



Cégep **André-Laurendeau**

420-235-AL (Hiver 2023)

# Évaluation Certificative (Version alternative)

Orienté-Objet, Héritage, UML, Classe interne,  
Interface, Swing, Git, Entrées-Sorties

**NOTE** : Ce travail de synthèse alternatif est pour permettre à ceux d'entre vous qui n'auraient pas complété le travail original avant le mardi 16 mai à minuit de le faire, ou même de le substituer à leur remise originale s'ils le souhaitent.

Ceux qui avaient terminé **avant le 16 mai à minuit** (et qui sont satisfaits de leur travail original) n'ont donc rien à remettre d'autre. Si ce n'est pas votre cas, vous devrez faire les exercices de cet énoncé correspondants au TODOs originaux, puisque nous n'accepterons pas les *commits* git datés d'après le 16 mai à minuit pour le TP original.

**Échéance : 26 mai à minuit, ou tel qu'indiqué sur Léa**

Michel Généreux; David Giasson

## Objectif

Utiliser les concepts de programmation orientée-objet vus dans le cours pour compléter le logiciel fourni. Utiliser les diagrammes de classes UML correspondants pour guider votre travail. Accessoirement, utiliser la bibliothèque graphique Swing, les tests unitaires, lire et écrire dans un fichier de texte, réaliser une interface graphique, se servir du dépôt infonuagique GitHub et produire la documentation Javadoc.

## Directives importantes

- Ce travail pratique **doit être fait seul**.
- La remise se fera par Léa :
  - Zippez le répertoire qui contient le projet IntelliJ et l'historique git de votre code
  - Remettez ce fichier .zip sur Léa
  - **[Bonus]** Indiquez en commentaire sur la première ligne du fichier Main.java l'adresse URL de votre dépôt sur GitHub
- **La qualité du français est également importante.** Jusqu'à 10% de la note finale de votre travail pourrait être retirée pour cause d'un mauvais usage du français.
- Le travail doit être remis au plus tard à la date officielle indiquée sur Léa. Après cette date, 10% de pénalité sera enlevé par jour de retard jusqu'à un maximum de 5 jours de retard, après quoi la note 0 sera automatiquement attribuée.
- Bien entendu, toute forme de plagiat entraînera automatiquement la note zéro (0), ou pire encore. N'oubliez pas d'indiquer vos références si vous utilisez des extraits de documents ou de code écrits par une autre personne.

## Composition

Ce travail est constitué de **deux parties** distinctes qui doivent être réalisées **dans l'ordre**.

## Critères d'évaluation

- Qualité du code (formatage, choix des noms de variables, etc.)
- Production des classes et méthodes conformément au diagramme UML
- Fonctionnement correct du programme
- Gestion appropriée des exceptions
- Interface graphique fonctionnelle
- Utilisation judicieuse de git et GitHub
- Production de documentation JavaDoc (seulement pour les méthodes publiques, pas besoin de commenter les méthodes privées)

## Résumé des actions requises pour compléter votre travail

1. Clonez le nouveau projet de départ depuis GitHub à l'adresse suivante :  
[https://github.com/alfclul/TSubs\\_H23.git](https://github.com/alfclul/TSubs_H23.git)
2. Ouvrez et configurez le projet IntelliJ contenu dans le dossier cloné sur votre ordinateur.
3. À partir du projet IntelliJ de départ, complétez les quinze (15) « *TODOs* » dans votre programme. L'ordre numérique place le code le plus creux (i.e. ne dépendant pas d'autres *TODO*) d'abord, ce qui peut faciliter le codage, mais vous pouvez bien sûr les compléter dans l'ordre que vous voulez. Des indices placés près des mentions *TODO* dans le code pourraient vous être utiles pour compléter vos tâches.
4. Tout au long de votre travail, effectuez régulièrement des *commit* dans votre repo git (localement seulement, pas de « push » vers GitHub exigé).
5. [Bonus] Créez un dépôt privé sur GitHub pour votre projet, et copiez-y le code de votre travail. Inscrivez l'adresse (URL) de votre dépôt sur la première ligne de votre fichier source *Main.java* et invitez votre prof à ce repo pour obtenir quelques points bonus.
6. Remettez votre travail final sur Léa (un fichier zip contenant l'entièreté de vos deux projets, l'original et le nouveau), **ainsi que la « nouvelle grille de correction » (à remplir à la dernière page de cet énoncé pour nous permettre de savoir quels TODO corriger dans quelle version du TP).**

