Samdo Ghislain
Karengera Gustave

IFT3325:Rapport TP2

Dans ce projet, il s'agit "de simuler un sous-ensemble du protocole HDLC".
Pour ce faire, on a implementé quatres classes:

-Parser
-Trame
-Sender
-Receiver

Description des classes.

Parser:

Cette classe prend un fichier text et pour chaque ligne du fichier, elle crée une trame (objet trame).
Les méthodes implémentés dans cette classe sont les suivantes:

+getTrame(): Cette méthode ne prends pas de paramètre et elle retourne une liste chaine contenant toutes les trames crées.

+toBin(String): Cette méthode prends une chaine de caractère et elle retourne une chaine de caractère qui est la conversion binaire de la chaine de caractère en entrée (elle permet de convertir une trame en binaire).

Trame:

Cette classe permet de construire une trame sous le format énoncé dans le projet. Elle possède deux constructeurs qui lui permettent de construire la trame de 2 différentes manières:

Trame(String): constructeur qui prend une chaine de caractère et construit la trame en découpant la chaine de caractère.

Trame(String, int, String, String): constructeur qui prend les différents champs de la trame à construire en entrée et construit la trame avec ces champs.

Les méthodes implémentés dans cette classe sont les suivantes:

+getnumeroBi(): Cette méthode ne prends pas de paramètre et elle retourne une chaine de caractère qui est la conversion du champ "*Num"* de la trame.

+getData(): Cette méthode ne prends pas de paramètres et elle retourne une chaine de caractère qui représente le champ "*Données"* de la trame .

+getFlag(): Cette méthode ne prends pas de paramètres et elle retourne une chaine de caractère qui est la conversion binaire du champ "*Flag"* de la trame.

+getCrc(): Cette méthode ne prends pas de paramètres et elle retourne une chaine de caractère qui représente du champ "*CRC"* de la trame.

+getType(): Cette méthode ne prends pas de paramètres et elle retourne une chaine de caractère qui représente du champ *"Type"* de la trame.

+getTypeBin(): Cette méthode ne prends pas de paramètres et elle retourne une chaine de caractère qui est la conversion binaire du champ "*Type"* de la trame.

+getNumero(): Cette méthode ne prends pas de paramètres et elle retourne une chaine de caractère qui représente du champ "*Num"* de la trame.

+toString(): Cette méthode ne prends pas de paramètres et elle retourne une chaine de caractère qui représente la trame construite.

+toStringBi(): Cette méthode ne prends pas de paramètres et elle retourne une chaine de caractère qui est la conversion de tous les champs de la trame construite.

+toBin(): Cette méthode prends une chaine de caractère et elle retourne une chaine de caractère qui représente la conversion binaire de la chaine de caractère en entrée.

+calculCrc(string,string): Cette méthode prends 2 chaines de caractères et elle retourne une chaine de caractère qui représente le "*checksum"* i.e le reste de la division euclidienne de la première chaine de caractère par la deuxième chaine de caractère en entier.

Sender:

Cette classe permet à l'émetteur d'envoyer les trames au récepteur. Elle hérite de la classe TimeTask, ce qui permet à l'émetteur d'avoir un comportement d'un "exécuteur de taches"; c'est à dire qu'à chaque intervalle maximale de 5 secondes, l'émetteur exécute une tache.

Les méthodes implémentés dans cette classe sont les suivantes:

+cancel(): cette méthode ne prends pas de paramètres et retourne un booléen "true"; qui permet d'arrêter les exécutions de l'émetteur à chaque intervalle de 5 secondes.

+send(String, Socket): cette méthode prends 2 paramètres, une chaine de caractère et un socket et elle ne retourne rien. Elle permet d'envoyer la chaine de caractère via le socket.

+affiche(Integer []): cette méthode prends un tableau d'entiers et ne retourne rien. Elle permet d'afficher les éléments du tableau en entrée.

+ecrire(String, String): cette méthode prends 2 paramètres, 2 chaines de caractère et elle ne retourne rien. Elle permet d'écrire la deuxième chaine de caractère dans le fichier qui a pour nom la première chaine de caractère.

+ecouter_client(BufferedReader): cette méthode prends un paramètre, un “tampon” et lit continuellement ce qui se trouve dans le tampon.

+run(): cette méthode ne prends pas de paramètre, et appelle la méthode send() à chaque intervalle de 5 secondes pour renvoyer la trame pour la quelle on a pas reçu d’acquittement

Receiver:

Cette classe permet au récepteur de recevoir les trames et de les traiter.

Les méthodes implémentés dans cette classe sont les suivantes:

+getRandomNumberInRange(Int, Int): cette méthode prends deux entiers et retourne le maximum entre les deux entiers

+conversion(String): cette méthode prends une chaine de caractère en binaire et retourne la conversion de la chaine de caractère en “String”

+getQuit(): cette méthode ne prends pas de paramètres et retourne un booléen qui permet au récepteur de savoir si il doit mettre fin à la communication.

+send(String, Socket): même méthode que celle implémentée du coté de l’émetteur.

+ajouteZero(String, Int): cette méthode prends une chaine de caractère et retourne une chaine de caractère sur le nombre de bits qui correspond à l’entier en entrée.

+calculCrc(String,String): même méthode que celle crée du coté du client.

+gestionTrame(String,Socket,Map): cette méthode prends une chaine de caractère, un socket et une structures Map. Elle ne retourne rien. La chaine de caractère correspond à la trame envoyé par l’émetteur via le meme socket, et si la trame n’est pas erroné, elle est sauvegardé dans la structure map.

+ecouter_serveur(String,Socket, Map): cette méthode prends une chaine de caractère, un socket et une structure Map. Elle appelle la méthode “gestionTrame(.)” et permet de stocker les trames envoyé après l’envoie d’une trame erronné.