BERTIERE Thomas
DE BRITO Guillaume

INSA TOULOUSE

Programmation Orientée Objet
Projet : ChatSystem

Rapport de test

Groupe : 4I TD B2

Année : 2016 - 2017

Introduction :

Nous allons présenter les tests que nous avons réalisés dans le cadre du projet ChatSystem. Nous avons établi avec notre groupe de TP une liste de tests à réaliser. Nous voulions nous échanger nos tests et implémenter le design pattern Adapter afin que ces tests puissent s'exécuter sur chacun de nos programmes. Seulement, aucun de nous n'a réussi à implémenter le design pattern Adapter et nous avons donc décidé de nous échanger nos test en les adaptant nous mêmes à nos programmes.

Nous avons organisé nos tests en différentes parties pour tester l'intégralité du projet.

Nous testerons donc :

- La classe Message
- La connexion et la déconnexion d'un utilisateur
- L'envoi de messages
- La réception de messages
- La classe Model et la liste d'utilisateurs
- L'envoi et la réception de fichiers

Test de la classe message :

Par le biais des ces tests nous allons tester si la classe Message est fonctionnelle, c'est à dire que nous avons bien plusieurs types de messages qui ne se confondent pas. Il faut aussi que ces messages se sérialisent correctement et puissent s'envoyer sur le réseau. De plus il faut que ces messages puissent être reconnu par l'application les recevants.

1) **Test :** Sérialisation d'un message et envoi
   Nom de la fonction JUnit : testSerializationMessageEtSend
   Description : Crée un message, le sérialise et l'envoi sur le réseau. Le test passe si ces trois étapes sont réalisées sans erreurs.
   Résultat : Test OK.

2) **Test :** Réception d'un paquet réseau, déserialisation d'un Message
   Nom de la fonction JUnit : testReceivePacket
   Description : Réception d'un paquet réseau sur une application, désérialisation du paquet en Message. Le test passe si le résultat de la désérialisation est un message.
   Résultat : Test OK.

3) **Test :** Réception d'un paquet réseau, déserialisation d'un message texte
   Nom de la fonction JUnit : testReceiveMessage
   Description : Réception d'un paquet réseau sur une application, désérialisation du paquet en Message. Le test passe si le résultat de la désérialisation est un message texte.
   Résultat : Test OK.

4) **Test :** Réception d'un paquet réseau, déserialisation d'un message texte
   Nom de la fonction JUnit : testReceiveMessageTypeOk
   Description : Réception d'un paquet réseau sur une application, désérialisation du paquet en Message. Le test passe si le résultat de la désérialisation n'est pas un message hello. Cela nous permet de confirmer que l'on reçoit le bon type de message et pas un autre.
   Résultat : Test OK.

5) **Test :** Réception d'un paquet réseau, déserialisation d'un message texte, vérification du contenu
   Nom de la fonction JUnit : testContentReceiveMessageText
   Description : Réception d'un paquet réseau sur une application, désérialisation du paquet en Message. Le test passe si le contenu du message envoyé est le même que celui qui a été envoyé. Cela permet de confirmer que l'on ne perd pas d'information lors de la sérialisation ou de la désérialisation.
   Résultat : Test OK.

Test de la classe connexion et de la déconnexion d'un utilisateur :

Avec ces test, nous allons tester l'état connecté d'un utilisateurs après des lancements de connexion ou encore de déconnexion. Nous allons pouvoir observer si l'application réagit correctement à ces évènements.

1) **Test :** Lancement de la connexion d'un utilisateur
   Nom de la fonction JUnit : test1Connexion
   Description : L'utilisateur lance le processus de connexion. Le test passe si le booléen qui contrôle l'état connecté est bien passé à true.
   Résultat : Test OK.

2) **Test :** Lancement de la déconnexion d'un utilisateur
   Nom de la fonction JUnit : test2Deconnexion
   Description : L'utilisateur lance le processus de connexion. Puis après un court temps d'attente, lancement du processus de déconnexion. Le test passe si le booléen qui contrôle l'état connecté est bien passé à false.
   Résultat : Test OK.

