Lola Drouin INFO5

Spécification

Class Borker :

Broker :
Constructor of the class. Uses a string representing the name of the object to create it.
Parameters :
name (string) – the name of the broker

accept :
Listen for a connection to be made on the given port and accepts it. The method blocks until a connection is made.
Parameters:
port (int) – the port on which the connection is made.
Returns :
the channel
Exception :
IOException – if an I/O exception of some sort has occurred.

connect :
Connects to the broker corresponding to the given name on the given port.
Parameters:
port (int) – the port on which the connection is made.
name (string) – the name of the broker we want to connect to.
Returns :
the channel
Exception :
IOException – if an I/O exception of some sort has occurred.

Class Channel :

read :
Reads a maximum of « length » bytes from the input stream and put them into the given byte array « bytes ». The first byte read is stored in « bytes[offset] », the second in « bytes[offset+1 », ect… until « bytes[offset+lenght-1] ». If there is not enought bytes to read, the k bytes read will be stored in « bytes[offset] » through « bytes[offset+k-1] » and the end of « bytes » will not be filled.
Parameters:
bytes (byte[]) - the buffer in which the data is read.
offset (int) - the start offset in array bytes.
length (int) - the maximum number of bytes to read.
Returns:
the total number of bytes read in the buffer (Comprit entre 0 et length)
Exception :
NullPointerException – if bytes is null.
IndexOutOfBoundsException – if offset or lenght is negative or if offset+length > the lenght of the array bytes.
IOException – if an I/O exception of some sort has occurred.

write :
Writes a part from the given byte array (« bytes ») to the outpout stream. This part corresponds to the bytes from the indice « offset » to the indice « offset »+« length ».
Parameters:
bytes (byte[]) - the buffer containing the data to write.
offset (int) - the start offset in array bytes.
length (int) - the maximum number of bytes to write.
Returns:
the total number of bytes write in the buffer
Exception :
NullPointerException – if bytes is null.
IndexOutOfBoundsException – if offset or lenght is negative or if offset+length > the lenght of the array bytes.
IOException – if an I/O exception of some sort has occurred.

disconnect :
Disconnects the two brokers involved in the channel so closes the channel.

Disconnected :
Enables to know if the channel is connected or not. If the channel is deconnected return true.
Returns :
A boolean indicating if the channel is disconnected

Résumé de correction :

Les méthodes sont abstract. Pour utiliser les méthodes des classes abstraites, utiliser une sous classe concrète de celles ci.
La classe Channel correspond à un moyen de communication entre 2 taches. Un channel est un flux d'octets FIFO lossless.
La classe Broker permet de créer ce channel

Channel :
- La méthode Read se bloque tant qu'elle n'a rien à lire. Lorsqu'il y a quelque chose à lire, on lit ce qu'on peut. L'entier length représente donc le nombre maximum d'octets qu'il est possible de lire mais ce maximum n'est pas forcement atteint.
Lors d'un problème on lève une exception I/O.Dans le traitement de l'exception, on ferme la connexion.
- La méthode Write ne bloque pas, elle retourne directement ce qu'elle a et n'attend pas d'avoir écrit length octets.
Lors d'un problème on lève une exception I/O.Dans le traitement de l'exception, on ferme la connexion.
Dès qu'un channel se rend compte que l'autre s'est déconnecté, il lève une exception et arrête ce qu'il faisait. Écrire puis fermer ne garanti donc pas la lecture des données.

Broker : gère les channels, 1 tâche crée 1 broker avec 1 nom
- Accept : Le broker se met en attente d'une connexion sur le port précisé.

- Connect : Connexion à un autre Broker dont le nom est donné sur le même port précisé.
Ces deux méthodes sont bloquantes en attendent qu'un rendez-vous soit accepté , le channel partagé est alors créé.

Les classes Broker et Channel peuvent être partagées par des threads mais c'est à l'utilisateur de gérer le contrôle de concurrence.

Design:

Class CircularBuffer : La classe correspond à une file d'attente dans laquelle seront stockés les octets afin de permettre la communication entre les deux channels.

CircularBuffer :
Constructeur de la classe. Permet de construire un CircularBuffer d'une capacité donnée.
Paramètre :
- capacity (int) : la capacité du buffer crée

full :
Return : t
rue si le buffer est plein

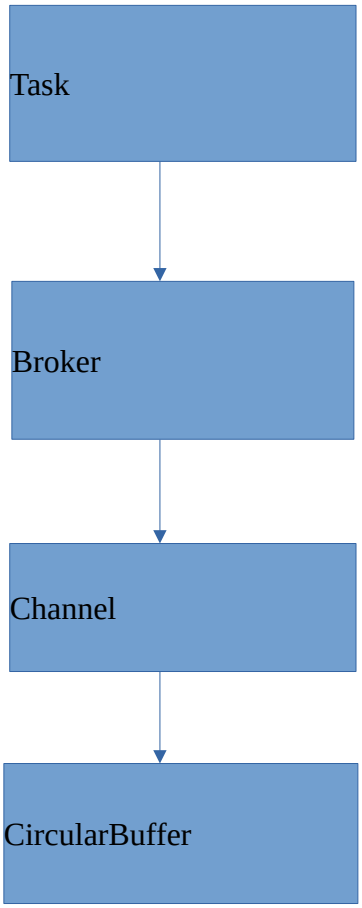
empty :
Return :
true si le buffer est vide

