Travail pratique #2

Ce travail doit être fait individuellement.

Notions mises en pratique : le respect d'un ensemble de spécifications, la conception et l'utilisation de méthodes (séparation fonctionnelle), et l'utilisation de la classe `String`.

Classe fournie (À NE PAS MODIFIER) à importer dans votre projet

Classe `TP2Util` : contient une méthode permettant d'obtenir la date courante sous le format : `jj/mm/aaaa hh:mm:ss`.

1. Spécifications

1.1 Description générale

Il s'agit de concevoir un petit prototype d'une application d'authentification qui permet de créer des comptes utilisateurs en fournissant un nom d'utilisateur et un mot de passe, et qui permet à un utilisateur de se connecter à son compte en fournissant ses informations de connexion (son nom d'utilisateur et son mot de passe).

1.2 Lancement du programme

Au démarrage, le programme affiche une petite présentation de l'application suivie d'un menu de 3 options, et attend que l'utilisateur entre son choix :

```
APPLICATION D'AUTHENTIFICATION - PROTOTYPE VERSION 1.0

====
MENU
====
1. Creer un compte
2. S'authentifier
3. Quitter

Entrez votre choix au menu :
```

Votre programme doit valider le choix au menu saisi par l'utilisateur. Les SEULES valeurs valides sont EXACTEMENT 1, 2 ou 3. Toute autre réponse doit être considérée comme invalide. Par exemple, la saisie de `11` ne doit pas être considérée valide, la saisie de seulement `<ENTRÉE>` ne doit pas être considérée valide, etc.

Note : au lancement du programme, il n'existe aucun compte.

Les sections suivantes décrivent chacune des 3 options du menu. Notez qu'à la fin de chaque option, on demande à l'utilisateur de taper `<ENTRÉE>` pour revenir au menu.

1.3 Option 1 du menu principal : Créer un compte

Cette option permet de créer un nouveau compte utilisateur. Au choix de cette option, le programme demande d'abord d'entrer le nom d'utilisateur pour ce nouveau compte. Votre programme doit valider le nom saisi par l'utilisateur. Un nom d'utilisateur valide doit :

- avoir entre 3 et 25 caractères
- ne contenir que des lettres (minuscules ou majuscules), des chiffres (0 - 9), le caractère *point* `(.)` ou le caractère *trait bas* `(_)`.

OU

- être la chaine vide.

Si le nom n'est pas égal à la chaine vide, le programme vérifie d'abord la longueur du nom entré, et si celle-ci n'est pas valide, il affiche un message d'erreur à cet effet, et redemande d'entrer le nom. Si la longueur du nom est valide, le programme vérifie si le nom est composé des caractères permis. Si ce n'est pas le cas, il affiche un message d'erreur (différent) à cet effet, et redemande d'entrer le nom. Le nom (non vide) est valide lorsqu'il respecte la contrainte de longueur, et qu'il ne contient que les caractères permis.

Si le nom est égal à la chaine vide (lorsque l'utilisateur saisit `<ENTRÉE>` seulement), le programme annule l'opération de création d'un nouveau compte (en affichant un message à cet effet).

Si l'opération n'est pas annulée (on a saisi un nom non vide), le programme vérifie s'il existe déjà un compte ayant le même nom d'utilisateur. Si c'est le cas, le programme affiche un message à cet effet et annule l'opération de création du nouveau compte (il ne peut pas exister deux comptes ayant le même nom d'utilisateur). Sinon, le programme demande alors d'entrer le mot de passe pour ce nouveau compte utilisateur. Votre programme doit valider le mot de passe saisi par l'utilisateur. Un mot de passe valide doit :

- avoir entre 8 et 16 caractères
- ne contenir que des lettres (minuscules ou majuscules), des chiffres (0 - 9), ou des caractères spéciaux parmi `*?#@!.+&`
- contenir obligatoirement au moins 2 lettres majuscules, au moins 1 chiffre, et au moins un caractère spécial parmi `*?#@!.+&`

OU

- être la chaine vide.

