Travail Pratique 2

Manuel d'utilisation

Michael Munger

Departement d'Informatique et Mathématique

Université du Québec à Chicoutimi

8INF422 : Systemes Distribués

Professor :Hamid Mcheick

17 Novembre 2020

Question 3 – Vente de voiture

Début du programme

Pour débuter le programme, veuillez exécuter Server.java sur la machine servant de serveur, puis exécuter Client.java sur la machine servant de client. Le côté client et serveur doivent avoir accès à Voiture.java et Facture.java. De plus, le côté serveur doit avoir accès a DatabaseHandler.java. Modifier, au besoin, la variable hôte de la commande Socket à la ligne 3 de Client.java avant l'exécution.

Navigation du menu (côté client)

À l'execution de Client.java, une connection Socket est établit avec le serveur et un menu est affiché au client. Le menu est en boucle while et elle ne peut finir que si le client entre 0 au menu. Si le client entre un nombre invalide ou entre autre chose qu'un nombre, un message d'erreur apparait. Juste avant de recommencer la boucle, après l'exécution de l'option choisi, le client entre en attente pour 2500 milisecondes pour permettre à l'utilisateur de lire les données ou le message affichés.

MenuHandler (côté client)

Lorsque l'utilisateur choisi une option, le choix est envoyé à la fonction Handler de la classe MenuHandler(Client.java).

Si l'option 1 est choisi, un objet voiture est créé et envoyé au serveur. Le serveur retourne alors un message exprimant que la voiture a été ajouté ou non et le client imprime ce message à l'utilisateur. La création d'un objet voiture à l'option un est fait du côté client en faisant appel au constructeur de voiture sans paramètre. Ce constructeur fait lui-même appel aux fonctions setters des paramètres du nouvel object voiture. Chaque setter imprime à l'utilisateur une demande de valeur pour le paramètre.

Si l'option 2 est choisi, un objet facture est créé et envoyé au serveur. Le processus est identique à celui de l'option 1, mais il fait appel aux fonctions équivalente dans Facture.java.

Si les options 3 ou 4 sont choisis, un message String est envoyé et serveur et ce dernier retourne une liste (Arraylist) des options disponibles. Le client fait alors appel à sa fonction ArraytoList pour imprimer la liste à l'utilisateur et lui demander son choix. Le client transmet ensuite le choix au serveur, qui lui retourne une Arraylist d'objets voiture. Le client fait ensuite appel a la fonction AfficherVoitureArray pour imprimer l'information des voitures à l'utilisateur. Ceci conclut la boucle donc le menu réapparait.

Si les options 5 ou 6 sont choisit, un message String est envoyé et serveur et l'exécution est équivalente aux options 4 ou 5. Au lieu d'imprimer des voitures, on imprime des factures à l'aide de la fonction AfficherFactureArray.

Si l'option 0 est choisi, un objet Exit (Client.java) est créé, puis il est envoyé au serveur et ferme la boucle while.

Navigation de menu (côté serveur)

À l'execution de Server.java, la fonction ReadFile de la classe MenuHandler est appelé et le serveur ouvre un ServerSocket, puis il se met en attente de client dans une boucle while. Lorsqu'un client se connecte, un thread ClientHandler est créé et la connection y est assignée. On appel la fonction run sur le thread et on recommence la boucle, sauf si la variable exit est à true, alors on arrête le serveur. La boucle while devrait prendre le parametre true au lieu de exit dans le cas ou on veut plus d'un client.

ClientHandler (côté serveur)

Lorsqu'il y a une nouvelle connection, le thread ClientHandler (Server.java) se met en attente du choix du client.

Si le client envoit un objet Voiture au serveur, le client envoit l'objet à la fonction AddVoiture de la classe DatabaseHandler qui lui retourne un String. Ce String de résultat est ensuite retourné au client.

Si le client envoit un objet Facture au serveur, le client envoit l'objet à la fonction AddFacture de la classe DatabaseHandler qui lui retourne un String. Ce String de résultat est ensuite retourné au client.

Si le client envoit un objet String alors on évalue la valeur de l'option choisi et on appel la fonction approprié de DatabaseHandler :

- Option 3: GetSerie
- Option 4 : GetMarque
- Option 5 : GetID
- Option 6 : GetNom

Ensuite, la thread attend une nouvelle String du client. Lorsqu'elle est reçue, on appel la fonction approprié de DatabaseHandler avec le paramètre donné :

- Option 3: VoitureParSerie
- Option 4 : VoitureParMarque
- Option 5 : FactureParID
- Option 6 : FactureParNom

Finalement, l'Arraylist retournée par la fonction est envoyée au client.

Si le client envoit un object Exit, la valeur exit du Serveur devient true et la boucle while prend fin.

DatabaseHandler (côté serveur)

Lors de l'exécution de Server.java, ce dernier fait appel à la fonction ReadFile. En faisant appel à cette fonction, une Arraylist de Voitures et une de Factures sont créées. Puis, on vérifie si voitures.dat et factures.dat existes. Dans le cas ou l'un ou l'autre n'existe pas, les fichiers manquant sont créés et une voiture est créée et ajouté au fichier à l'aide de la fonction CreateReadFile. Après cette vérification, les fichiers sont lus et leur Arraylist d'objets sont sauvegardé dans la mémoire du programme.

Si l'option 1 est choisi, la fonction AddVoiture est appelée. Cette fonction ajoute l'objet Voiture envoyé à l'Arraylist de voitures, puis elle place la nouvelle Arraylist dans le fichier voitures.dat. Si tous ce passe biens, la fonction renvoie un message de succès. Sinon, elle renvoit un message d'erreur.

Si l'option 2 est choisi, la fonction AddFacture est appelée. Le fonctionnement est equivalent à celui de l'option 1.

Si les options 3, 4, 5 ou 6 sont choisis, les fonctions GetSerie, GetMarque, GetID ou GetNom sont appelées respectivement. Ces fonctions ouvrent l'Arraylist de voiture ou de facture, au besoin, et retourne une Arraylist avec toutes les differentes valeurs possibles. Puis, les fonctions VoituresParSerie, VoituresParMarque, FactureParID ou FactureParNom sont appelées respectivement. Ces fonctions ouvrent l'Arraylist de voiture ou de facture, au besoin, et retourne une Arraylist avec tous les objets qui correspondent au critère de recherche donné.