Implantation d'un un serveur de tâches apériodiques

Nous cherchons à améliorer le comportement général du système, pour cela nous allons utiliser une tâche particulière qui sera chargée de servir les requêtes apériodiques: Le serveur de tâches apériodiques

Trois type de politiques peuvent être prises en compte: « Background », « Polling », « Defferable ».

Politique Background Server

Le principe de fonctionnement de la politique de service Background Server repose comme son nom l'indique, sur le traitement des tâches apériodiques en tâche de fond. Cela se traduit par l'affectation d'une plus faible priorité à ces tâches, ce qui entraine leurs traitements lors des temps libres du système. Ainsi cela permet de ne pas interférer avec le système et l'exécution des tâches périodiques, cependant il ne fournit aucune garantie quant au traitement des tâches apériodiques. Il se peut alors, dans certains scénarios, que ces tâches ne soient jamais traitées vu que leurs ressources ne sont pas réservées.

Politique Polling Server

Le Polling Server sert les évènements en attente tant qu'il possède de la capacité. S'il n'y a pas ou plus d'évènement en attente, sa capacité tombe à 0. A chaque période, il récupère sa capacité maximale correspondant à sont coût.

Politique Defferable Server

Comme pour le Polling Server, nous allons utiliser une tâche particulière qui sera chargée de servir les requêtes apériodiques. A la différence du Polling Server, le Defferable Server, sa capacité ne tombe pas à 0 lorsqu'il n'y a pas ou plus de tâches apériodiques en attente, ce qui va offrir un meilleur temps de réponse aux tâches apériodiques.

Travail a effectuer

On modélisera les événements "servables" par la classe ServableAsyncEvent, leurs handlers par la classe ServableAsyncEventHandler et un serveur par la classe abstraite TaskServer.
Les classes à réaliser : ServableAsyncEvent, ServableAsyncEventHandler, TaskServer

Rappel avant l'implantation ...

Pour modéliser un évènement apériodique, RTSJ dispose de la classe AsyncEvent. Cette classe nous permet de créer un objet représentant un événement apériodique. Grâce à cette classe, nous pouvons ajouter ou supprimer un handler à l'évènement, positionner des paramètres d'activation et demander le lancement de la tâche.
Un code est associé à l'évènement apériodique par l'intermédiaire de la classe AsyncEventHandler. Le AsyncEventHandler par l'intermédiaire de sa méthode handleAsyncEvent va contenir la portion de code qui sera exécuté au traitement de la tâche.

Des évènements servables (objet ServableAsyncEvent)

Quelques réflexions avant implantation

La classe ServableAsyncEvent doit-elle hériter de AsyncEvent ?

La classe ServableAsyncEvent extends la classe AsyncEvent. Pour cela il faudra surcharger la méthode addHandler afin de prendre en argument un ServableAsyncEventHandler au lieu d'un AsyncEventHandler à l'origine, et redéfinir la méthode fire() pour permettre l'activation de la tâche.

La classe ServableAsyncEventHandler doit-elle implanter l'interface Schedulable ?

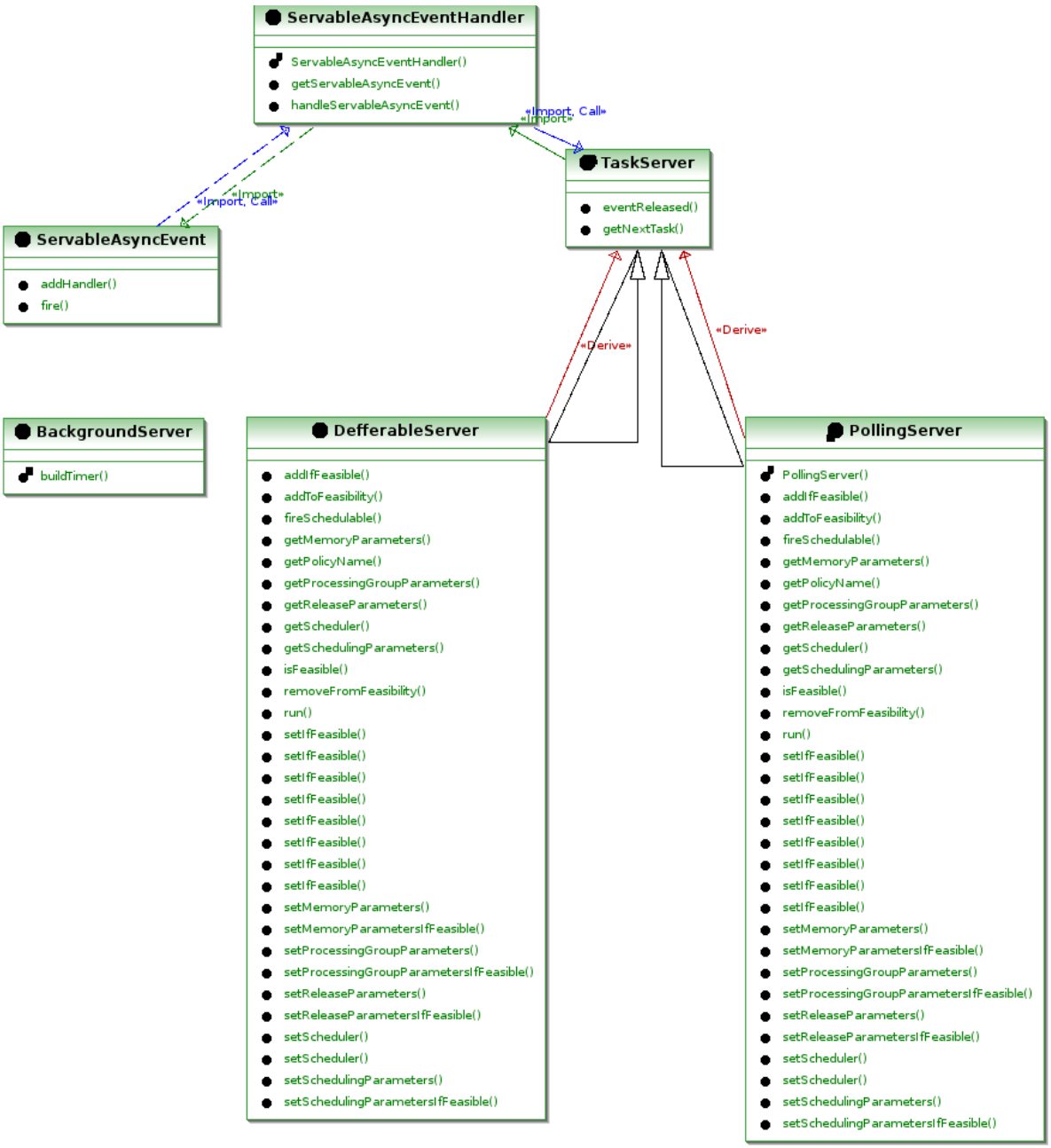
Non, cette classe ne doit pas étendre l'interface Schedulable car cet objet est associé à un serveur de tâche qui implémentera cette interface. Cette classe représente un handler qui peut être associé à un