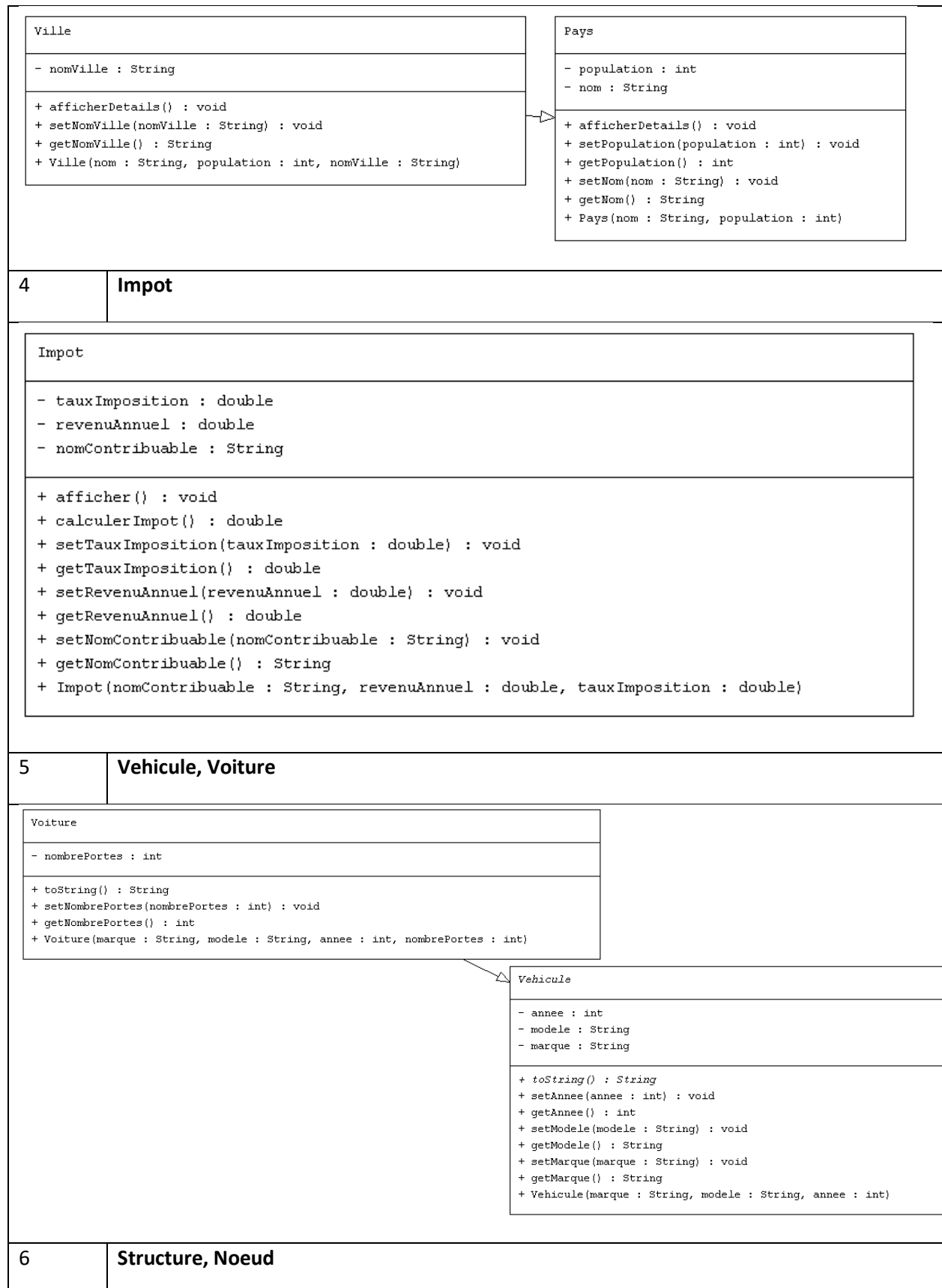
## Préambule

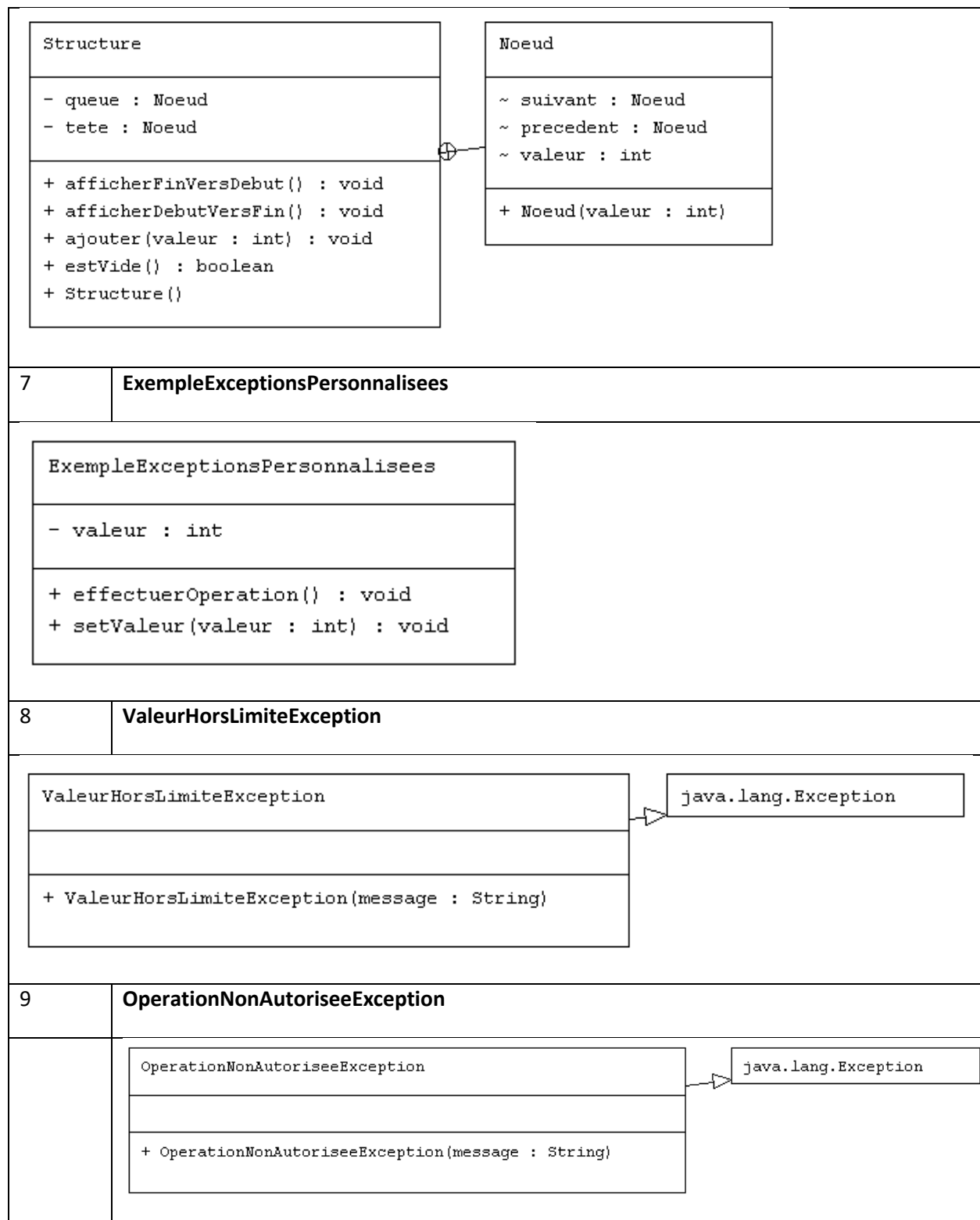
Ce travail de substitution est constitué d'une série de « *TODOs* » portant sur des volets spécifiques du travail de synthèse, sans constituer un travail « continu » où chaque partie dépend des autres. L'idée ici est d'évaluer les mêmes notions visées par le travail original par l'entremise de questions indépendantes ciblées.

