LOG735 – Systèmes distribués

Laboratoire 2 : Transmission synchronisée d'événements entre applications

Auteurs : Simon Paquette, Charles-André Bouchard, Lévis Thériault
Réviseurs : Jérôme Gagnon (E11), Jean-Philippe Legault (E14)

Résumé

Vous devez permettre, dans le cadre d'une simulation de système distribué, l'envoi d'événements entre applications de manière synchronisée, offrant une alternative à la méthode asynchrone de départ.

Durée du laboratoire

Deux séances : 21 mai et 28 mai 2015

Objectifs

- Appliquer la notion de synchronisation lors de la propagation d'événements dans un système distribué basé sur un bus d'événements.

Matériel fourni

- Le code source de l'application de départ avec les *packages* suivants :
    - Application
    - Eventbus
    - Events

Contexte

Dans le contexte de la simulation, le *bus d'événements* consiste en une composante logicielle qui relie plusieurs applications d'un système distribué fictif entre elles. Ces liens sont effectués à l'aide de *connecteurs* qui sont responsables de la transmission des événements entre les applications et le bus d'événements.

Lorsqu'une application déclenche un *événement* et souhaite le propager aux autres applications du système, les connecteurs de chaque application déterminent si l'événement fait partie de la liste d'événements qu'ils doivent intercepter. Si c'est le cas, l'événement est transféré à l'application et celle-ci, dans le contexte de la simulation, l'affiche dans sa liste d'événements reçus

Application de départ

L'application de départ consiste en trois *packages* avec les classes suivantes :

- Application : Contient trois différentes Applications simulées qui ont la capacité d'envoyer un type d'événement qui leur est propre aux autres applications du système (voir Figure 1).
    - MainPartOne.java, MainPartTwo.java, MainPartThree.java : les fichiers d'exécution des *App Un, App Deux* et *App Trois* simulés.
    - UIMainWindow.java : l'interface graphique de chacune des Applications.
    - EventBusConnector.java et IEventBusConnector.java : le connecteur et son interface qui reçoivent et transmettent les événements du des Applications au bus d'événements.

