Projet de programmation JAVA

1. Sujet : Application client-serveur de conjugaison

L'objectif est de réaliser une application client-serveur dont la tâche consiste à conjuguer un verbe saisi par l'utilisateur.

2. Hypothèses

L'application accepte les verbes du premier groupe (verbes en *ger* et verbes en *cer* compris), les verbes du deuxième groupe et les verbes du troisième groupe en *indre*, en *aître*, en *ettre* et en *oudre*. Le mode indicatif et la forme affirmative sont fixés. Les temps acceptés sont le futur, le présent et le passé composé.

3. Fonctionnement général de l'application

L'application réalise la saisie de l'infinitif du verbe et la saisie du temps, calcule la conjugaison aux six personnes puis affiche le résultat.

3.1 Exemple 1

infinitif saisi : *croire*. temps saisi : *présent*
résultat :
Je crois
Tu crois
Il croit
Nous croyons
Vous croyez
Ils croient

3.2 Exemple 2

infinitif saisi : *dissoudre*. temps saisi : *présent*
résultat :
Je dissous
Tu dissous
Il dissout
Nous dissolvons
Vous dissolvez
Ils dissolvent

4. Client

Le client fonctionne de la façon suivante :
Etape 1 : Il procède à la saisie des données "adresse IP du serveur" et "port du serveur" et se connecte ensuite au serveur.
Etape 2 : Il procède à la saisie de l'infinitif d'un verbe et du temps de conjugaison puis envoie la requête (infinitif + temps) au serveur.
Etape 3 : Il réceptionne la réponse du serveur (la conjugaison aux 6 personnes) qu'il affiche.
En général, pour une session, l'étape 1 est réalisée une seule fois alors que les étapes 2 et 3 sont répétées autant de fois que le souhaite l'utilisateur.

Le client est une application GUI (c'est-à-dire graphique) disposant des composants suivants :

1. vue saisie de l'adresse IP du serveur
2. vue saisie du port du serveur
3. vue saisie de l'infinitif du verbe
4. vue saisie du temps
5. bouton "connexion"
6. bouton "envoi requête"
7. vue indiquant si le client est connecté
8. vue montrant la conjugaison aux six personnes.

Les composants 1, 2, 3, 7 et 8 sont des champs de texte. L'utilisateur ne peut pas insérer du texte dans les vues 7 et 8.
Le client dispose d'une boîte à message "popup" pour signaler les erreurs (verbe inconnu, serveur injoignable, etc.)
Le bouton 6 est activé seulement lorsque le serveur est connecté. La vue 7 sert de témoin indiquant si le client est connecté.
Le client ne gère pas les erreurs de saisie de verbe ou de temps. Si une telle exception se produit, elle est détectée par le serveur qui envoie alors une réponse adéquate qui est affichée par le client dans la boîte "popup".

5. Serveur

Le serveur est multi-client. C'est une application console. Son rôle est de réceptionner les requêtes de conjugaison, d'effectuer les conjugaisons puis de renvoyer les réponses aux clients respectifs. Il contient le modèle (ou métier) de l'application. Le serveur fonctionne bien sûr de façon perpétuelle.

6. Protocole de communication

Pour le format des informations circulant entre client et serveur, vous êtes libres d'utiliser le format qui vous convient le mieux, il est toutefois conseillé d'utiliser le format texte.

7. Conseils pour réaliser ce projet

Qualité de la conception
Veillez à ce que les méthodes de réponse aux événements graphiques **RESTENT COURTES**. Les classes écouteurs d'événement graphique, peuvent bien sûr, s'il y a lieu, être organisées en hiérarchies. Pensez ces classes comme devant satisfaire une requête de l'utilisateur et non pas comme devant répondre à un clic (cf. Design Pattern ***Command***). Efforcez-vous de les rendre indépendantes du moyen physique que l'utilisateur a choisi pour s'exprimer (bouton, menu, raccourci clavier, etc.). Par ailleurs, l'emploi du design pattern *Model View Controller* peut considérablement simplifier la conception de l'application.

8. Contraintes :

- Evitez les instructions *switch* pour distinguer plusieurs cas.

- Evitez les redondances de codes

Attention, votre solution doit être pensée de telle sorte qu'il soit facile de l'étendre à la gestion de nouveaux cas tout en impactant le moins possible les classes existantes.

La notation tiendra compte du strict respect de ces consignes.

9. Extensions possibles

- raccourcis clavier et menu côté client
- temps *conditionnel*
- mode *subjonctif*

10. Organisation du travail

Les étudiants peuvent se grouper par équipes de 1 à 2 pour réaliser le travail.

11. Notation

La note finale est constituée de deux notes :

1. Note de qualité de code :
L'examen des fichiers source produit une note de "qualité de code" récompensant la rigueur de programmation (cf. paragraphes 7 et 8).

2. Note de fonctionnement :
Elle évalue si le programme satisfait (à l'exécution) le cahier des charges défini.

La note finale tient compte des éventuelles extensions réalisées.

12. Documents à rendre

Toutes classes JAVA (source et bytecode) composant l'application.

13. Date de remise

Tous les documents concernant le projet sont à rendre au plus tard le lundi 8 mai 2023 à minuit à D. Michel. N'oubliez pas d'indiquer : noms et prénoms des membres de l'équipe, nom de la filière et nom du projet. Le projet doit être rendu par courrier électronique à l'adresse suivante : domic62@hotmail.com.

14. Soutenance

Les soutenances seront organisées à partir du mardi 9 mai 2023. Elles se dérouleront à l'IUT. Les lieux et créneaux horaires de soutenance seront communiqués le lundi 1er mai 2023. Les créneaux disponibles pour placer ces soutenances seront communiqués sous la forme d'un fichier excel partagé. Les étudiants sont invités à s'y inscrire eux-mêmes pour prendre RDV. Tous les membres de l'équipe devront être présents. La soutenance dure environ 40 minutes.

N’hésitez pas à me poser des questions ou à solliciter des RDVs pour discuter de points de programmation.

Bon travail