3) **Test :** Lancement de la déconnexion d'un utilisateur puis lancement de la connexion
   Nom de la fonction JUnit : test3Reconnexion
   Description : L'utilisateur lance le processus de connexion. Puis après un court temps d'attente, lancement du processus de déconnexion. Puis après un autre temps d'attente, lancement du processus de connexion. Le test passe si le booléen qui contrôle l'état connecté passe correctement à true ou false selon l'état de la connexion ou de la déconnexion.
   Résultat : Test OK.

4) **Test :** Lancement de la connexion d'un utilisateur avec échec puis reconnexion
   Nom de la fonction JUnit : test4ReconnexionHelloNotOk
   Description : L'utilisateur lance le processus de connexion. Un autre utilisateur lui envoie un message helloNotOk qui entraîne un échec de la connexion. L'utilisateur tente ensuite de se reconnecter sans recevoir de message helloNotOk cette fois. Le test passe si le booléen qui contrôle l'état connecté reste bien à false après la première tentative de connexion et passe à true après le deuxième essai.
   Résultat : Test OK.

Test de l'envoi de message :

Nous allons tester tous les messages qui s'envoient lors des différentes étapes de l'application chatsystem. Cela permet de vérifier si l'application envoient bien les bons messages aux bons moments.

1) **Test :** Envoi du message hello lors de la connexion
   Nom de la fonction JUnit : test1EnvoiMessageHello
   Description : L'utilisateur lance le processus de connexion. Un autre utilisateur reçoit le message hello envoyé lors de la connexion. Le test passe si ce message reçu est bien un message hello.
   Résultat : Test OK.

2) **Test :** Envoi du message check lorsque l'utilisateur est connecté
   Nom de la fonction JUnit : test2EnvoiMessageCheckApresConnexion
   Description : L'utilisateur envoie des messages check à répétition. Un autre utilisateur reçoit ces messages. Le test passe si ces messages reçus sont bien des check et que ceux-ci ne correspondent pas à des checkOk (représentants la réponse d'un check) mais à des check.
   Résultat : Test OK.

3) **Test :** Envoi du message goodbye lors de la déconnexion d'un utilisateur
   Nom de la fonction JUnit : test3EnvoiMessageGoodbyeDeconnexion
   Description : L'utilisateur envoie un message goodbye lors de la déconnexion. Le test passe si un autre utilisateur reçoit bien ce message goodbye.
   Résultat : Test OK.

4) **Test :** Non réception de message check après la déconnexion
   Nom de la fonction JUnit : test4VerifPasDeCheckApresDeconnexion
   Description : Après une déconnexion, l'utilisateur n'est pas censé continuer d'envoyer des messages check. Le test passe si un autre utilisateur ne reçoit plus de message check durant un certains temps (supérieur à l'intervalle de temps entre deux envois de check) après la déconnexion.
   Résultat : Test OK.

Test de la réception de message :

Ces test permettent de vérifier si l'application réagit bien à toutes sortes de messages reçus.

1) **Test :** Réception d'un message hello et envoi d'un message helloOk
   Nom de la fonction JUnit : test1ReponseHelloOK
   Description : Un utilisateur connecté reçoit un message hello. Il renvoie un message helloOK. Le test passe si l'autre utilisateur reçoit bien le message helloOk.
   Résultat : Test OK.

2) **Test :** Réception d'un message hello et envoi d'un message helloNotOK
   Nom de la fonction JUnit : test2ReponseHelloNotOK
   Description : Un utilisateur connecté reçoit un message hello. Il renvoie un message helloNotOK. Le test passe si l'autre utilisateur reçoit bien le message helloNotOK.
   Résultat : Test OK.

3) **Test :** Réception d'un message check et envoi d'un message checkOK
   Nom de la fonction JUnit : test3CheckEnvoiCheckOK
   Description : Un utilisateur connecté reçoit un message check. Il renvoie un message checkOK. Le test passe si l'autre utilisateur reçoit bien le message checkOK.
   Résultat : Test OK.

