilités suivantes :

1. Le client lance la Façade client qui a pour responsabilité de lancer l'application et d'interagir avec le client en général.
2. Le controller est initialisé par la façade client. Cet objet a la responsabilité de lancer l'initialisation de l'affichage et de faire le lien entre les observer et les observables de l'interface (les boutons et les panneaux).
3. L'affichage a pour responsabilité de gérer la création, le stockage et l'accès aux objets java swing qui composent l'interface.
4. La classe statique Interactions gère l'interaction avec l'utilisateur. Cette classe affiche des boites de dialogues pour demander une chaîne de caractère à l'utilisateur ou pour afficher des messages d'erreurs ou d'informations.

4 Gestion de la liste de contact

4.1 Diagramme de classe

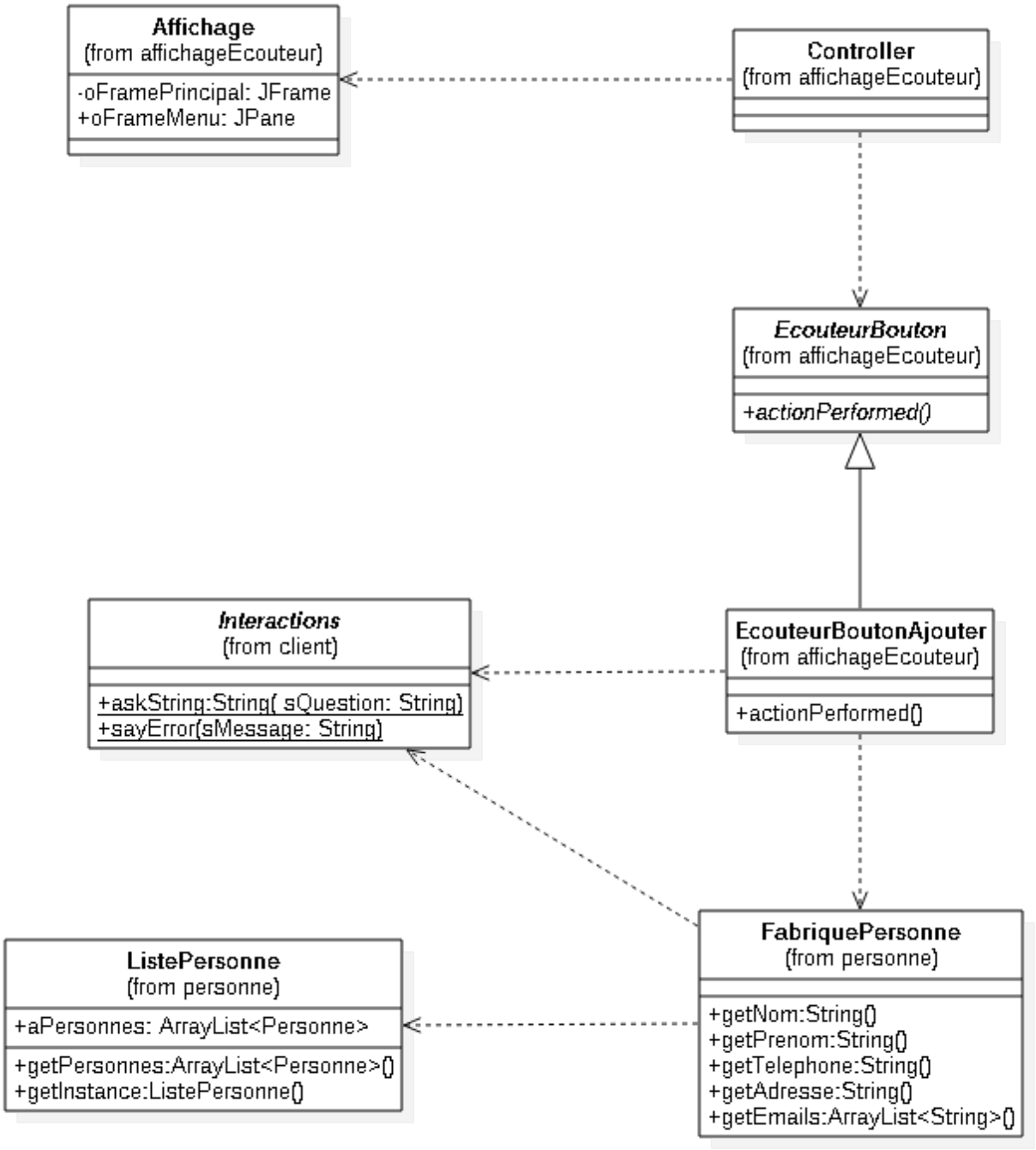


4.2 Responsabilité

1. Les objets Nom, Prenom, Telephone, Adresse et Emails composent l'objet Personne qui contient les informations sur un contact.
2. La ListePersonne est un singleton accessible de toute l'application qui stocke la liste de tous les contacts enregistrés.
3. La FabriquePersonne est responsable de la création et de l'ajout d'une personne à la liste personne. Cette classe demande les informations à l'utilisateur et vérifie ensuite le format des entrées. Si une entrée n'est pas valide alors la création d'un contact échoue.

