Projet RMI – Blaguemule – part 2
Architecture pair à pair

1 Préalable

Créez tout d'abord un nouveau projet dans lequel vous pourrez faire une nouvelle application (inspirée de l'application de la partie 1).

2 Fonctionnement général

L'objectif de cette partie est de faire communiquer les **BlagueProvider** selon une architecture pair à pair. Chaque **BlagueProvider** est serveur et client à la fois. Il peut récupérer les blagues des autres serveurs et met à disposition toutes les blagues qu'il connaît (contenant éventuellement celles qu'il a déjà récupérées).

Désormais, il n'y aura plus de **BlagueProviderClient** et **BlagueProviderServeur.** Il n'existera qu'une classe **BlagueProvider** qui fera office de client et de serveur.

Au lancement du **BlagueProvider**, on passera comme paramètre le nom du **BlagueProvider** et les noms sous lesquels sont référencés les autres **BlagueProvider** déjà enregistrés.

3 Interface BlagueProviderPairApair

BlagueProviderPairAPair qui dispose des méthodes

- **String getNom()** retournant le nom du **BlagueProvider** (nom est attribut de la classe)
- **String[] getAllNAmes()** retournant la liste des noms des blagues connues par le BlagueProviderPairAPair
- **Blague getBlague(String nom)** retournant la blague ayant pour nom le String passé en parametre.

4 Classe BlagueProvider

La classe **BlagueProvider** implémente l'interface **BlagueProviderPairAPair**

Cette Classe dispose en outre

- des attributs
  - **nom** le nom du **BlagueProvider**
  - la liste des références distantes nommée **listeRef** des autres **BlagueProvider** (ces références sont à récupérer au lancement de l'application)
- des méthodes

- **ajoutBlague(Blague b)** permettant d'ajouter une blague en local
- **getAllNames** permettant de récupérer la liste des noms des blagues d'un serveur donné à partir de sa référence distante
- **telechargeBlague** qui permet de télécharger une blague en l'ajoutant aux blagues référencées à partir d'une référence distante et du nom de la blague

Question a :
Écrire la classe **BlagueProvider.**

Question b
Quel est le codebase nécessaire ? Générez le fichier .jar correspondant.

5 Main

Le main de la classe **BlagueProvider** a pour objectif de remplir la liste **listeRef**, de créer une interface graphique et de s'enregistrer auprès du rmiregistry.

Le main doit fonctionner de la manière suivante

- création du **BlagueProvider** ayant pour nom le premier argument passé dans le main.
- ajout de la blague en fonction du nom du **BlagueProvider** (pour avoir des serveurs avec des contenus différents)
- récupération du registry
- récupération des références distantes (autres cases du tableau args)
- création de la référence distance et enregistrement au registry (référencement au nom du **BlagueProvider**)

Question e :
Ecrire le main sans vous préoccupez de l'interface utilisateur Vérifier que tout fonctionne correctement à l'aide de tests adaptés.

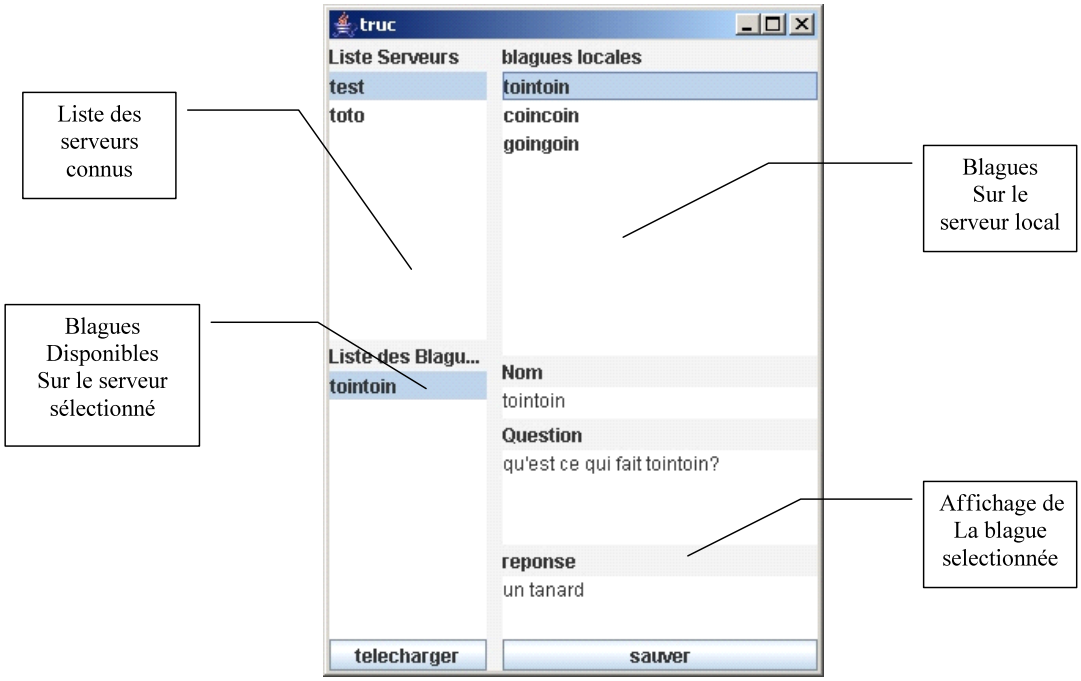
6 Interface graphique (optionnelle)

Cette interface est optionnelle. Ce TD met l'accent sur les objets distribués. Vous pourrez réaliser l'interface utilisateur par ligne de commande.

Néanmoins, l'interface graphique pourrait être structurée de la manière suivante :

- une partie (à droite sur l'exemple) contenant ce qui est distant
  - la liste des serveurs références
  - quand on clique sur un serveur, la liste des blagues disponibles sur le serveur se met à jour
  - quand on sélectionne une blague et qu'on clique sur le bouton télécharger, la blague est ajoutée à la liste des blagues locales
- une partie (a gauche sur l'exemple) contenant les blagues locales
  - quand on sélectionne une blague, celle-ci s'affiche dans la partie information (des JtextArea modifiables)

- quand on clique sur le bouton sauvegarder, on ajoute une nouvelle blague correspondant aux informations des JTextArea, cette blague est ajoutée à la liste au dessus et est instantanément disponible pour les autres serveurs.



Question d :
Créez l'interface graphique

On utilisera la clase **JList** pour gérer les listes. Pour modifier le contenu de la **JList**, il faut utiliser la méthode **setListData** et pour accéder au contenu sélectionné, utilisez la méthode **getSelectedValue**.

7 Exécution

Question e :
Testez que tout fonctionne en lançant plusieurs clients.

Il est nécessaire que
- tous les clients soient lancés sur la même machine (on ne peut pas faire de rebind sur un rmiregsitry non local pour des raisons de sécurité)
- les clients reçoivent en paramètre les noms de clients déjà lancés
- les clients soient lancés dans le bon ordre (conformément aux paramètres passés)