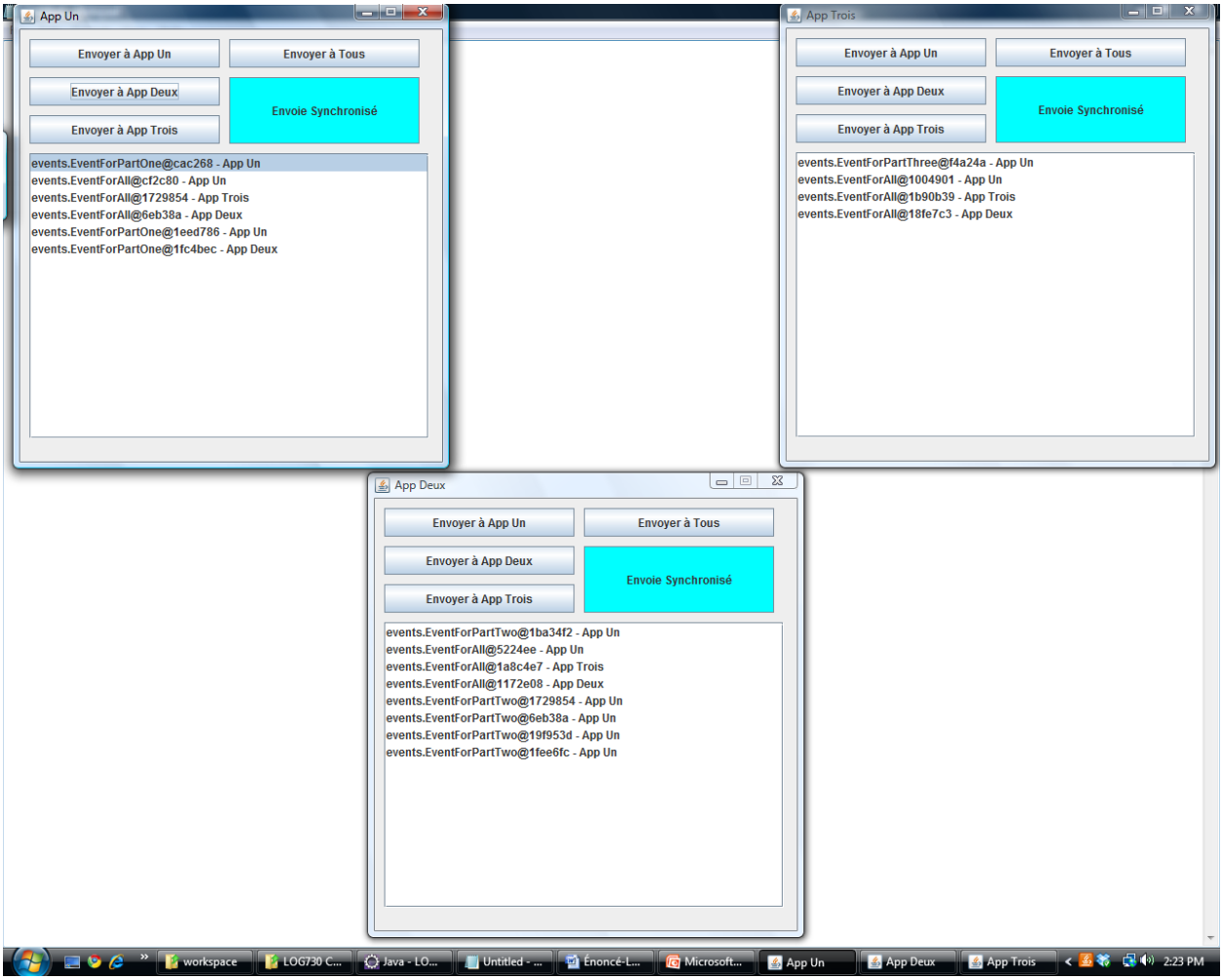

Figure 1 - Applications en cours d'exécution

- EventBus : Contient les classes qui gèrent le bus d'événements reliant les Applications et permettant la transmission d'événements.
    - EventBusCommunicator.java et IEventBusCommunicator.java : le communicateur et son interface qui reçoivent et transmettent les événements du bus d'événements aux Applications.
    - MainServerBus.java, EventBusThread.java, IEventBusThread.java et EventBusServerThread.java : les classes qui gèrent le bus d'événements.
- Events : Contient les classes qui définissent les différents types d'événements ainsi que leurs interfaces.

Contraintes d'implémentation

- Le laboratoire doit être travaillé à l'aide du langage de programmation Java.
- Vous ne devez **pas** changer la valeur du délai passé au constructeur des interfaces graphiques dans les classes MainPartOne, MainPartTwo et MainPartThree. L'introduction de ce délai est volontaire, dans le but de simuler une latence réseau. Cette méthode de résolution ne donnera **aucun point** pour la partie implémentation.

Instructions de démarrage

1. Créez un projet Java dans votre IDE à partir de l'application de départ.
2. Exécutez la classe EventBus.MainEventBus.java : le bus d'événements est démarré et prêt à recevoir les connexions provenant des Applications sur le port 12045.
3. Exécutez les classes MainPartOne.java, MainPartTwo.java et MainPartThree.java : les trois instances *App Un*, *App Deux* et *App Trois* sont en cours d'exécution et peuvent être utilisées pour lancer des événements.

Étapes d'implémentation et questions

L'implémentation requise dans ce laboratoire consiste à implémenter adéquatement la fonctionnalité d'envoi synchronisé de chacune des applications. Lorsque le bouton « Envoi Synchronisé » est exécuté, le système doit **constamment et systématiquement** afficher le message « Vous Avez Réussi » dans l'ordre numérique des trois applications.

Ainsi, dans l'ordre, *App Un, App Deux* et *App Trois* doivent afficher « Vous », « Avez » et « Réussi ». Votre implémentation sera considérée réussie si ce résultat est obtenu tout en utilisant convenablement les connecteurs ainsi que le bus d'événements.

1. [15%] De façon générale, expliquez les différents défis à relever par rapport à la synchronisation (de messages et/ou d'applications) et parlez brièvement des solutions possibles pour chacun des défis mentionnés.

2. [55% (incluant l'implémentation)] Expliquez l'approche que vous avez prise pour permettre l'envoi synchronisé des messages. Expliquez ensuite les modifications apportées à chaque classe (création de classes, modification des attributs et des méthodes.

3. [15%] Discutez des améliorations possibles à effectuer à votre implémentation en fonction des notions d'extensibilité et de performance : que devriez-vous corriger pour que l'envoi de messages synchronisés fonctionne avec un nombre variable d'applications?

4. [15%] Discutez d'au moins une solution alternative qui aurait pu répondre à la tâche demandée ; pensez à comment les responsabilités auraient pu être différemment assignées entre les applications et le bus d'événements.

Correction interactive

Le lendemain de la remise, une évaluation sommaire et **obligatoire** de la fonctionnalité de l'application sera faite pour chaque équipe. Les deux étudiants de l'équipe doivent être présents ; en cas d'absence non-justifiée, les points alloués au bon fonctionnement de l'implémentation seront retirés.

Pénalités de correction

Les pénalités suivantes ainsi que leur valeur sont applicables comme suit :

- Fautes de français et erreurs de rédaction : -0.5% par erreur, maximum 20 fautes (-10%);
- Documentation du code source : -5% si insuffisante, -10% si pratiquement inexistante;
- Erreurs de compilation : perte de **tous** les points alloués au bon fonctionnement;
- **Retard de remise non-justifié : note de zéro automatique et non négociable**;
- **Non-respect des normes de remise : note de zéro automatique et non négociable**.

Normes de rédaction et de remise

- Le rapport doit contenir une courte introduction ainsi qu'une conclusion sur le laboratoire;
- Les normes de rédaction contenues dans la présentation « Normes de rédaction et de remise » s'appliquent dans leur intégralité;
- La remise électronique doit contenir l'application améliorée avec tous les *packages* nécessaires.

Échéances

- Remise électronique (code source + rapport électronique) : **Le 3 juin avant 23h59**;
- Remise papier du rapport lors de la correction interactive;
- Correction interactive : **Le 4 juin à 9h00 (groupe A) ou à 11h00 (groupe B)**.