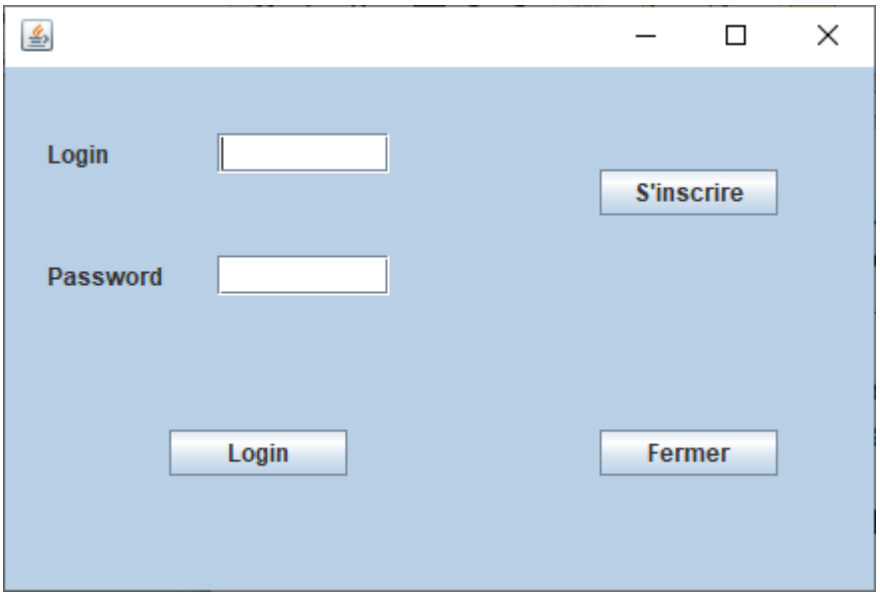
TP-TD N°7 : Les Entrées/Sorties EN JAVA

Dans ce TP nous souhaitons développer une application Java avec interface graphique qui assure l'authentification des utilisateurs. En outre, cette application permet aux nouveaux utilisateurs de s'inscrire pour pouvoir s'authentifier.

Code source Java de cette application se compose de :

- Account.java : la classe Account (chaque objet de cette classe représente un compte d'un utilisateur).
- Collectionaccount : la classe collection d'objets de type Account.
- InterfaceAuthentification.java (fenêtre principale de l'application).



- InterfaceInscription.java (fenêtre pour s'inscrire : créer un nouveau compte).

N.B :

Les interfaces de cette application seront créées à l'aide de WindowBuilder.

- **Lien pour installer windowbuilder**
  https://www.eclipse.org/windowbuilder/download.php

- **lien utile (vidéo) pour savoir comment créer des interfaces graphiques à l'aide de WindowBuilder :** https://www.eclipse.org/windowbuilder/

Questions :

1. Créer un nouveau projet java nommé Application.
2. Dedans ce projet java, créer la classe Account et Collectionaccount.
3. Créer les interfaces graphiques à l'aide de WindowBuilder.
4. Pour assurer la persistance de données, un fichier collection.txt sera créé pour sauvegarder les comptes.
5. Créer les interfaces mentionnées ci-dessus à l'aide de WindowBuilder.
6. Tester l'application.

A lire attentivement avant de commencer le TP :

- Pour pouvoir stocker les comptes des utilisateurs de façon permanente, on procède comme suit :
  - Lors de la création d'un nouveau compte, un nouvel objet de la classe Account sera instancié. Ensuite, sera ajouté dedans la collection de comptes. A la fin, la collection sera sauvegardée sur un fichier. (La sérialisation de la collection)
- Pour pouvoir lire les comptes des utilisateurs à partir du fichier, on procède comme suit :
  - Lors de l'authentification, le fichier sera lu pour récupérer la collection de comptes (désérialisation de la collection de comptes) afin de vérifier l'existence de votre login et password dedans la collection de comptes.

Programmation événementielle :

Les événements sont des actions ou des effets indirects d'action de l'utilisateur du GUI (Interface Graphique Utilisateur) : clic de souris, frappe de touche, fenêtre masquée par une autre, ...qu'il faut gérer en mettant en place un "écouteur" de l'événement souhaité sur le composant graphique considéré, puis définir le comportement (les instructions à exécuter, le handler d'événement) lorsque l'événement surviendra.

- ActionListener est une interface pour écouter et traiter les ActionEvents

Par exemple, bouton.addActionListener(composant) met un écouteur (d'ActionEvent) sur le bouton, le "handler" sera assurée au niveau du composant désigné qui implémente l'ActionListener; pour un bouton,l'ActionEvent est provoquée par un clic de souris (bouton enfoncé puis relaché).

- removeActionListener(ActionListener écouteur) supprime un "écouteur" d'événement

La classe qui implémente l'interface ActionListener doit définir la méthode :

- public void actionPerformed(ActionEvent e) { ...} qui est le "handler".
- getActionCommand() est une méthode de ActionEvent qui récupère le nom de la commande d'action associé au composant par exemple un bouton (par défaut son label).

Exemple : quitter l'application (bouton quitter)

```
JButton Quitter = new JButton("quitter"); // l'objet bouton quitter
        Quitter.addActionListener(new ActionListener() { // ajout d'un écouteur au bouton
        public void actionPerformed(ActionEvent e) { // méthode à exécuter au moment du clic sur le
bouton quiiter
                dispose();      //méthode pour fermer la fenêtre principale de l'application
                        }
                });
```