Programmation fonctionnelle en Java
(NFP101)

L'interface `Future<T>`

L'interface `Future<T>` de la JDK représente un calcul qui doit ultimement (à un moment dans le futur) renvoyer un résultat de type `T`. L'interface `RunnableFuture<T>` de la JDK étend en plus l'interface `Runnable`, et la classe `FutureTask<V>` implante cette interface.

Si `e` est une expression de type `T`, on peut créer un tel calcul en utilisant simplement le constructeur ainsi `new FutureTask(() -> e)`. Ce calcul peut alors être exécuté de manière concurrente.

On peut aussi stocker ce calcul en cours d'execution, par exemple dans une variable `x`, et il est possible ensuite de récupérer le résultat avec l'expression `x.get()`, en précisant éventuellement le temps que l'on est prêt à attendre pour obtenir le résultat. La méthode `get()` qui est une opération bloquante.

Questions

Ecrire une classe `FutureTask<T>` qui implante l'interface `RunnableFuture<T>` en utilisant les threads et les `BlockingQueue<E>` pour réaliser la communication entre une instance de `FutureTask<T>` et le serveur qui réalise effectivement de calcul (le calcul sera lancé lorsque `run` est invoqué).

Les méthodes à implanter sont les suivantes :

`boolean` `cancel(boolean mayInterruptIfRunning)`

Attempts to cancel execution of this task.

`V` `get()`

Waits if necessary for the computation to complete, and then retrieves its result.

`V` `get(long timeout, TimeUnit unit)`

Waits if necessary for at most the given time for the computation to complete, and then retrieves its result, if available.

`boolean` `isCancelled()`

Returns `true` if this task was cancelled before it completed normally.

`boolean` `isDone()`

Returns `true` if this task completed.