Si le mot de passe n'est pas égal à la chaine vide, le programme vérifie d'abord la longueur du mot de passe entré, et si celle-ci n'est pas valide, il affiche un message d'erreur à cet effet, et redemande d'entrer le mot de passe. Si la longueur de la chaine est valide, le programme vérifie si le mot de passe est composé des caractères permis. Si ce n'est pas le cas, il affiche un message d'erreur (différent) à cet effet, et redemande d'entrer le mot de passe. Finalement, si le mot de passe ne contient que des caractères valides, le programme vérifie s'il contient tous les caractères obligatoires. Si ce n'est pas le cas, il affiche un message d'erreur (différent) à cet effet, et redemande d'entrer le mot de passe.Le mot de passe (non vide) est valide lorsqu'il respecte la contrainte de longueur, qu'il ne contient que les caractères permis, et qu'il contient tous les caractères obligatoires.

Si le mot de passe est égal à la chaine vide (lorsque l'utilisateur saisit `<ENTRÉE>` seulement), le programme annule l'opération de création d'un nouveau compte (en affichant un message à cet effet).

Si l'opération n'a pas été annulée, le compte utilisateur est créé avec le nom d'utilisateur saisi (et valide) et le mot de passe saisi (et valide), et le programme affiche un message à cet effet.

=> Comme spécifications complémentaires, voir le fichier **exempleExec_1.txt** fourni avec l'énoncé du TP.

Note sur l'implémentation

Pour conserver en mémoire la liste de tous les comptes utilisateurs créés, vous **DEVEZ** utiliser une chaine de caractères (`String`), disons `infosComptes`, dans laquelle seront concaténées les informations de chacun des comptes. Pour chaque compte, on veut conserver le **nom d'utilisateur**, le **mot de passe**, et la **date** (`jj/mm/aaaa hh:mm:ss`) du dernier accès au compte, qui correspond au moment de la dernière connexion à ce compte utilisateur (voir option 2). Lorsqu'un nouveau compte est créé, il n'y a pas encore eu de connexion au compte, donc il n'y a pas de date du dernier accès. Voici un exemple de chaine qui contient 4 comptes :

```
"\n" +
"Bob.Bin|caL!Four6cHon|\n" +
"Pifou|Bar#Tendre24|06/03/2022 19:08:17\n" +
"Magnolia|volTAgeX!130|06/03/2022 19:11:25\n" +
"ver.ba_tim|vr0Um?Pow|\n"
```

Chaque ligne de la chaine `infosComptes` contient les informations d'un compte (séparées par le caractère `'|'`) : le nom d'utilisateur, suivi du mot de passe, et s'il y en a une, la date de dernier accès (sinon, il n'y a rien après le dernier caractère `'|'`). Dans la chaine ci-dessus, on voit que `Bob.Bin` et `ver.ba_tim` ne se sont jamais connectés à leur compte, contrairement à `Pifou` et `Magnolia` (on voit la date de leur dernier accès => 6 mars 2022 à 19h08 et 19h17).

Lors de l'ajout d'un nouveau compte, on concatène donc ses informations (nom d'utilisateur et mot de passe) sur une nouvelle ligne, à la fin de la chaine `infosComptes`, en conservant son format : `nomUtilisateur|motPasse|`. Un nouveau compte n'a jamais de date du dernier accès, on devra cependant l'ajouter ou la modifier chaque fois qu'un utilisateur se connecte à son compte en utilisant l'option 2.

Note : les caractères `'\n'` au début, et à la fin de la chaine `infosComptes`, ne sont pas obligatoires, mais facilitent le traitement.

1.4 Option 2 du menu principal : S'authentifier

Cette option permet de se connecter à un compte existant. Au choix de cette option, le programme demande d'abord d'entrer le **nom d'utilisateur**, puis le **mot de passe** du compte auquel on veut se connecter. Vous n'avez pas à valider ces deux entrées. Si le compte n'existe pas, le programme affiche un message à cet effet, et l'opération de connexion est annulée. Si le compte existe (le couple nom d'utilisateur et mot de passe existe dans la liste des comptes existants `infosComptes`), le programme entre dans le compte en affichant un mot de bienvenue contenant le nom de l'utilisateur de ce compte, la date du dernier accès à ce compte et finalement, un nouveau menu pour la gestion de ce compte (et le programme attend le choix de l'utilisateur au menu). Par exemple :

```
******************
* BIENVENUE TOTO *
******************

  Dernier acces : 09/03/2022 15:25:08



====
MENU
====
1. Modifier le mot de passe
2. Supprimer le compte
3. Se deconnecter

Entrez votre choix au menu :
```