## Partie 1 – TODOs 1 à 9

À partir des fichiers de départ et des diagrammes de classes suivants, complétez les *TODOs* pour que la méthode **Main.main()** fonctionne correctement.

TODO	CLASSES
1.1 et 1.2	<b>Chien, Chat, Animal</b>
2	<b>Footballeur, Joueur</b>
3	<b>Pays, Ville</b>





## Partie 2 – TODOs 10 à 15

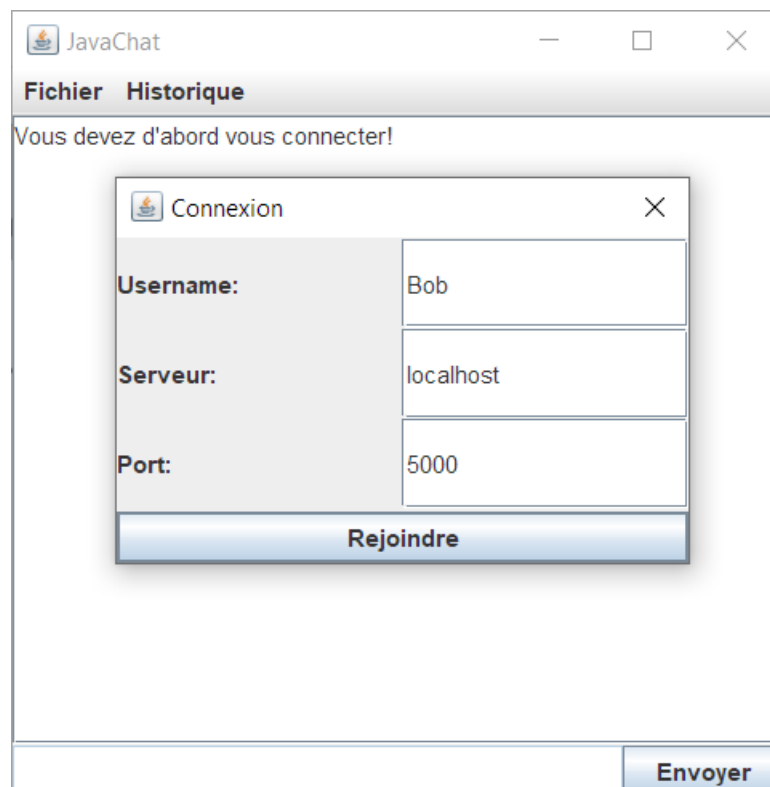
À partir des fichiers de départ et des indications ci-dessous, complétez les *TODOs* pour que les méthodes **messages.Historique.main()** et **GUIClient.main()** fonctionnent correctement.

NB. Pour tester votre code du *GUIClient*, assurez-vous de lancer d'abord la configuration Serveur, puis vous pourrez lancer le ou les clients. Vous pouvez aussi vous aider du *CLIClient* pour vérifier que les messages sont bien reçus et envoyés.

TODO  
10

Dans le fichier **GUIClient.java** :

Ajoutez les composants manquants pour demander l'information nécessaire à la connexion (nom d'utilisateur, adresse du serveur, port de l'application).



