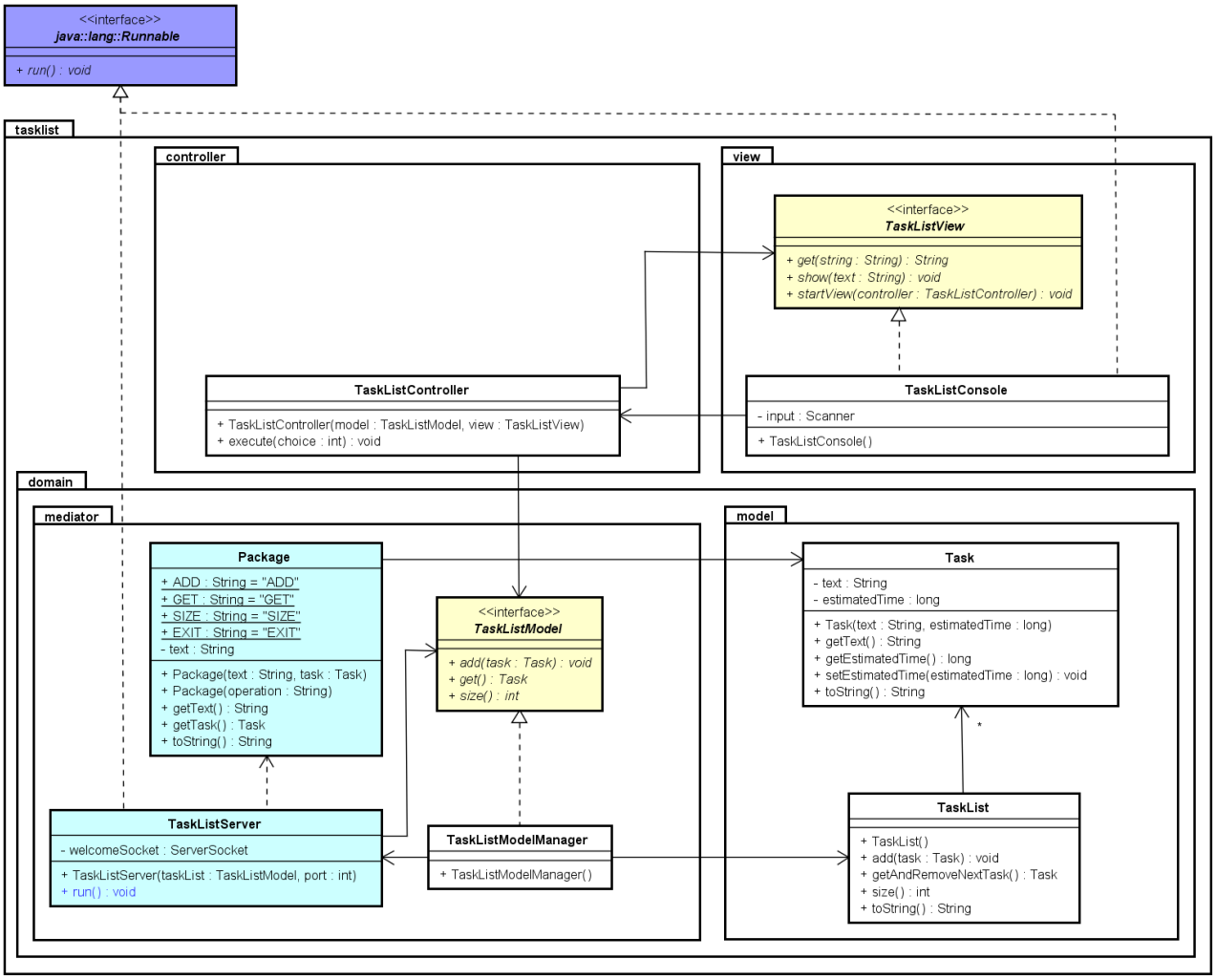
Exercise 11.01 – Server side of a Task List

The purpose for the exercise is to combine a client/server version and an MVC version of the task list (i.e. exercise 6.03 and exercise 10.01).