5 Ajout d’un contact

5.1 Diagramme de classe

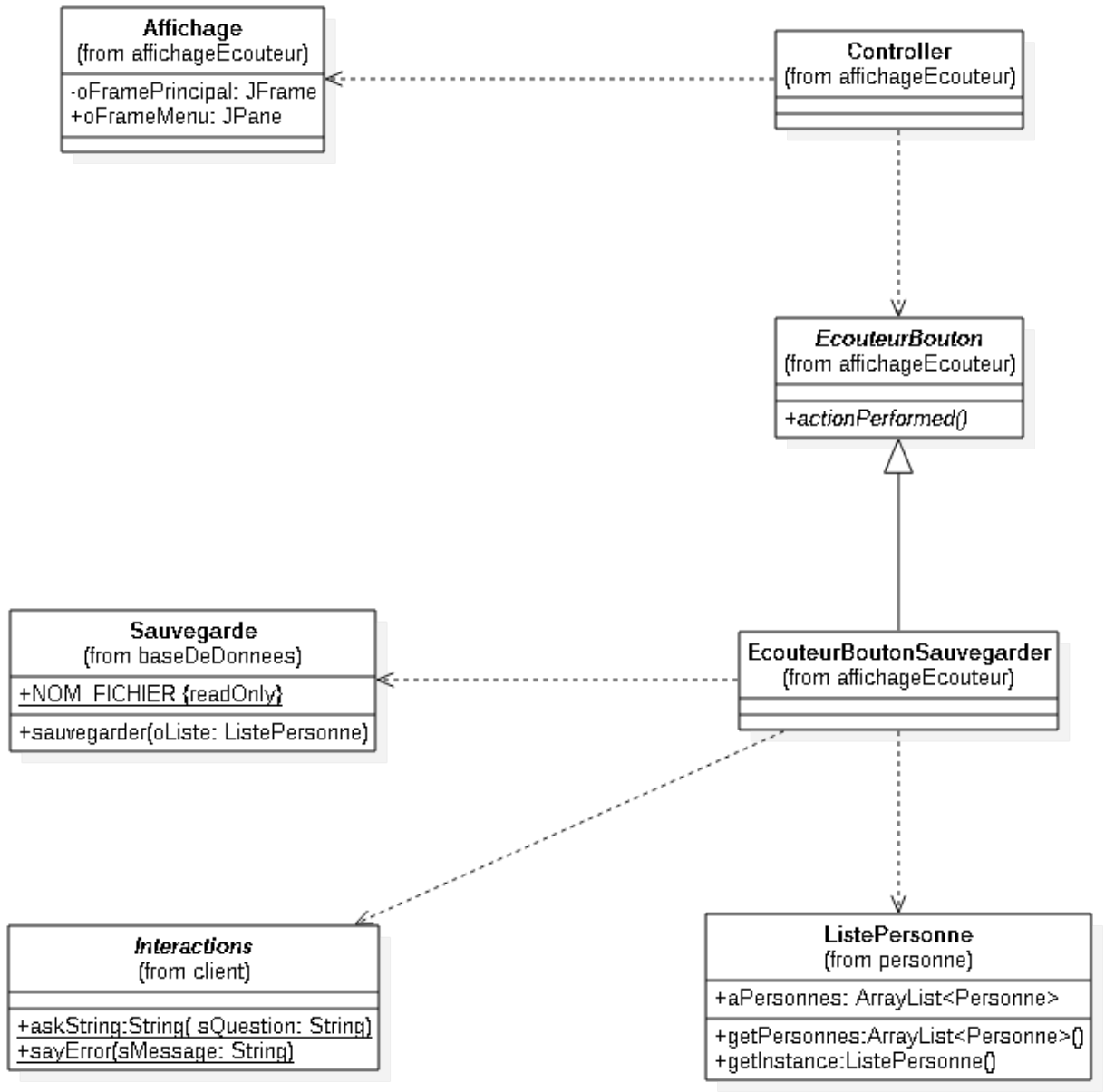


5.2 Responsabilité

1. La classe EcouteurBoutonAjouter implémente l’action lors du clique du bouton Ajouter grâce au pattern observer. Elle appel la classe FabriquePersonne pour ajouter une personne dans la liste personne et informe l’utilisateur du succès ou de l’échec de l’opération à travers la classe Interactions.