Si c'est la première fois que l'utilisateur se connecte à son compte (et qu'il n'y a donc pas de date pour le dernier accès), la date est remplacée par un tiret "-". Par exemple :

```
******************
* BIENVENUE TOTO *
******************

  Dernier acces : -



====
MENU
====
1. Modifier le mot de passe
2. Supprimer le compte
3. Se deconnecter

Entrez votre choix au menu :
```

Votre programme doit valider le choix au menu saisi par l'utilisateur. Les SEULES valeurs valides sont EXACTEMENT 1, 2 ou 3. Toute autre réponse doit être considérée comme invalide. Par exemple, la saisie de `11` ne doit pas être considérée valide, la saisie de seulement `<ENTRÉE>` ne doit pas être considérée valide, etc.

Note sur l'implémentation

Aussitôt que la connexion au nouveau compte est effectuée, le programme récupère la date courante, et va l'insérer au bout de la ligne qui correspond à ce compte, dans la liste des comptes existants `infosComptes`. Ce sera cette date qui sera affichée lors de la prochaine connexion au compte. Par exemple, si avant la connexion, la liste des comptes est la suivante :

```
"\n" +
"Bob.Bin|caL!Four6cHon|\n" +
"Pifou|Bar#Tendre24|06/03/2022 19:08:17\n" +
"Magnolia|volTAgeX!130|06/03/2022 19:11:25\n" +
"ver.ba_tim|vr0Um?Pow|\n"
```

Et que la date de connexion à ajouter est `09/03/2022 15:43:15` pour le compte de `Bob.Bin`, après la connexion, la liste des comptes ressemblera à ceci :

```
"\n" +
"Bob.Bin|caL!Four6cHon|09/03/2022 15:43:15\n" +
"Pifou|Bar#Tendre24|06/03/2022 19:08:17\n" +
"Magnolia|volTAgeX!130|06/03/2022 19:11:25\n" +
"ver.ba_tim|vr0Um?Pow|\n"
```

Si `Bob.Bin` se déconnecte, et se reconnecte plus tard, disons le `10/03/2022 11:02:25`, il faudra alors aller modifier la date de la dernière connexion avec cette nouvelle date, comme ceci :

```
"\n" +
"Bob.Bin|caL!Four6cHon|10/03/2022 11:02:25\n" +
"Pifou|Bar#Tendre24|06/03/2022 19:08:17\n" +
"Magnolia|volTAgeX!130|06/03/2022 19:11:25\n" +
"ver.ba_tim|vr0Um?Pow|\n"
```

Pour obtenir la date courante, vous devez utiliser la méthode `dateCourante()` de la classe `TP2Util` fournie avec l'énoncé de ce TP. **NE COPIEZ PAS** cette méthode dans votre classe. Importez plutôt la classe `TP2Util` dans votre projet, et appelez la méthode comme ceci : `String dateCourante = TP2Util.dateCourante();`.

Vous NE DEVEZ PAS modifier la classe `TP2Util`.

1.4.1 Option 1 du menu de gestion du compte – modifier le mot de passe

Cette option permet à l'utilisateur de modifier son mot de passe. Au choix de cette option, le programme demande à l'utilisateur d'entrer le nouveau mot de passe. Le programme doit valider le mot de passe comme expliqué à la section 1.3. Si le mot de passe est égal à la chaine vide, l'opération de modification du mot de passe est annulée. Le programme affiche alors un message à cet effet puis réaffiche le compte (et le menu de gestion du compte). Si le mot de passe n'est pas égal à la chaine vide, le programme modifie le mot de passe de l'utilisateur par le nouveau mot de passe entré, affiche un message de succès de l'opération, puis réaffiche le compte (et le menu de gestion du compte).

Note sur l'implémentation

La modification du mot de passe consiste à remplacer l'ancien mot de passe par le nouveau, dans la ligne contenant les informations de ce compte, dans la liste des comptes `infosComptes`.

1.4.2 Option 2 du menu de gestion du compte – supprimer le compte

Cette option permet à l'utilisateur de supprimer son compte. Au choix de cette option, le programme supprime le compte, affiche un message comme quoi le compte a été supprimé avec succès, puis demande à l'utilisateur de taper <ENTRÉE> pour revenir au menu principal. Après cette opération, le compte n'existe plus.

Note sur l'implémentation

La suppression d'un compte consiste à supprimer la ligne contenant les informations de ce compte, dans la liste des comptes `infosComptes`.

