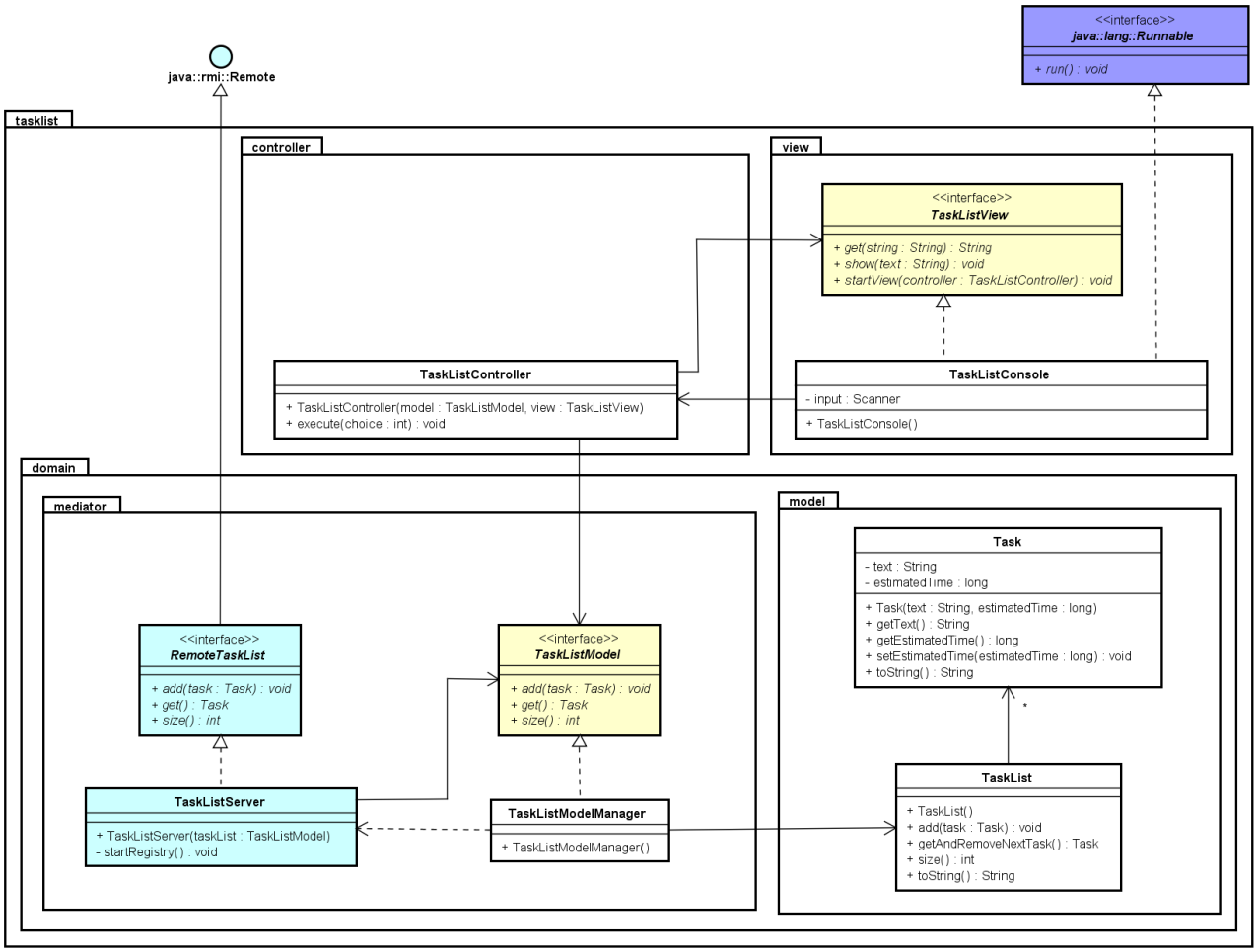
Exercise 11.03 – RMI Server side of a Task List

The purpose for the exercise is to make an RMI version of the task list in an MVC design pattern.



Step 0 – Eclipse (or any other IDE)

Use the socket-MVC version as basis (exercise 11.01/11.02). Create a new projects in Eclipse copying all files from exercise 11.01 (the socket server task list in MVC) and name the project something to make it easy for you to see that this includes an RMI server.

Delete the two socket related classes

- Delete class `Package`
- Delete class `TaskListCommunicationThreadHandler`

Server side: Step 1 – Model to be used in RMI

Define a `Remote` interface `RemoteTaskList` including the methods you want to open up for clients (in this example we just use the same three methods as in the model interface):

```
package tasklist.domain.mediator;

import java.rmi.Remote;
import java.rmi.RemoteException;

import tasklist.domain.model.Task;
```

```
public interface RemoteTaskList extends Remote
{
   void add(Task task) throws RemoteException;
   Task get() throws RemoteException;
   int size() throws RemoteException;
}
```

Note: Objects of type `Task` are used to in two of the methods and thus used in RMI and therefore this class needs: `implements Serializable`.