TODO  
11

**Dans le fichier GUIClient.java :**

Ajoutez le code qui permettra de créer le menu suivant (qui appelle les méthodes en italique) :

Fichier

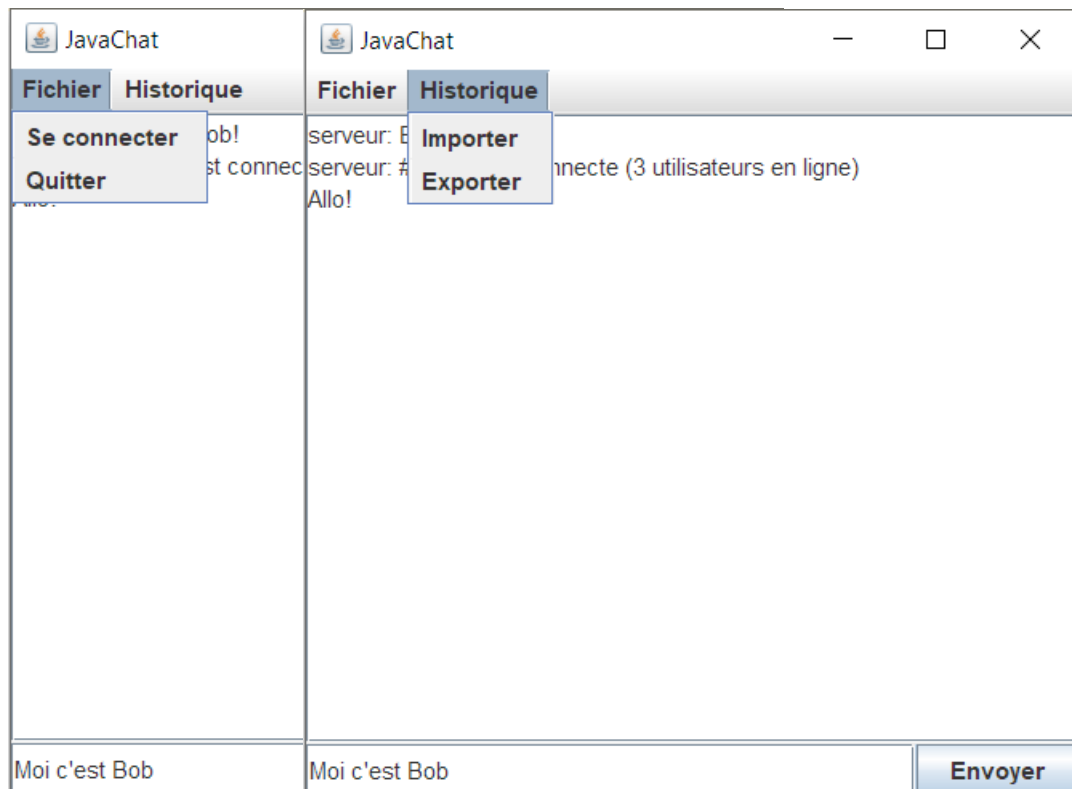
Se connecter -> Ouvre le dialogue de connexion: *connecter()*;

Quitter -> Quitte l'application: *System.exit(0)*;

