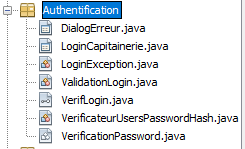
# Explication et code correspondant à un login



Voici mon package contenant les fenêtres « Dialog » et les différentes classes/interfaces utiles au Login.

Lorsqu’un utilisateur démarre l’application, il se retrouve sur la Capitainerie (inutile car la plupart des boutons sont disable). Seulement le bouton « Se connecter » et « Format Date » sont actifs. L’utilisateur, lorsqu’il veut se connecter, doit appuyer sur le bouton « Se connecter ». Une fenêtre modale s’ouvre avec deux champs (Login et Password). L’utilisateur rentre ses données de login et clique sur le bouton de connexion.



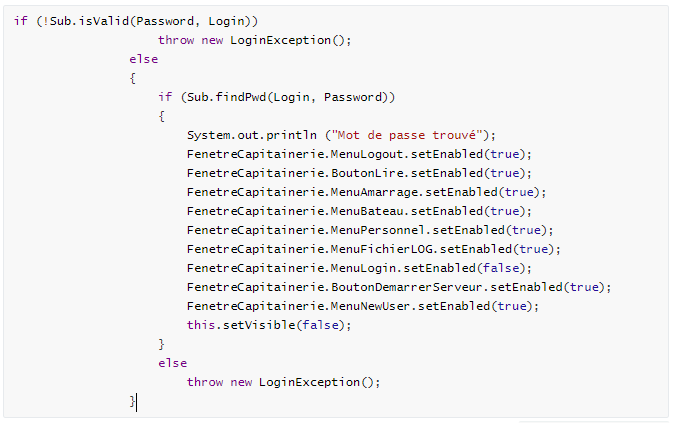
Je récupère son log et son pwd, je vérifie si le texte du bouton de connexion est bien en « Se connecter » (voir plus tard). Si oui, je crée un objet « VerificateurUsersPasswordHash » avec son Login en paramètre.



Voici le constructeur de l’objet. Je vais tout d’abord vérifier si le répertoire « Users » existe (s’il y déjà un utilisateur d’encodé). Si le répertoire n’existe pas, je vais alors en créer un et y ajouter un premier utilisateur. Les données de l’utilisateur seront son login et son mdp, stockés sous la forme d’une hashmap (clé + valeur). La clé sera son nom d’utilisateur et la valeur de cette clé, son mot de passe.



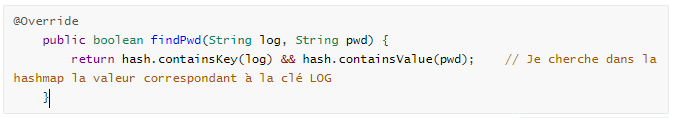
Si un répertoire « Users » existe déjà, je vais alors vérifier si un fichier portant le nom « users+.data » existe dans ce même répertoire. Si oui, je l’ouvre et stocke sa valeur dans une Hashmap (this.hash).



Si aucune exception n’a été déclenchée, je vais alors vérifier si le mot de passe rentré par l’utilisateur est valide (pas de caractères bizarres, une longueur de login raisonnable, etc.).



Si les données sont valides, je vais maintenant comparer les deux hashmaps pour vérifier la correspondance des mots de passe.



Si une exception est déclenchée comme le fichier user qui n’existe pas, le login pas valide, le mot de passe pas valide -> Je lance une LoginException qui va créer une fenêtre Dialog.