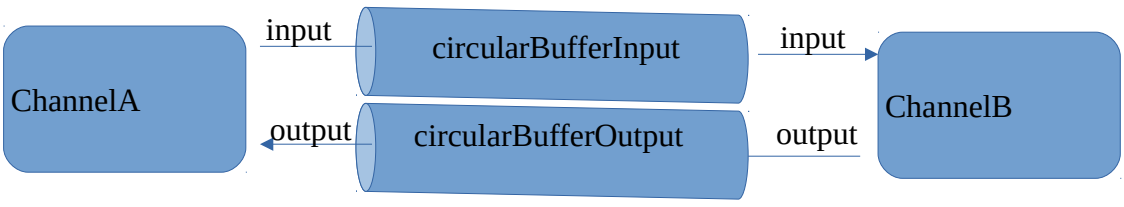
push :
Ajoute un octet au buffer si celui ci n'est pas plein.
Paramètre :
- elem (byte) : l'octet à ajouter au buffer

Exception :
- IllegalStateException – si le buffer est plein

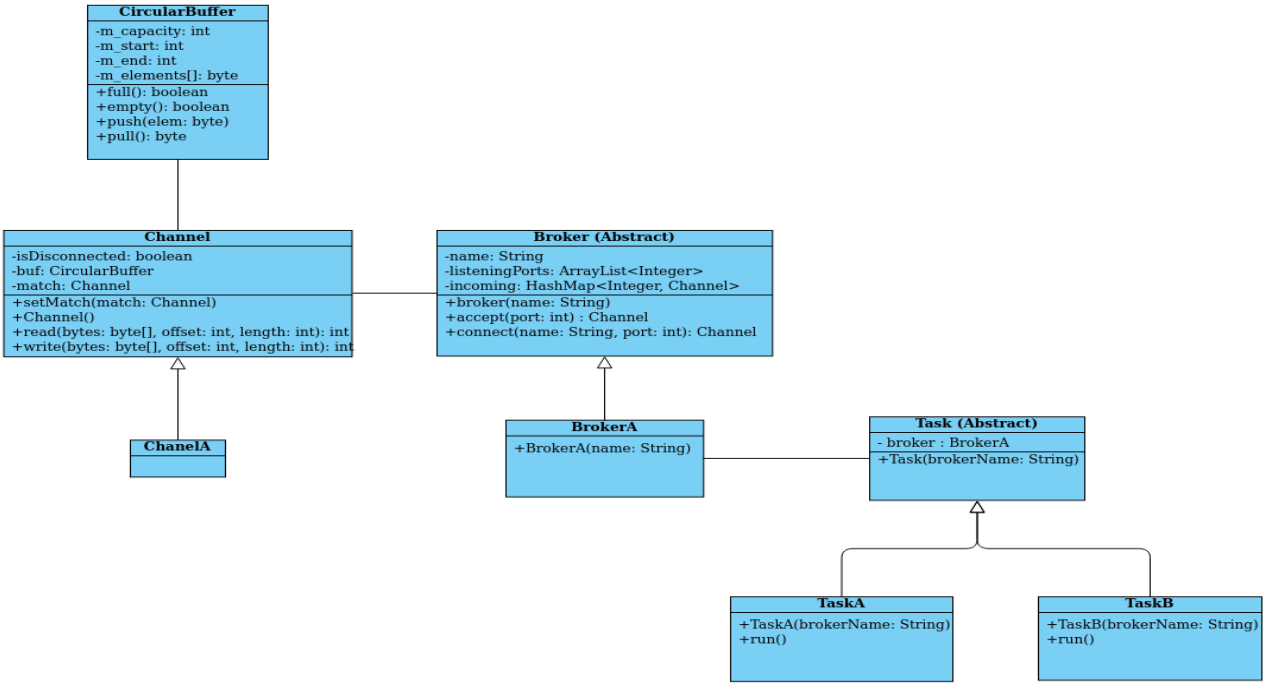
pull :
Enlève un octet au buffer si celui ci n'est pas vide.
Return:
Le prochain octet disponible
Exception :
- IllegalStateException – si le buffer est plein

Lola Drouin INFO5





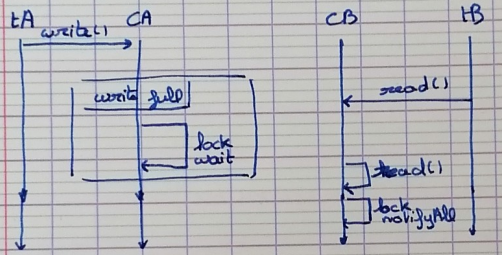
Lola Drouin INFO5



Lola Drouin INFO5

Note de la correction faite en cours

abstract pour implém fct de façon ≠ (ex accept)
pas modif classe abstraite → modif ds ss classe
pas de ramasse miette → besoin d'avoir channelManager pour savoir qd et quoi nettoyer / libérer. ~~Ici pas besoin~~
↳ si broker pas de ref à channel perd channels qd task disconnect
s'assurer pas de ref à channel appart pour qu'ils soient garbage collection
B put ds buffA et A fait 1 get
buffA peut être plein → put que si pas plein
→ si plein en attente que sa read
si write rien → bloc



th connaissance mémoire
Pas besoin synchro ds circularBuffer (pour 1 producteur 1 consommateur) mais doit pvr expliquer sinon synchro obligatoire

Au nv broker
vérifier y a pas 2 broker avec même nom (nom unique)
table de hash < nom, Broker >
lors du connect → cherche ds hashMap
↳ si trouve pas → soit bloc mais mettre timeout sinon bloc à ∞
→ soit le connect revient + tard

1er qui arrive se bloc, le 2e débloc → comment ?
(+ grosse granularité) • 1 obj partagé qui notifyAll → tt le monde se réveil et re-teste si condit° ok
↳ c broker manager (chapeau mieux que static)
(+ fine granularité) • ~~thread se bloc~~ qd on se lock créé 1 objet lock pour ce nom là et ce port
1 hashmap donne obj rdv si 1/accept connect crée cet objet et se met en wait dessus
le 1/connect arrive et débloc
accept

Lola Drouin INFO5