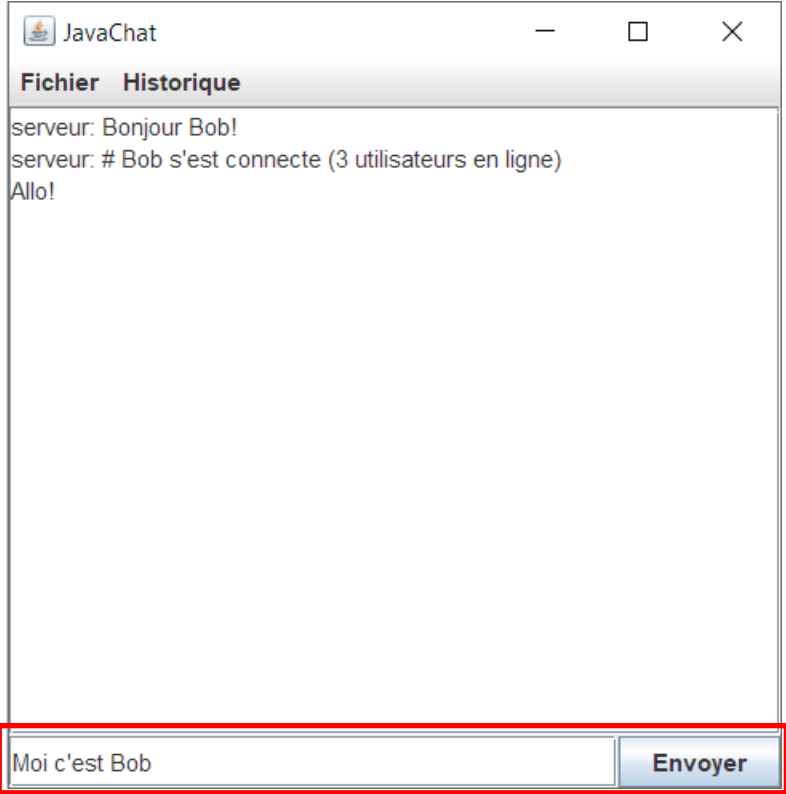
Historique

Importer -> Lit l'historique: *chargerHistorique()*;

Exporter -> Écrit l'historique: *sauvegarderHistorique()*;





<p>TODO 12</p>	<p><b>Dans le fichier GUIClient.java :</b></p> <p>Ajoutez le code nécessaire pour reproduire le bas de l'interface (une boîte de texte pour entrer un message (<i>sendBox</i>) et un bouton avec le texte "Envoyer").</p> 
<p>TODO 13</p>	<p><b>Dans le fichier GUIClient.java :</b></p> <p>Ajoutez une action au bouton "Envoyer" qui va lui permettre</p> <ul style="list-style-type: none"> <li>a) d'envoyer le message au serveur (appelez la méthode <i>send(String message)</i>),</li> <li>b) d'ajouter le texte du message au <i>textArea</i> 'chatBox' (déjà créé par le code fourni),</li> <li>c) de vider le contenu du <i>textField</i> 'sendBox' (que vous avez créé au TODO précédent).</li> </ul>

<p>TODO 14</p>	<p><b>Dans le fichier Historique.java (dans le package 'messages') :</b></p> <p>Ajoutez le code nécessaire pour permettre d'écrire chaque message (un par ligne) dans le fichier passé en paramètre à la fonction <i>écrire(String cheminFichier)</i> à partir des messages contenus dans le tableau 'messages'.</p> <p>NB. Chaque type d'objet message possède déjà une méthode <i>toString()</i> qui gère sa conversion en <i>String</i> de la manière attendue. Vous n'avez donc pas à modifier ces fichiers.</p> <p><b>Exemple de sortie attendue : historique.test.out</b> (devrait être identique à <b>historique.test.in</b>)</p>
<p>TODO 15</p>	<p><b>Dans le fichier Historique.java (dans le package 'messages') :</b></p> <p>Ajoutez le code nécessaire pour permettre de lire chaque message (un par ligne) depuis le fichier passé en paramètre, et de remplir le tableau 'messages' avec les objets 'Message' créés.</p> <p>NB. Chaque type d'objet message possède un constructeur qui pourra vous aider à interpréter les lignes de texte lues, mais vous devrez écrire le code qui permet de choisir le type d'objet approprié. Pour ce faire, observez les différences entre les sorties des différents types d'objets Message et trouvez les « marqueurs » qui permettent d'identifier le type de message correspondant.</p> <p><b>Exemple de résultat attendu :</b></p> <p>=&gt; TEST Lire le contenu de l'historique depuis le fichier historique.test.in  Lecture de l'historique depuis historique.test.in (5 lignes)  Admin: Bienvenue sur le serveur  Allo tout le monde!  Alice: Salut Bob!  @Bob Charlie: Yo, tu as fini le TP?  @Charlie Pas encore mais presque, toi?</p>

## Barème de correction (à remplir pour choisir ce qui sera évalué)

Étant donné que la correction se fera par un mélange des *TODOs* complétés dans le travail de synthèse original et ceux de la nouvelle version, vous devez inscrire ci-dessous (avec un « X ») ceux que vous souhaitez que je corrige.

