Rq : si votre chargé de TP vous demande de rendre votre travail n'oubliez pas de le faire.

## Exercice 1

1. Écrivez une classe `Utilisateur` ayant les attributs suivants :
   — `private String pseudo;`
   — `private String motDePasse;`
   — `private final String adresseMail;`

2. Ajoutez un constructeur qui prend en arguments toutes les données nécessaires.
3. On voudrait que le mot de passe reste inaccessible en dehors de la classe `Utilisateur`, mais permettre sa vérification, ainsi que sa modification si l'on est capable de fournir l'ancien mot de passe. Écrivez une méthode booléenne `testMotDePasse` qui prend en argument une chaîne de caractères et teste si elle est identique au mot de passe. Écrivez ensuite une méthode `changerMotDePasse(String motdePasse, String nouveauMotDePasse)` qui change le mot de passe uniquement si le premier argument est identique au mot de passe actuel.
4. Écrivez des méthodes `getPseudonyme` et `setPseudonyme` avec des précautions d'utilisation réalistes.
5. Dans une classe `Test`, testez les méthodes que vous avez écrites.

## Exercice 2

Dans un contexte ou l'on souhaite décrire une messagerie entre utilisateurs, on s'intéresse d'abord aux messages. Il s'agit de textes signés par un seul utilisateur, écrits en une et une seule fois. Proposez une classe qui permette de définir un message, avec tout ce qui est nécessaire pour pouvoir l'utiliser dans votre classe `Test`. Vérifiez que tout fonctionne.

## Exercice 3

Dans un salon de discussion, les messages sont visibles par tous les participants. Il regroupe donc des utilisateurs et des messages, en nombre variable.

1. On souhaite utiliser des tableaux comme attributs d'une classe `Salon` pour stocker les utilisateurs, et les messages dans l'ordre dans lequel ils ont été émis. Définissez la classe `Salon` avec deux attributs tableaux `tabUtilisateur`, `tabMessage` et ajoutez deux attributs entier `indexLibreMessage`, `indexLibreUtilisateur`. Vous construirez un salon en réservant 10 références dans vos tableaux, et en initialisant à 0 les deux attributs entiers.
2. Écrivez une méthode `estPresent` qui prend en argument un utilisateur et teste s'il est présent dans le salon.
3. Écrivez une méthode `ajouterUtilisateur` qui prend en argument un utilisateur et l'ajoute au salon s'il reste de la place. (Vous l'ajouterez à la place repérée comme étant libre, puis incrémenterez l'index)
4. Écrivez une méthode `ajouterMessage` qui prend en argument un message, vérifie que son auteur est présent parmi les utilisateurs, et l'ajoute à la séquence des messages s'il reste de la place. Vous ne ferez rien si l'une des conditions n'est pas remplie, mais retournerez un message adéquat.

5. Écrivez une méthode `afficher` qui affiche l'historique des messages et de leurs auteurs sous la forme :

   ```
   Joe: Bonjour à tous !
   Jack: Bonjour à toi.
   Alan: Comment allez-vous ?
   ```

6. Testez les fonctionnalités précédentes.
7. (plus difficile) On souhaite pouvoir exclure un utilisateur d'un salon : la personne en question ne fait plus partie du salon, et aucun de ses messages n'est plus conservé. Proposez une/des méthodes pour implémenter cette fonctionnalité proprement, tout en maintenant la cohérence de la signification des index tels que nous les avons utilisés.

## Exercice 4 (Si vous avez le temps)

1. Créez une classe `Chat` qui regroupe plusieurs `Salon`. Elle sera munie d'un constructeur sans argument, et d'une méthode `ajouterSalon`. (Vous travaillerez comme précédemment avec un tableau couplé à un index)
2. Écrivez une méthode `estPresent` qui prend en argument un utilisateur et teste s'il est présent dans un des salons du chat.
3. Écrivez une méthode `nombreMessages` qui prend en argument un utilisateur et renvoie le nombre total de messages qu'il a envoyés sur les salons. N'hésitez pas à écrire des méthodes intermédiaires pour conserver un code lisible.
4. Écrivez une méthode `bavard` qui donne l'utilisateur qui a envoyé le plus de messages sur les salons.
5. Testez l'ensemble