TP n°1

Découverte de Jade

1 Organisation de l'UE

— Examen final sur table : 50% de la note. Projet : 50% de la note.
— Toutes les ressources de l'UE seront mises en ligne sur Moodle. Vérifiez que vous avez bien accès à l'ue : MU4IN202 - S2 - FoSyMa.
`https://moodle-sciences-24.sorbonne-universite.fr/course/view.php?id=4543`

— Tous les TP, et donc le projet, sont à faire en binome.
Le suivi des TP est fait en séance et sur Discord depuis le serveur dédié au projet : `https://discord.gg/JZVz6sR`.

Une fois sur le serveur, allez sur le canal #roles et indiquez votre binome.

2 Prise en main de l'environnement de travail

Les TPs seront réalisés en JAVA (ou Scala si vous le souhaitez) sur la plateforme multi-agents JADE, et sont préparés pour l'ide Eclipse. Vous pouvez utiliser un autre ide, cependant le SAV n'est garanti que pour Eclipse.

— **Présentation de Jade**
— **Jade Programming Tutorial**
`https://jade-project.gitlab.io/docs/JADEProgramming-Tutorial-for-beginners.pdf`

— **Programmer's Guide**
`https://jade-project.gitlab.io/docs/programmersguide.pdf`

Ces documents, également disponibles sur moodle, vous permettront de vous familiariser avec les différentes fonctionnalités de JADE et la façon de programmer les agents. Vous devrez impérativement en prendre connaissance afin d'être en mesure de réaliser le projet (qui démarre la semaine du 14 février).

Exercice 1 Installation et exécution de StartJade : `https://startjade.gitlab.io`

Le projet `StartJade` que vous allez installer pour ce TP va vous permettre de découvrir le fonctionnement de JADE, de ses agents, et la façon dont ils communiquent entre eux. `StartJade` est, comme son nom l'indique, une bibliotheque d'exemples, des plus basiques (comment envoyer un message, comment creer un comportement de type machine-a-etat) aux plus complexes (mécanismes d'enchère, de coordination,..)

Elle contient contient 5 packages (`fundamentals, princ, exercices, protocols, tools`). Lors de cette séance introductive, nous nous intéresserons principalement à la classe `Principal.java` du package `princ`, utilisée comme pour lancer le code, et aux packages `fundamentals` et `exercices`.

Installation

On supposera ici que Java et Eclipse sont déja installés sur votre machine, si tel n'est pas le cas, suivre les consignes indiquées ici : `https://startjade.gitlab.io/install-and-execute-the-project/`.

Sous Eclipse :

1. `File/Import/Git/Projects from Git/ Clone URI/`
2. Collez l'adresse ci-après dans le champ 'URI' : `https://gitlab.com/startjade/startJade.git`
3. `Next/Next/Next/Next/Finish`
4. Votre programme est prêt à être lancé.

Exécution

1. Lisez le code source de l'exemple fourni dans `>fundamentals.communication` pour en comprendre la structure.
2. Sous Eclipse, sélectionnez le fichier `Principal.java` dans le package `princ` puis faites `Clic-droit/Run-as/Java-application`. L'exemple est automatiquement lancé.

   Lorsque l'exemple est lancé, le programme crée :
   — 2 fenetres graphiques associées à 2 agents qui gèrent la plateforme (Rma et Sniffeur)
   — 1 main container, 3 containers, 4 agents
   — Agent0 sur le container1
   — Agent1 et Agent3 sur le container2
   — Agent2 sur le container3
   — Puis attend que vous appuyiez sur une touche dans la fenetre de la console (ne pas le faire tout de suite). Le cas échéant, l'Agent0 va envoyer un message à l'Agent1, c'est tout :)

3. Lisez les messages qui s'affichent dans la console eclipse. Le Sniffeur est un agent fourni par JADE vous permettant de regarder quels sont les messages échangés entre les agents de la plateforme. Nous ne l'avons pas activé par défaut afin de vous permettre de choisir quels agents seront "sniffés".
   (a) Dans l'interface du Sniffeur, dépliez `AgentsPlatforms` et les `containers`
   (b) Faites un clic-droit sur l'Agent0 et sélectionnez `doSnif`, faites de même avec l'Agent1 et l'Agent2.
   (c) Retournez dans la console d'eclipse, cliquez dedans pour la rendre active et appuyez sur entrer.

   Le sniffeur vous permettra dans la suite de vérifier manuellement le comportement de vos agents.
4. Lisez, comprenez puis exécutez l'exemple `>exercices.sum1_1`.
   Plus complexe, cet exemple présente un échange entre deux agents. En terme de code, il sépare également les classes définissant les comportements des classes d'agents. Cette organisation est à privilégier.