4) **Test :** Réception d'un message helloNotOK et contrôle de l'état de la connexion
   Nom de la fonction JUnit : test4CheckEtatConnectionApresHelloNotOk
   Description : Un utilisateur tente de se connecter mais reçoit un message HelloNotOK. Le test passe si le booléen qui contrôle l'état de la connexion est à false.
   Résultat : Test OK.

Test de la classe Model :

À travers ces tests, nous allons vérifier que la classe modèle s'adapte aux différents changements effectués par l'application. Nous allons vérifier que la liste des utilisateurs connectés se met à jour correctement, à travers les différentes connexion et déconnexion d'utilisateurs.

1) **Test :** Ajout d'un utilisateurs après la réception d'un message Hello
   Nom de la fonction JUnit : test0CheckAddConnectedUser
   Description : Un utilisateur connecté reçoit un message hello. Il ajoute cet utilisateur dans sa liste d'utilisateur connecté. Le test passe si le nom de l'utilisateur qui vient de se connecter est présent dans la liste d'utilisateurs connectés.
   Résultat : Test OK.

2) **Test :** Suppression d'un utilisateurs après la réception d'un message Goodbye
   Nom de la fonction JUnit : test1CheckRemoveConnectedUser
   Description : Un utilisateur connecté reçoit un message goodbye. Il supprime cet utilisateur dans sa liste d'utilisateur connecté. Le test passe si le nom de l'utilisateur qui vient de se connecter n'est pas présent dans la liste d'utilisateurs connectés.
   Résultat : Test OK.

3) **Test :** Vérification de la présence d'un utilisateur après avoir reçu un checkOk
   Nom de la fonction JUnit : test2CheckUserConnectedOk
   Description : Un utilisateur connecté reçoit un message checkOk. Cet utilisateur reste présent dans sa liste d'utilisateur connecté. Le test passe si le nom de l'utilisateur qui vient de d'envoyer un checkOk est toujours présent dans la liste d'utilisateurs connectés.
   Résultat : Test OK.

4) **Test :** Vérification de la suppression d'un utilisateur si on a pas reçu un checkOk
   Nom de la fonction JUnit : test3CheckUserConnectedNotOk
   Description : Un utilisateur connecté ne reçoit pas de message checkOk. Cet utilisateur est supprimé dans sa liste d'utilisateur connecté. Le test passe si le nom de l'utilisateur qui n'a pas envoyé un checkOk n'est plus présent dans la liste d'utilisateurs connectés.
   Résultat : Test OK attention ce test est long car demande l'envoi de plusieurs check.

5) **Test :** Vérification permettant de s'assurer que la liste d'utilisateurs connectés n'a pas changé si on reçoit un goodbye d'une personne inconnue
   Nom de la fonction JUnit : test4CheckGoodbyeInconnu
   Description : Un utilisateur connecté reçoit un message goodbye d'un utilisateur inconnu. Le test passe si la liste d'utilisateurs connecté n'a pas changé
   Résultat : Test OK.

Test de la classe l'envoi et réception de fichiers :

Grâce à ces tests nous allons vérifier le bon fonctionnements de la sérialisation des fichiers, leur envoi et leur réception.

1) **Test :** Sérialisation et envoi d'un fichier
   Nom de la fonction JUnit : test1SerializationFileSend
   Description : Un messageFile est sérialisé et envoyé sur le réseau. Le test passe si ces deux étapes se déroulent correctement.
   Résultat : Test OK.

2) **Test :** Vérifie que le type de message reçu est bien un messageFile
   Nom de la fonction JUnit : test2ReceiveFileTypeOK
   Description : Un messageFile est sérialisé et envoyé sur le réseau. Le test passe si le message reçu par l'autre utilisateur est bien un messageFile.
   Résultat : Test OK.

3) **Test :** Vérification du contenu d'un message reçu
   Nom de la fonction JUnit : test3ReceptionFile
   Description : Réception d'un messageFile et vérifie si le contenu de ce fichier est identique à celui envoyé. Le test passe si le contenu est identique.
   Résultat : Test OK.