6 Sauvegarde

6.1 Diagramme de classe

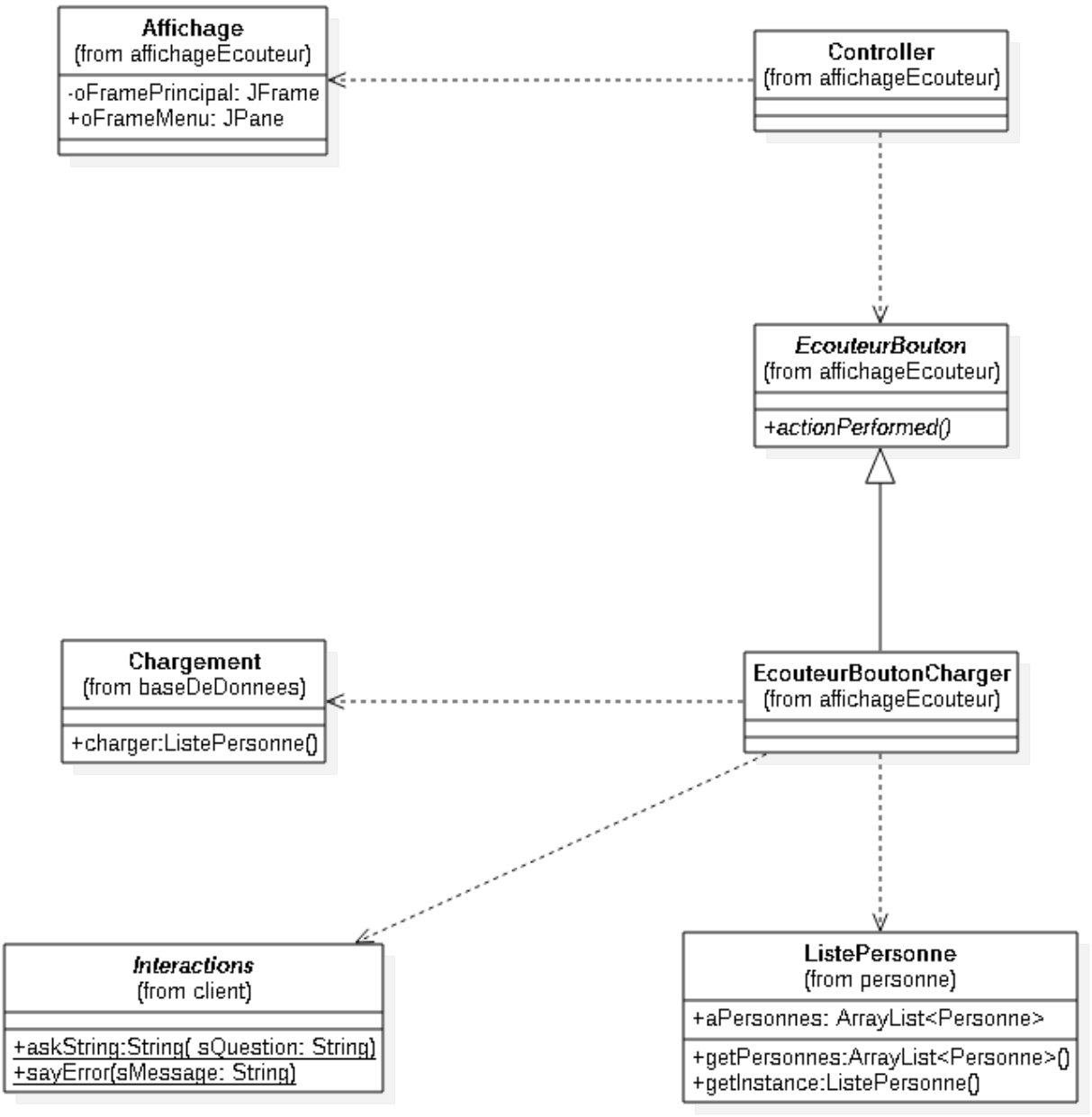


6.2 Responsabilité

1. La classe sauvegarde est responsable de la sauvegarde des données fournies dans un fichier. Elle stocke aussi le nom du fichier de sauvegarde dans une constante.
2. La classe EcouteurBoutonSauvegarder est responsable de l'action qui arrive quand l'utilisateur clique sur le bouton Sauvegarder. Cette fonction fournie les données de la ListePersonne à sauvegarder à l'objet Sauvegarde. Elle informe l'utilisateur du succès ou de l'échec de l'opération.