Server side: Step 2 – RMI server

Modify Class `TaskListServer`:

- Only one instance variable – this is of type `TaskListModel`.
- Let the class implement interface `RemoteTaskList`. All three methods just delegates the work to the instance variable.
- Delete the implementation of `Runnable` and the `run` method.
- The constructor tries to start up the registry, export the object, upload the stub to the registry and bind it to a name (exporting the object either extending `UnicastRemoteObject` or calling the static method `UnicastRemoteObject.exportObject(this, 0)`

Server side: Step 3 – TaskListModelManager

Modify class `TaskListModelManager`:

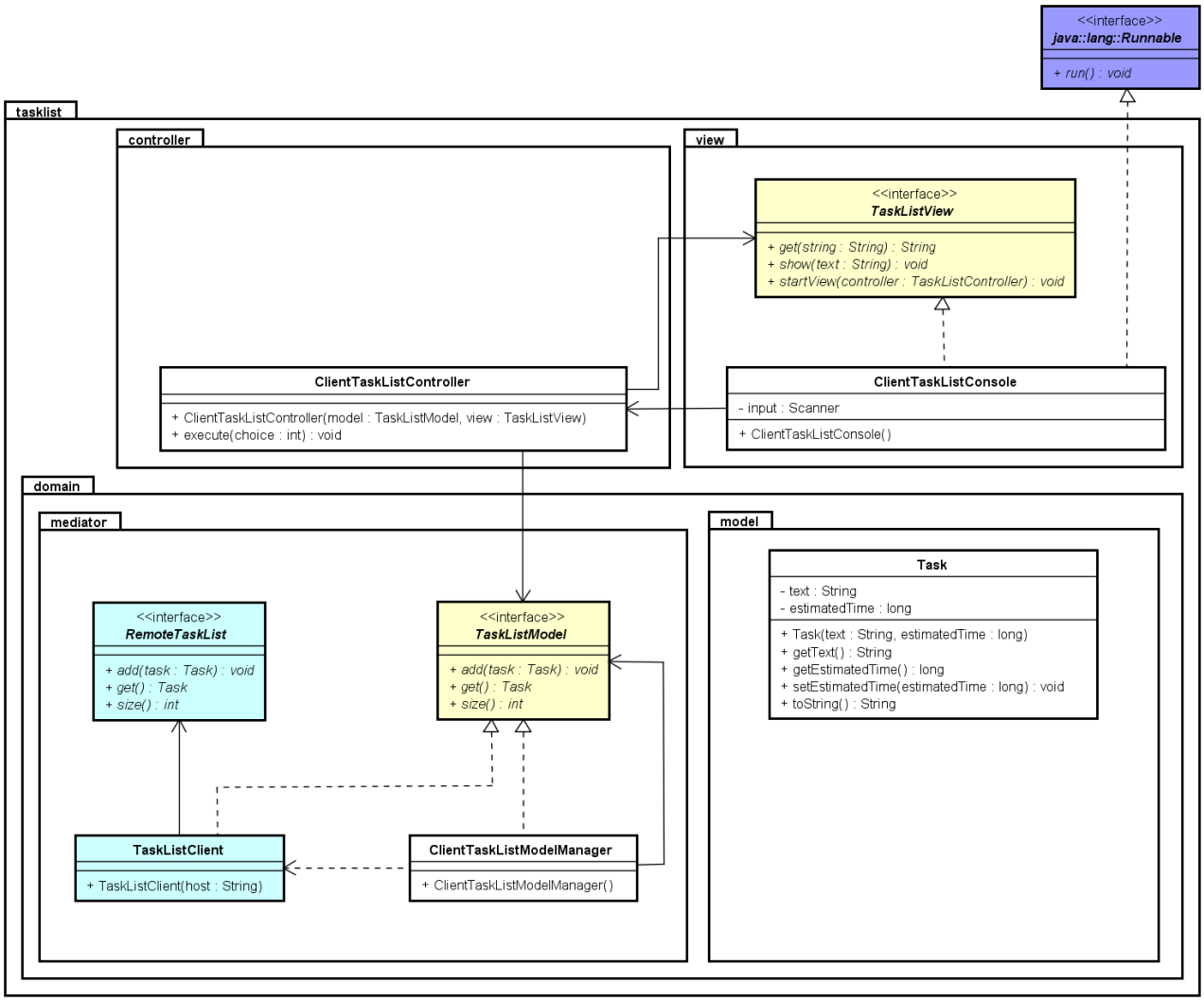
- Only one instance variable (because the model is tiny) – this is of type `TaskList`
- Constructor: initialize the task list and in a try-catch block initialize the `TaskListServer` object. Delete the thread part (because RMI automatically creates threads)

Server side: Step 4 – Main method

This is done.

Exercise 11.04 – RMI Client side of a Task List

The purpose for the exercise is to make an RMI version of the task list in an MVC design pattern.



Step 0 – Eclipse (or any other IDE)

Use the client part of the socket-MVC version as basis (exercise 11.02). Create a new projects in Eclipse copying all files from exercise 11.02 (the socket client task list in MVC) and name the project something to make it easy for you to see that this includes an RMI client.

Delete the socket related class

- Delete class `Package`

Client side: Step 1 – Model to be used in RMI

Copy the interface `RemoteTaskList` from the server version (previous exercise). Either copy the class `Task` from the same exercise or add `implements Serializable`.

Client side: Step 2 – RMI client

Modify Class `TaskListClient`:

- Only one instance variable – this is of type `RemoteTaskList`.
- Keep on implementing interface `TaskListModel`. Change the implementation such that all three methods just delegates the work to the instance variable (in a try-catch block).

- The constructor initializes the instance variable with a lookup in the registry.

Client side: Step 3 – ClientTaskListModelManager

Modify class `ClientTaskListModelManager`:

- Only one instance variable (because the model is remote) – this is of type `TaskListModel`
- Constructor: in a try-catch block initialize to a `TaskListClient` object e.g. with host equal to `"localhost"`.
- Methods from the interface delegates to the instance variable.

Client side: Step 4 – Main method

This is done.