TP Java RMI - 2

Introduction

Dans ce TP, vous allez implémenter un "jeu du pendu" utilisant la techno Java RMI. Il s'agit pour le client de deviner un mot qui est mémorisé sur le serveur, en envoyant successivement des lettres, et ce en un nombre d'essais minimal. On propose de faire 3 versions de ce jeu client-serveur, nommés **`Pendu1C`**, **`Pendu1S`** (C pour Client, S pour Serveur), **`Pendu2C`**, **`Pendu2S`**, etc.

Version 1

Dans cette 1re version, on se contentera d'une interface utilisateur (UI) minimale : un essai d'une lettre sera fait par un lancement du client, en donnant la lettre à essayer en argument. Le client devra ensuite afficher combien de fois le mot contient cette lettre, puis afficher le mot caché, avec les lettres déjà trouvées apparentes.

```
E.T._.D._._.N.T
```

Le mot à chercher sera donné en ligne de commande, en 1er argument du programme "serveur".

Architecture logicielle

On définira une classe **`HiddenWord`** qui contiendra le mot à chercher. Un objet de cette classe sera instancié dans le serveur, et le client va envoyer une requête sur cet objet, via la techno Java RMI.

Cette classe (et son interface) disposera des 3 méthodes suivantes :

- **`int tryLetter( char a )`** : envoi pour essai le charactère contenu dans 'a'. La fonction retournera le nombre de lettres trouvées.
- **`int nbTry()`** : renvoie le nombre d'essais négatifs déjà effectués.
- **`String getCurrent()`** : renvoie le mot avec les lettres inconnues remplacées par des `'_'`

Cette classe aura comme attributs :

- un entier **`nbBadTry`**, contenant le nombre d'essais erronés.
- une chaîne de caractères **`hiddenWord`** qui sera initialisée par le constructeur de la classe à partir de la valeur donnée en argument au lancement du serveur.
- une chaîne de caractères **`current`** contenant le mot caché avec les lettres déjà validées apparentes, et les autres remplacées par des `'_'`

Pour des raisons pratiques, ce dernier attribut sera mémorisé sous forme d'un tableau de caractères. En effet, une fois instancié, on ne peut pas modifier un objet de la classe **`String`**. Il faudra le convertir en **`String`** (pour affichage) via un appel à **`String.valueOf(current);`**

Version 2 : boucle infinie dans le client

Une fois cette première version validée, modifier le client en y insérant une boucle "tant que". On sortira de la boucle dès qu'une des conditions suivantes est remplie :

- Gagné : mot trouvé en un nombre d'essai inférieur au seuil
- Perdu : mot non trouvé une fois le nombre d'essais autorisés atteint.

Le nombre d'essais autorisés sera donné en 2e argument du serveur.

Version 3 : jeu multijoueurs

En partant de la version 2, on souhaite pouvoir se connecter au serveur depuis plusieurs clients. Le gagnant étant celui trouve la dernière lettre manquante. Modifier la méthode **`int tryLetter()`** en lui ajoutant un 2e argument de type **`String`** qui contiendra le nom du joueur. Celui-ci sera donné en argument du programme client.

Le serveur devra afficher le nom du gagnant.