ScholarProjects
rapport d'experimentation

Programmation Objet avancée

Léa Cohen-Solal
Melvin Guirchoun
Dirigé par ZELLAMA KHADDOUJA

1 Interface graphique

Pour créer notre interface graphique, nous avons sélectionné Swing. Cette bibliothèque est assez simple et intuitive à utiliser, même si elle peut sembler complexe au premier abord. Elle a été très efficace pour réaliser une interface claire, lisible et qui semble professionnelle.

Nous avons également décidé de ne pas utiliser la gestion automatique de l'agencement des composants de l'interface. Cela nous a permis de les positionner exactement comme nous le voulions, bien que cela ait nécessité un effort supplémentaire pour aligner correctement les éléments à la fois horizontalement et verticalement, tout en veillant à ne pas dépasser les limites de la fenêtre.
En y repensant, il est possible que nous aurions pu bénéficier d'une disposition automatique des composants. (voir rapport dev)

2 Base de données

Nous avons décidé de ne pas utiliser de base de données SQL pour gérer nos données. À la place, nous avons choisi d'utiliser une base locale composée d'ArrayLists et de HashMaps. Cette approche nous a permis de modifier facilement notre "base de données" durant l'exécution, grâce aux bibliothèques Java intégrées comme ArrayList et HashMap.

Pour la sauvegarde de nos données, nous avons initialement envisagé d'utiliser des fichiers .json, mais nous avons finalement opté pour des fichiers .bin, qui se sont révélés plus pratiques. Chaque élément clé à sauvegarder, comme la liste des projets et de leurs binômes, la liste des formations et des cours, ainsi que les informations sur le professeur utilisant l'interface, est stocké dans un fichier séparé.

Au début de chaque exécution, nous récupérons les données sauvegardées en utilisant la méthode read() de la classe ReadObject. En fin d'exécution, les données éventuellement modifiées sont sauvegardées à l'aide de la méthode write() de la classe WriteObject.
Cette méthode s'avère très efficace et assure une restitution précise des données d'une exécution à l'autre.
Cependant, nous avons rencontré un problème spécifique : pour quitter l'application, l'utilisateur doit impérativement utiliser le bouton "exit" prévu à cet effet. Si l'utilisateur ferme l'application en utilisant le bouton rouge de fermeture standard, les données ne sont pas sauvegardées correctement.
Nous avons bien précisé ce point important dans le rapport user.

3 Gestion des exceptions

Nous avons opté pour la gestion des exceptions en utilisant la structure 'try-catch'. Cette approche a été principalement mise en œuvre pour contrôler le format des dates et des notes saisies par les enseignants, ainsi que pour vérifier

que certaines cases obligatoires ne soient pas laissées vides. De plus, nous avons utilisé certaines exceptions intégrées à Java, en particulier lors de l'écriture et de la lecture de fichiers .bin.

4 IDE

Nous avons initialement commencé à coder notre projet avec VSCode, mais lorsque nous avons eu besoin de travailler avec des fichiers .json, il s'est avéré nécessaire de télécharger des extensions. Nous avons constaté que la gestion des extensions était plus facile avec Eclipse qu'avec VSCode, ce qui nous a incités à changer d'IDE.