1.4.3 Option 3 du menu de gestion du compte – se déconnecter

Cette option permet à l'utilisateur de se déconnecter de son compte. Au choix de cette option, le programme affiche un message d'au revoir, et demande à l'utilisateur de taper <ENTRÉE> pour revenir au menu principal.

ATTENTION : l'utilisation des tableaux (qu'on n'a pas encore vus) est interdite, dans ce TP. Ceci signifie que vous ne pouvez pas utiliser la méthode `split` de la classe `String` (qui retourne un tableau). De plus, l'utilisation des expressions régulières est interdite, ce qui veut dire que vous ne pouvez pas utilisez des méthodes comme `matches`, `replaceAll`... de la classe `String`.

=> Comme spécifications complémentaires, voir le fichier **exempleExec_2.txt** fourni avec l'énoncé du TP.

1.5 Option 3 du menu principale : Quitter

Au choix de cet option, le programme affiche un message de fin et se termine.

=> Comme spécifications complémentaires, voir le fichier **exempleExec_1.txt** ou **exempleExec_2.txt** fourni avec l'énoncé du TP.

2. Précisions supplémentaires

- Vous devez écrire votre programme dans une classe nommée `GestionComptesUtilisateurs` qui contiendra votre méthode `main` et toutes les autres méthodes que vous aurez conçues.
- Prenez soin de bien documenter (Javadoc) chacune de vos méthodes.
- Votre classe `GestionComptesUtilisateurs` doit se trouver dans le paquetage par défaut.
- Prenez soin de bien découper votre code à l'aide de méthodes. Voici quelques lignes directrices pour vous guider dans la conception de vos méthodes :
  - Chaque fois que vous vous apercevez que différentes parties de votre code se ressemblent énormément, essayer d'en faire une méthode bien paramétrée (si possible).
  - Faites une méthode pour chaque petite tâche spécifique à accomplir dans la résolution du problème.
- N'oubliez pas d'écrire les commentaires de documentation (Javadoc) pour chacune de vos méthodes (sauf la méthode `main`).
- Lorsqu'on vous demande de valider une valeur saisie par l'utilisateur, cela sous-entend une **BOUCLE** de validation qui sollicite la valeur, lit la valeur, et lorsque la valeur saisie est invalide, affiche un message d'erreur, sollicite et lit de nouveau la valeur, et ce tant que la valeur saisie n'est pas valide.
- Toutes vos méthodes doivent être `public static`.
- Vous DEVEZ utiliser la classe `Clavier.java` pour effectuer toutes les saisies.
- Les seules CLASSES PERMISES sont `String`, `Character`, `Math`, `Clavier` (pour la saisie), et `System` (pour l'affichage). L'utilisation d'autres classes que celles-ci sera pénalisée.
- Vous **NE DEVEZ PAS utiliser les expressions régulières** (qu'on ne verra pas dans ce cours), ni la classe `StringTokenizer`.
- **Vous NE DEVEZ PAS utiliser les tableaux (qu'on n'a pas vus encore). ATTENTION, l'utilisation de tableaux peut engendrer une pénalité pouvant aller jusqu'à 30% puisque vous passerez à côté de ce qu'on veut noter dans ce TP sur la manipulation des chaines de caractères.**

- Vous DEVEZ respecter à la lettre les **informations, les messages**, et le **format** de l'affichage qui sont montrés dans les exemples d'exécution fournis avec l'énoncé du TP. Vous devez considérer les exemples d'exécution (fournis avec l'énoncé du TP) comme des spécifications complémentaires à celles décrites dans cet énoncé.
- Votre programme NE DOIT JAMAIS PLANTER lors d'une saisie faite par l'utilisateur.
- Toute **VARIABLE GLOBALE** est **INTERDITE : toutes les variables doivent être déclarées à l'intérieur (et au début) d'une méthode** (sauf pour les boucles for où vous pouvez déclarer la variable de contrôle dans la boucle)**.**
  - Pour passer une information du programme appelant (celui qui appelle la méthode) au programme appelé (la méthode appelée), vous devez passer cette information en paramètres.
  - Pour passer une information du programme appelé (une méthode quelconque) au programme appelant (celui qui appelle la méthode), utilisez la valeur de retour de la méthode appelée.
- Utilisez des constantes (`final`) autant que possible : les messages affichés, les bornes de validation, etc.
- Les constantes doivent être déclarées et initialisées au niveau de la classe (constantes globales) : `public static final`…
- L'affichage des résultats doit se faire à la console.
- Utilisez des variables réelles uniquement lorsque requis. Ex. : un compteur ne doit pas être `float` ou `double`.
- Vous DEVEZ respecter le style Java.
- Votre fichier à remettre doit être encodé en UTF8.
- Votre code doit compiler et s'exécuter avec le JDK 7. Si vous n'utilisez que ce qu'on a vu en classe, ce sera le cas.