Step 0 – Eclipse (or any other IDE)

Use the MVC version as basis. Create a new projects in Eclipse copying all files from exercise 10.01 (the single user task list in MVC) and name the project something to make it easy for you to see that this includes a server.



Note: When using exercise 10.01 as basis (MVC version of a single user task list) only the model manager need to be changed. The view, the controller and the model in package `tasklist.domain.model` are already done.

Server side: Step 1 – Socket server

Copy classes from the server side of socket exercise 6.03: `Package`, `TaskListServer` and `TaskListCommunicationThreadHandler` into package `tasklist.domain.mediator` and modify them the following way

Class `TaskListCommunicationThreadHandler`:

- Change instance variable of type `TaskList` to be of type `TaskListModel`. This also affect the constructor and methods called on this variable (use instead methods `add`, `get` and `size`)

Class `TaskListServer`:

- Let this class implement interface `Runnable` and rename method `execute` to `run`
- Change instance variable of type `TaskList` to be of type `TaskListModel`. This also affect the constructor and calling `TaskListCommunicationThreadHandler`

Server side: Step 2 – TaskListModelManager (mediator package)

Modify class `TaskListModelManager`:

- Add an instance variable of type `TaskListServer`
- Add initializing of this variable in the constructor:
    1) Create a `TaskListServer` object with `this` and a port, e.g. 6789 as argument,
    2) Create a thread with the `TaskListServer` object as argument
    3) Start the thread (i.e. starting the server in its own thread)