ServableAsyncEvent. Cet objet sera associé à un serveur de tâche unique et lorsque l'évènement auquel il est associé aura été activé, il sera ajouté à la liste des évènements en attente du serveur de tâche.

La classe ServableAsyncEventHandler doit-elle hériter de RealtimeThread ? De AsyncEventHandler ? De Servable AsyncEvent ? De Interruptible ?
La classe ne doit pas hériter de RealtimeThread car nous utilisons un objet Schedulable qui est le task server. L'utilisation combinée de ces deux objets ne nous permettra pas de dégager de bon résultat.

Architecture globale (proposition)

TaskServer doit être une classe abstraite
La classe TaskServer doit être abstraite car c'est la classe qui va servir de base commune aux implantations des différents serveurs (BackgroundServer, PollingServer et DefferableServer). C'est-à-dire que les serveurs étendront cette classe.
Cette classe va fournir les méthodes de base nécessaires aux différents serveurs, telles que l'ajout de handler, la récupération de la file d'attente des tâches, le lancement du serveur, etc.
De quelle classe hérite-t-elle et quelle interface implante-t-elle ?
Cette classe hérite de Scheduler et implante l'interface Schedulable.
A quelle classe de RTSJ, le Polling Server doit-il déléguer son comportement ?
Le Polling Server doit délèguer son comportement à la classe RealTimeThread.
Quelle priorité doit avoir le serveur pour pouvoir mesurer sa consommation de capacité ?
Le serveur doit avoir la priorité maximum pour pouvoir mesurer sa consommation de capacité.
Comment faire pour que les tâches apériodiques qui respectent leur coût ne soient pas interrompues ?
Il faut prévoir que le coût de la tâche apériodique soit inférieur au coût du PollingServer.


ServableAsyncEventHandler
ServableAsyncEventHandler()
getServableAsyncEvent()
handleServableAsyncEvent()
«Import, Call»
«Import»
TaskServer
eventReleased()
getNextTask()
«Import»
«Import, Call»
ServableAsyncEvent
addHandler()
fire()
«Derive»
«Derive»
BackgroundServer
buildTimer()
DefferableServer
addIfFeasible()
addToFeasibility()
fireSchedulable()
getMemoryParameters()
getPolicyName()
getProcessingGroupParameters()
getReleaseParameters()
getScheduler()
getSchedulingParameters()
isFeasible()
removeFromFeasibility()
run()
setIfFeasible()
setIfFeasible()
setIfFeasible()
setIfFeasible()
setIfFeasible()
setIfFeasible()
setIfFeasible()
setIfFeasible()
setMemoryParameters()
setMemoryParametersIfFeasible()
setProcessingGroupParameters()
setProcessingGroupParametersIfFeasible()
setReleaseParameters()
setReleaseParametersIfFeasible()
setScheduler()
setScheduler()
setSchedulingParameters()
setSchedulingParametersIfFeasible()
PollingServer
PollingServer()
addIfFeasible()
addToFeasibility()
fireSchedulable()
getMemoryParameters()
getPolicyName()
getProcessingGroupParameters()
getReleaseParameters()
getScheduler()
getSchedulingParameters()
isFeasible()
removeFromFeasibility()
run()
setIfFeasible()
setIfFeasible()
setIfFeasible()
setIfFeasible()
setIfFeasible()
setIfFeasible()
setIfFeasible()
setIfFeasible()
setMemoryParameters()
setMemoryParametersIfFeasible()
setProcessingGroupParameters()
setProcessingGroupParametersIfFeasible()
setReleaseParameters()
setReleaseParametersIfFeasible()
setScheduler()
setScheduler()
setSchedulingParameters()
setSchedulingParametersIfFeasible()