Le non-respect de toute spécification ou consigne se trouvant dans l'énoncé du TP (et les exemples d'exécution donnés avec l'énoncé du TP) est susceptible d'engendrer une perte de points.

Note : *Si quelque chose est ambigu, obscure, s'il manque de l'information, si vous ne comprenez pas les spécifications, si vous avez des doutes… vous avez la responsabilité de vous informer auprès de votre enseignante.*

3. Détails sur la correction

3.1 La qualité du code (40 points)

Concernant les critères de correction du code, lisez attentivement le document "**Critères généraux de correction du code Java"**. De plus, votre code sera noté sur le respect des conventions de style Java (dont un résumé se trouve dans le document **"Quelques conventions de style Java"**. Ces deux documents peuvent être téléchargés dans la section TRAVAUX PRATIQUES (ET BOITES DE REMISE) de la page Moodle du cours.

Note : Votre code sera corrigé sur la totalité ou une partie des critères de correction indiqués ci-dessus, sur le respect des spécifications/consignes mentionnées dans ce document, et sur le respect des bonnes pratiques de programmation.

3.2 L'exécution (60 points)

Un travail qui ne compile pas se verra attribuer la note 0 pour l'exécution.

Note : Votre code sera testé en tout ou en partie. Les points seront calculés sur les tests effectués. Ceci signifie que si votre code fonctionne dans un cas x et que ce cas x n'est pas testé, vous n'obtiendrez pas de points pour ce cas. Il est donc dans votre intérêt de vous assurer du bon fonctionnement de votre programme dans tous les cas possible. Notez que les exemples d'exécution fournis ne couvrent pas tous les cas à tester.

4. Date et modalité de remise

4.1 Remise

Date de remise : AU PLUS TARD le 1er avril 2022 à 23h59.
Le fichier à remettre : `GestionComptesUtilisateurs.java` (PAS dans une archive zip, rar, etc.)

Remise via Moodle uniquement.

Vous devez remettre (téléverser) votre fichier sur le site du cours (Moodle). Vous trouverez la boîte de remise du TP2 dans la section TRAVAUX PRATIQUES (ET BOITES DE REMISE).

Vérifiez deux fois plutôt qu'une que vous avez remis le bon fichier, car c'est celui-là qui sera considéré pour la correction.

4.2 Politique concernant les retards

Une pénalité de 10% de la note finale, par jour de retard, sera appliquée aux travaux remis après la date limite. La formule suivante sera utilisée pour calculer la pénalité pour les retards : `Nbr points de pénalité = m / 144,` où `m` est le nombre de minutes de retard par rapport à l'heure de remise. Ceci donne 10 points de pénalité pour 24 heures de retard, 1.25 point de pénalité pour 3 heures, etc.

Aucun travail ne sera accepté après 1 jour (24 h) de retard, et la note attribuée sera 0.

4.3 Remarques générales

- **Aucun programme reçu par courriel ne sera accepté**. Plus précisément, un travail reçu par courriel sera considéré comme un travail non remis.

- Vous avez la responsabilité de conserver des copies de sauvegarde de votre travail (sur disque externe, Dropbox, Google Drive, etc.). La perte d'un travail due à un vol, un accident, un bris… n'est pas une raison valable pour obtenir une extension pour la remise de votre travail.

- **N'oubliez pas d'écrire (entre autres) votre nom complet et votre code permanent dans l'entête de la classe à remettre.**

N'attendez pas à la dernière minute pour commencer le travail, vous allez fort probablement rencontrer des problèmes inattendus!

Le règlement sur le plagiat sera appliqué sans exception. Vous devez ainsi vous assurer de ne pas échanger du code avec des collègues ni de laisser sans surveillance votre travail au laboratoire. Vous devez également récupérer rapidement toutes vos impressions de programme au laboratoire.

BON TRAVAIL !