Nouvelle grille de correction	Travail original <i>Tous les TODO commis <u>avant</u> le 16 mai à minuit</i>	Travail de substitution <i>Tous les TODO commis <u>après</u> le 16 mai à minuit</i>	Points
Critères généraux			25
Utilisation de Git	Inscrivez l’adresse de votre dépôt GitHub privé ici et invitez-moi (user : doorjuice) pour obtenir 1 point bonus.		5
Qualité du code	Évalué à travers les TODOs sélectionnés plus bas.		5
Javadoc	Seulement pour les méthodes publiques des classes publiques des TODO 1-9. Indiquez quelle version corriger si vous l’avez fait pour plus d’une version du travail.		5
Respect du/des diagramme(s) UML	Évalué à travers les TODOs sélectionnés plus bas.		10
Partie 1 : Programmation orientée-objet			45
TODO 01	Celui-ci?	Ou celui-là?	5
TODO 02	Celui-ci?	Ou celui-là?	5
TODO 03	Celui-ci?	Ou celui-là?	5
TODO 04	Celui-ci?	Ou celui-là?	5
TODO 05	Celui-ci?	Ou celui-là?	5
TODO 06	Celui-ci?	Ou celui-là?	5
TODO 07	Celui-ci?	Ou celui-là?	5
TODO 08	Celui-ci?	Ou celui-là?	5
TODO 09	Celui-ci?	Ou celui-là?	5
Partie 2 : Interfaces graphiques			20
TODO 10	Celui-ci?	Ou celui-là?	5
TODO 11	Celui-ci?	Ou celui-là?	5
TODO 12	Celui-ci?	Ou celui-là?	5
TODO 13	Celui-ci?	Ou celui-là?	5
Partie 3 : Lecture et écriture de fichiers			10
TODO 14	Celui-ci?	Ou celui-là?	5
TODO 15	Celui-ci?	Ou celui-là?	5

## Par exemple

Supposons que vous avez eu le temps de compléter 5 *TODOs* du le travail original avant le 16 mai à minuit (vous pouvez toujours rajouter des commentaires Javadoc après cette date), et le reste vous l'avez complété sur le travail de substitution.

Vous nous remettrez donc la grille suivante sur LÉA avec votre travail:

Nouvelle grille de correction	Travail original <i>Tous les TODO commis <u>avant</u> le 16 mai à minuit</i>	Travail de substitution <i>Tous les TODO commis <u>après</u> le 16 mai à minuit</i>	Points
Critères généraux			25
Utilisation de Git	<a href="https://github.com/vous/TS_H23">https://github.com/vous/TS_H23</a> (ceci est un exemple)		5
Qualité du code	Évalué à travers les TODOs sélectionnés plus bas.		5
Javadoc	Corriger la version du nouveau TP (ceci est un exemple)		5
Respect du/des diagramme(s) UML	Évalué à travers les TODOs sélectionnés plus bas.		10
Partie 1 : Programmation orientée-objet			45
TODO 01	X		5
TODO 02	X		5
TODO 03	X		5
TODO 04	X		5
TODO 05	X		5
TODO 06		X	5
TODO 07		X	5
TODO 08		X	5
TODO 09		X	5
Partie 2 : Interfaces graphiques			20
TODO 10		X	5
TODO 11		X	5
TODO 12		X	5
TODO 13		X	5
Partie 3 : Lecture et écriture de fichiers			10
TODO 14		X	5
TODO 15		X	5