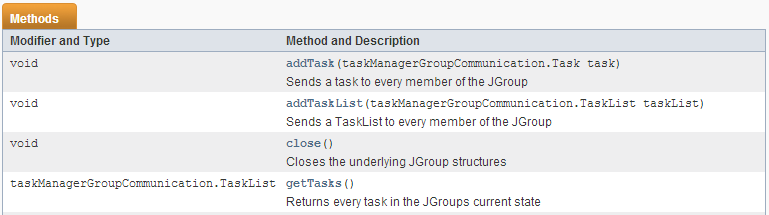
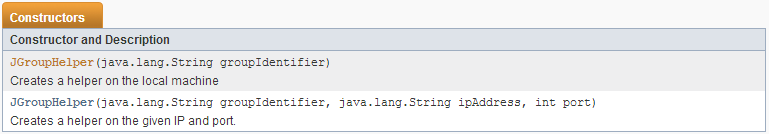


In this assignment we’ve made a TaskManager application, which utilizes some basic concepts of group communication.

We decided to model our application based on the proposed structure “TaskManager Application” from the assignment. This consists of three TaskManager clients, who handles synchronization by using the JGroups toolkit.

**JGroupHelper**We’ve implemented an abstraction over the JGroup toolkit in the form of a JGroupHelper class with only a few public methods as shown in the documentation below.  
This helper class makes it easy to implement and maintain new Task Manager Clients as these do not need to know anything about the specifics of the JGroups toolkit.



Inside the JGroupHelper we’ve implemented the JChannels receiver as a private inner class. We also make use of Raos ChannelHelper code snippet.

Our helper (and hence our TaskClient) does not support DELETE, as this was not a requirement for the assignment. We chose to focus on getting the JGroups functionality to work.

**TaskClient – the client**  
Our client is a simple console application and is largely a result of what happens when trying to make an otherwise boring task more exciting. Yes, the client is rude. Apart from that, it should be quite easy to follow. It supports three basic commands: GET, ADD and ADDLIST.

GET gets the TaskList state and then prints all tasks.

ADD walks you through adding a task. Behind the scene it creates a Task object and sends it to the group.

ADDLIST is a secret sabotage keyword. As it would make no sense to have the user type in several tasks and then add them to a tasklist and send them (it would be easier to continuously ADD), it has already created a TaskList, which is sent to the group.

Running two clients at once illustrates the functionality: If you add a task on one client, it is also added on the other.