   (a) Pour activer cet exemple, allez à la ligne 158 de la classe `Principal.java`, commentez-là et décommentez la ligne 159 : `private static EXAMPLE EXAMPLE_TO_USE=EXAMPLE.EXERCICES_SUM1_1;`
   (b) Lancer le programme sans oublier d'activer le sniffeur la première fois.

3 Protocoles, rôles et comportements

Vous allez maintenant implémenter quelques éléments simples vous permettant de vous familiariser avec les notions de Protocoles, rôles et comportements.

Exercice 2 Communication 2-1 : Envoi de 10 et 15 entiers

En vous inspirant de l'exemple précédent, écrivez un programme permettant à un agent A et un agent C d'envoyer respectivement 10 et 15 messages à l'agent Somme (chaque message contenant un unique entier tiré aléatoirement). L'agent Somme doit recevoir les différents entiers, réaliser la somme propre à chaque émetteur puis lui retourner son résultat (l'agent A reçoit donc la somme des 10 entiers, et l'agent C la somme de ses 15 entiers). Les agents A et C doivent afficher le résultat obtenu dans le terminal lors de sa réception.

Il existe différentes mise-en-oeuvre possibles. Réfléchissez à leurs avantages et inconvénients respectifs dans le cas ou le nombre d'agent serait amené à croitre ou changer au cours du temps. Testez dans un premier temps avec respectivement 2 et 3 entiers avant de passer à 10 et 15 entiers.

Pour la mise en oeuvre :

1. Créez un package `exercice2` sous le package `exercices` et intégrez y le code de vos agents et comportements.
2. Dans Principal.java, allez dans la methode `createAgents` et complétez le case `Exercices_EX02` en regardant ce qui a été fait pour les autres exemples.
3. Sélectionnez l'exemple que vous voulez lancer dans le bloc démarrant ligne 157, puis exécutez le programme.

Exercice 3 Communication 1-n : Réception puis délivrance de k d'entiers

Écrivez un protocole permettant à un agent *somme* de demander à l'ensemble des autres agents (supposés connus) de lui envoyer des nombres entiers (tirés aléatoirement). L'agent *somme* doit réceptionner les différents entiers jusqu'à atteindre k entiers reçu. Il doit alors afficher la somme obtenue et demander aux agents d'arrêter de transmettre de nouvelles valeurs.