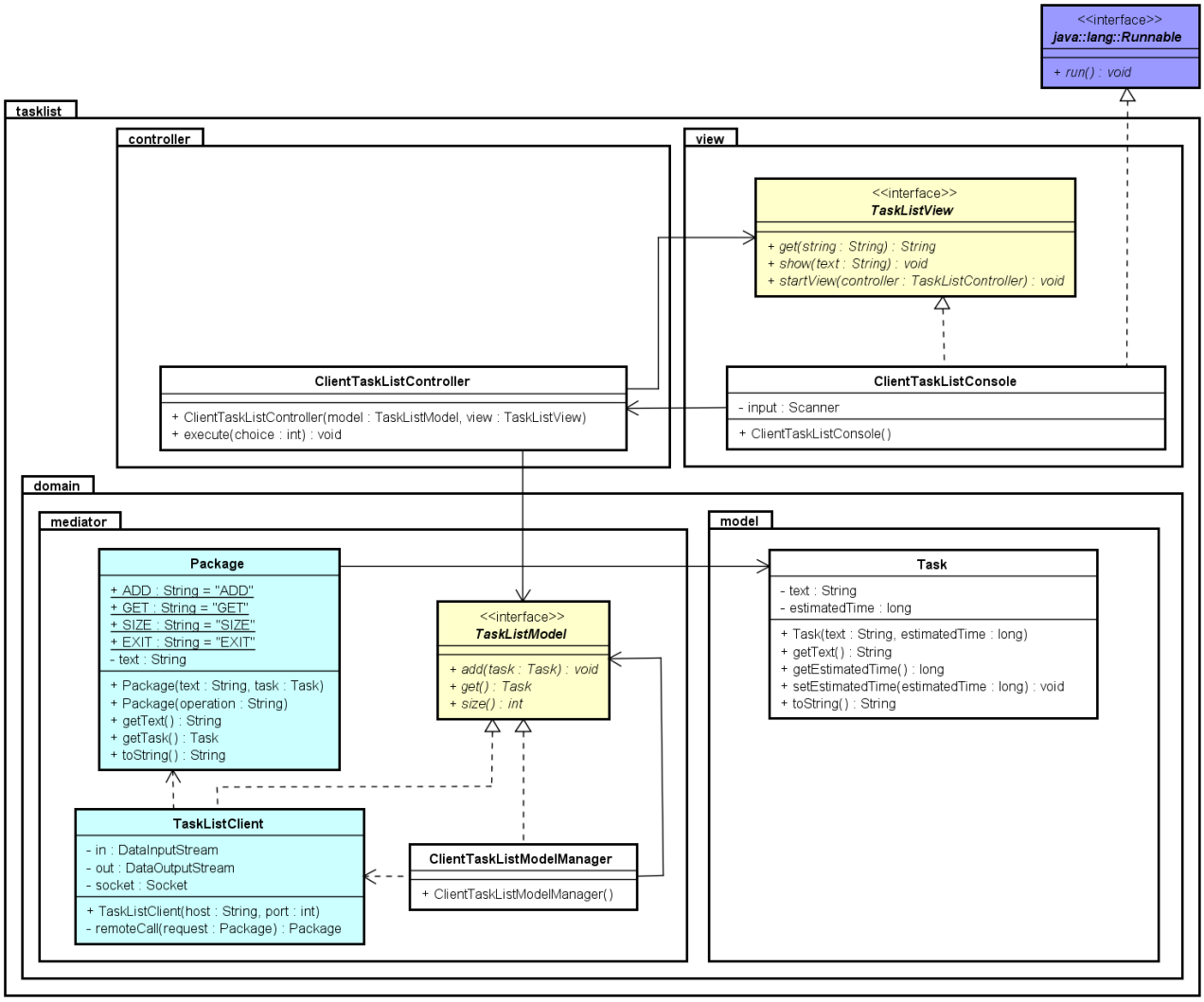
Server side: Step 3 – Main method

This is done.

Test your solution running the main method (which will run the server). See if you could add task's from the server. Run the client from exercise 6.03 and see if this will be connected to the server and if a client may add, get task's and get the size.

Exercise 11.02 – Client side of a Task List

The purpose for the exercise is to combine a client/server version and an MVC version of the task list (i.e. exercise 6.03 and exercise 10.01).



Step 0 – Eclipse (or any other IDE)

The easiest way is to create a new projects in Eclipse copying all files from the server side version you made in the previous exercise and name the project something to make it easy for you to see that this includes a server.

Delete the two server classes

- Delete class `TaskListServer`
- Delete class `TaskListCommunicationThreadHandler`

To distinguish between server and client classes make the following renaming

- Rename class `TaskListController` to `ClientTaskListController`
- Rename class `TaskListConsole` to `ClientTaskListConsole`
- Rename class `TaskListModelManager` to `ClientTaskListModelManager`
- Rename class `Main` to `ClientMain`

Client side: Step 1 – Socket client

Copy class `TaskListClient` from the client side of socket exercise 6.03 and keep class `Package` both in package `tasklist.domain.mediator` and modify class `TaskListClient` the following way

- Let this class implement interface `TaskListModel` (`add`, `get` and `size` methods)
- Implement a private method taking a `Package` object (request) and return a reply `Package` object. Copy the part converting the package to Json, writing it to the server, reading a reply from server, converting this from Json to a Package and return this.
- Implement the three methods from the interface. In each method create the proper `Package` object, call the private method and for methods `get` and `size` return the correct part of the package (in size you have to convert to int e.g. using method `Integer.parseInt`)
- Delete the `Scanner` instance variable and delete method `execute`.

Client side: Step 2 – Model

- Delete the model class `TaskList` (because the client is not storing information locally)
- Modify class `ClientTaskModelManager`
    - Only one instance variable and of type `TaskList`
    - Initialize the instance variable in the constructor to an object of type `TaskListClient`, for now just hardcode `"localhost"` and the port you used on the server side.
    - Methods `add`, `get` and `size` just calls the same methods on the instance variable. The methods don't need to be synchronized because only one thread at the time is calling these methods.

Client side: Step 3 – Main method

This is done (check that you are using the renamed client model manager, view and controller classes).

Test your solution running the server from the previous exercise and run a few client's and see if this will be connected to the server and if a client may add, get task's and get the size.