7 Chargement

7.1 Diagramme de classe

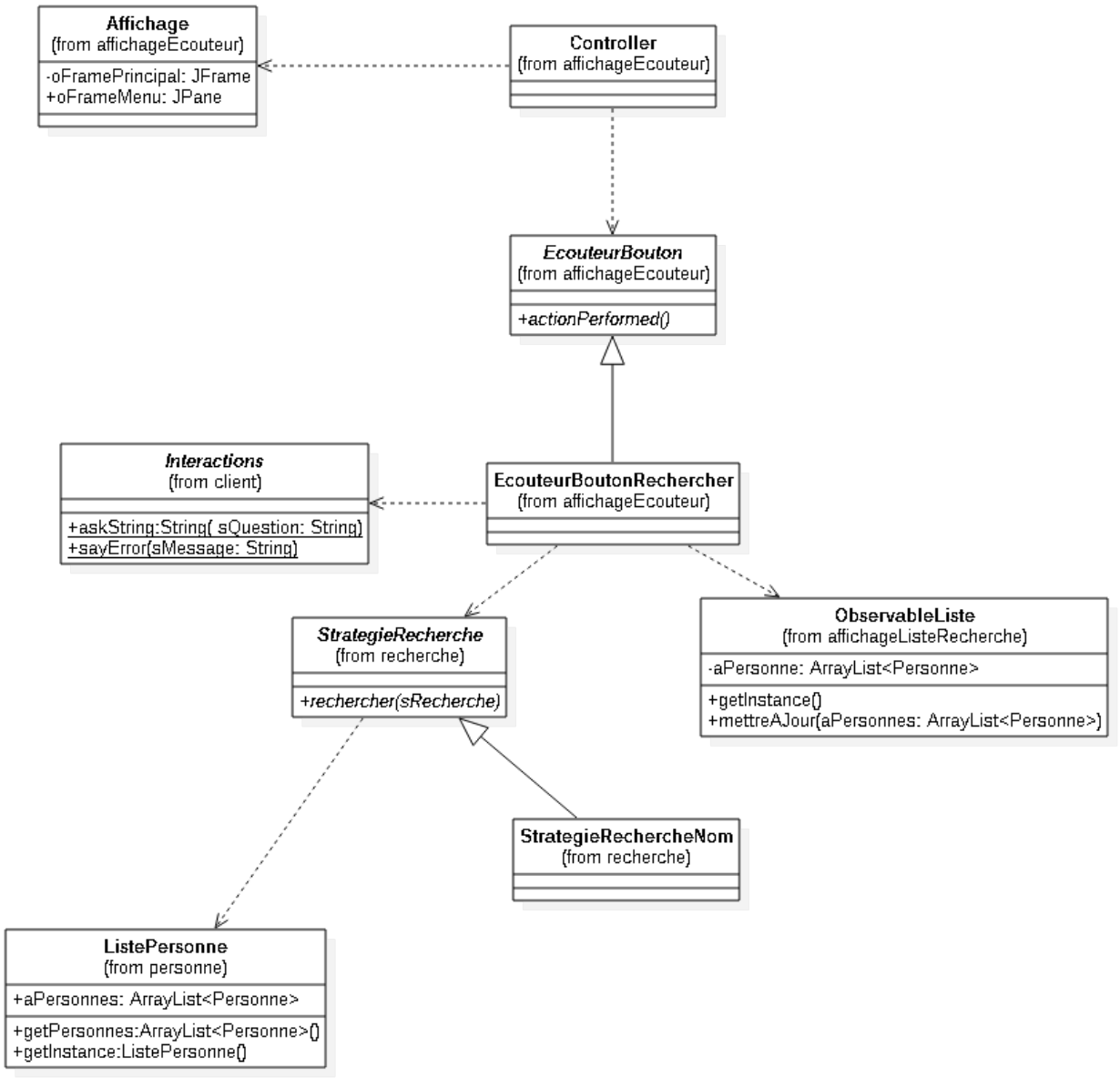


7.2 Responsabilité

3. La classe Chargement est responsable du chargement des données à partir d'un fichier.
4. La classe EcouteurBoutonSauvegarder est responsable de l'action qui arrive quand l'utilisateur clique sur le bouton Charger. Cette fonction entre les données fournies par l'objet Chargement dans l'objet ListePersonne. Elle informe l'utilisateur du succès ou de l'échec de l'opération.

8 Recherche par nom

8.1 Diagramme de classe



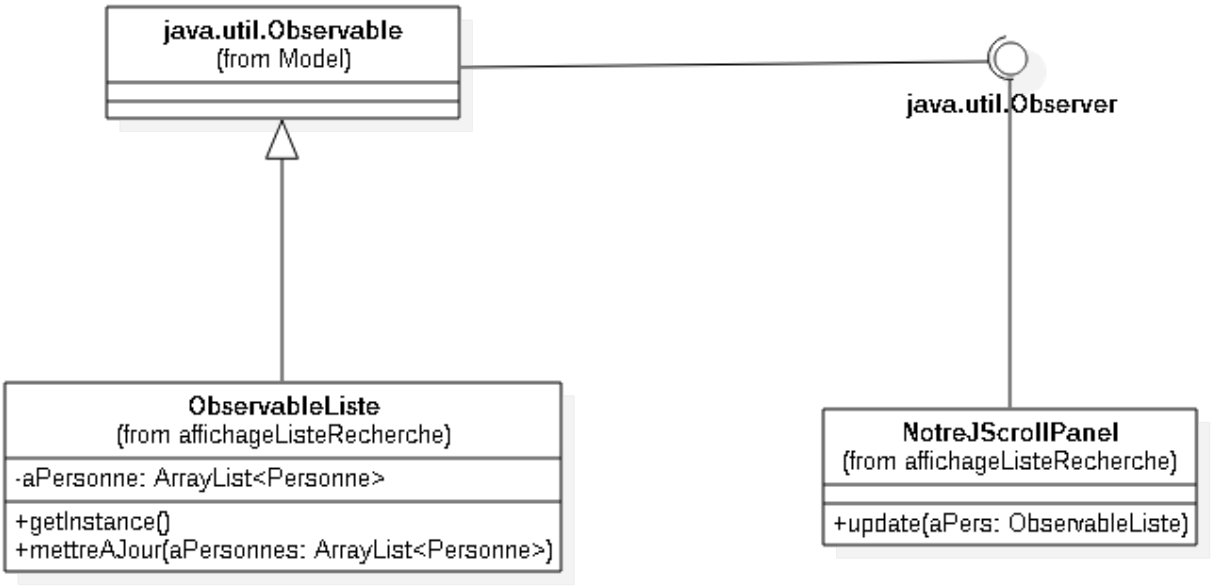
8.2 Responsabilité

1. La classe EcouteurBoutonRechercher est responsable de l'action qui arrive quand l'utilisateur clique sur le bouton Rechercher. La fonction demande la chaîne de recherche à l'utilisateur puis lance la recherche à travers un objet StrategieRechercheNom. Finalement la fonction met à jour la classe ObservableListe avec les résultats de la recherche.

2. La classe SrategieRechercheNom est responsable de la recherche par nom dans la ListePersonne. Elle prend un regex en entrée et retourne la liste des Personnes dont le nom correspond.
3. La classe ObservableListe stocke les résultats de la recherche et notifie ses observer qu'elle a été mise à jour. Cet objet est un singleton accessible partout dans le code.