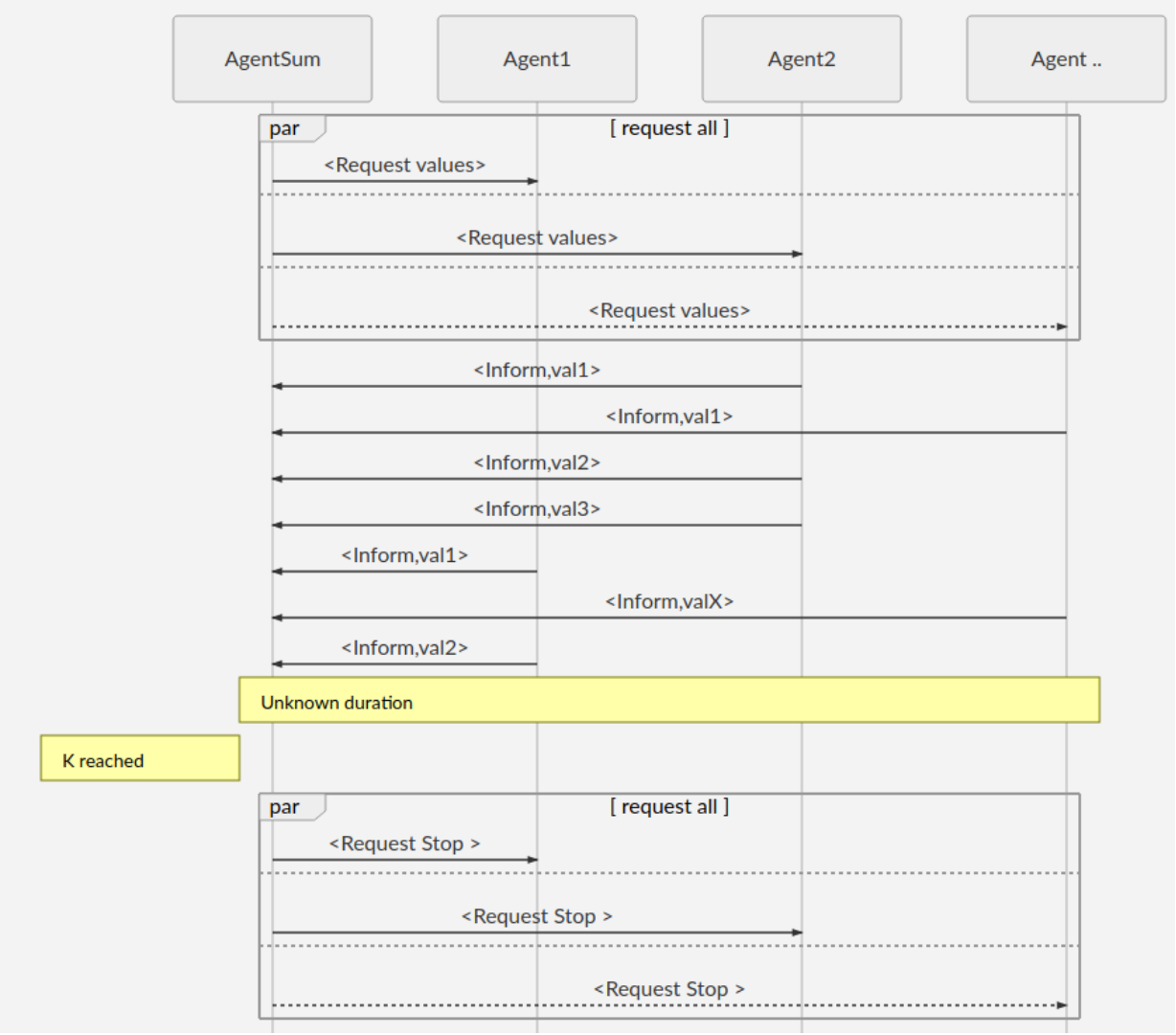

FIGURE 1 – Exercice 3 : Exemple de comportement minimal attendu

Réfléchissez bien aux différents behaviours nécessaires pour réaliser ce protocole. Plusieurs solutions sont possibles.

Quels problèmes apparaissent ? Comment les résoudre ?

Exercice 4 Communication n-1 : Envoi de différents entiers dont la valeur est choisie en-ligne

Dans l'exercice 2, l'agent en charge de faire la somme connait dès sa création les agents qui vont le contacter ainsi que le nombre d'entiers qu'il va recevoir en provenance de chacun. Ces valeurs sont en effet passées en parametres de ses behaviours lors de l'instanciation de l'agent (respectivement 10 pour A et 15 pour B). Cela fonctionne mais n'est pas générique, dans la pratique les paramêtres sont très rarement connus hors-ligne. Il devient dès lors nécessaire d'instancier les behaviours dynamiquement.

En repartant du travail réalisé pour l'exercice 2, créez un protocole SOMME plus générique et permettant à l'agent AgentSum de pouvoir répondre aux demandes de sommation arrivant de n'importe quel agent au cours du temps. Une fois opérationnel, le sniffeur devrait permettre d'observer un comportement de ce type :

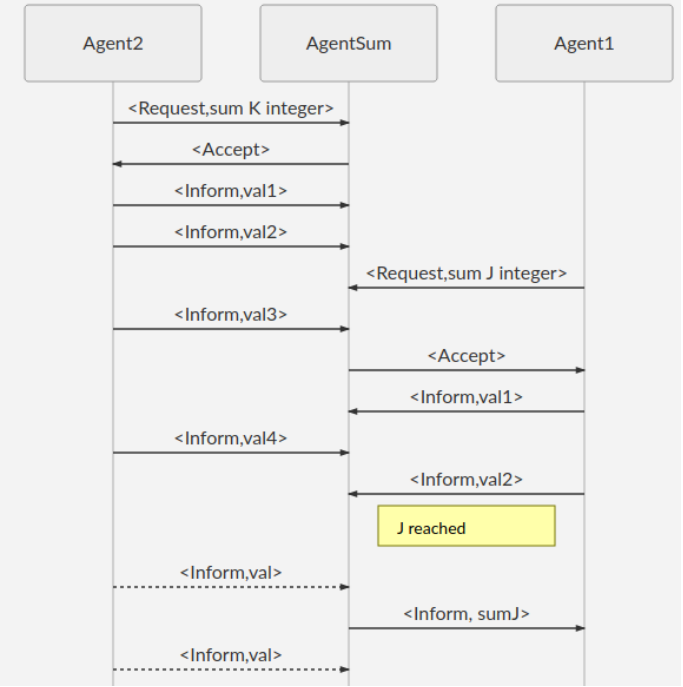

FIGURE 2 – Exercice 4 : Exemple de comportement minimal attendu.

Réfléchissez bien à la structuration de vos behaviours.

Exercice 5 Communication n-n : Uniquement sur papier

Réfléchissez à la généralisation de chaque protocole des exercices précédent de façon à ce qu'il soit possible d'ajouter autant d'agents de n'importe quel rôle sans que cela ne pose problème. Quels sont les éléments à prendre en compte, que faudrait-il modifier dans vos différents protocoles ?