9 Affichage des résultats de la recherche

9.1 Diagramme de classe

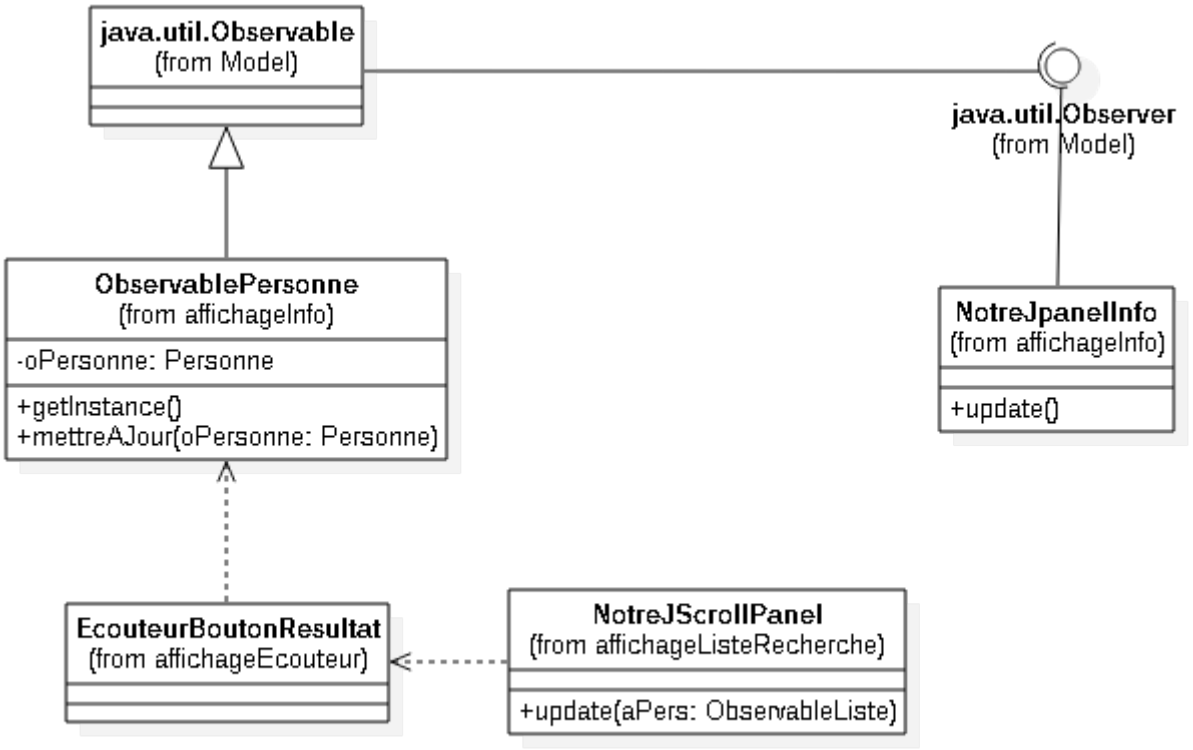


9.2 Responsabilité

1. La classe ObservableListe notifie l'objet NotreJScrollPanel de sa mise à jour grâce à un pattern observer. On rappel que c'est la classe Controller qui est responsable de l'enregistrement de l'ajout de l'observer tandis que la classe Affichage stocke l'objet NotreJScrollPanel.
2. L'objet NotreJScrollPanel se met à jour à partir de l'ObservableListe et créé, affiche et stocke les boutons pour l'affichage de chaque Personne présente dans l'observable liste.

10 Affichage des détails de la Personne sélectionnée dans la liste de recherche

10.1 Diagramme de classe



10.2 Responsabilité

1. L'objet NotreJScrollPanel s'occupe de lier les boutons qu'il créé dynamiquement avec l'EcouteurBouttonResultat.
2. La classe EcouteurBoutonResultat est responsable de l'action qui arrive quand l'utilisateur clique sur un des boutons des résultats de la recherche. A sa création elle stocke l'objet Personne correspondant et quand l'utilisateur clique le bouton la fonction met à jour l'objet ObservablePersonne avec l'objet Personne stocké.
3. La classe Observable personne est responsable du stockage de la Personne dont les informations sont affichées sur le cadran d'information. C'est un singleton accessible depuis toute l'application. Quand cet objet est mis à jour il en notifie ses observer (encore une fois configurer dans l'objet Controller).
4. NotreJPanelInfo observe l'objet ObservablePersonne et des que celui ci le notifie il mets à jour les informations du pannel pour correspondre aux informations de l'objet stocké par ObservablePersonne.