Software Engineering Lab 5 Report

1 Document Information

1.1 List of Authors

Group/Team	Names
<Creation>	<Hilary Ogalagu>
<Your team name>	<Akter Rokaia>

1.2 Versions

Version	Date	Author	Notes
0.1	28.05.2021	<Hilary Ogalagu>	Further research and understanding of the prep task with critical thinking of the modalities involved.
0.2	29.05.2021	<Hilary Ogalagu>	Creating the prototype with Bluej and trying to integrate the observer pattern in regards to my class diagram and sequence diagram.
0.3	30.05.2021	<Hilary Ogalagu>	Testing and working on GUI to make the prototype ready and testing if the buttons are functional.
0.4	05.06.2021	<Hilary Ogalagu>	Demonstration of the UC description with the selected design pattern.
0.5	05.06.2021	<Hilary Ogalagu>	Client-Server RMI design, MON based design
0.6	21.06.2021	<Hilary Ogalagu>	MON based design

Content

Glossary

The following domain-specific terms are used throughout the document.

Term	Abbreviation	Description	Reference
Route on map	ROUTE	<how it is defined in other terms>	<Website or App>
etc.			

2 Demonstrator (Partner A)

<Prep task #2: short documentation of your vertical prototype application>

<Prep task #3: comparison: name agreements and deviations in comparison with the corresponding sections of your lab 4 report>

Observer Pattern: UC5: DisplStatus

Classes	Agreements with Lab 4 report	Deviations with lab 4 report
Viewer Class	There was no need for this class as the Controller provides all the functionalities.	There was no need for a viewer class in the lab5 demo so there was a need to remove this class for a better improved code.
RobotStatus Class	This is in total agreement as it provides the state of the robot system and also the Observable.	No deviation.
Checker Class	This abstract class serves as the Observer, which get updates of all the other classes.	No deviation.
Location	This was not in agreement with my lab 4 report.	There was a need for new class of location that notifys the observer of the current location of the robot.
JobCompleted	Not in agreement with lab 4 report.	There was a need for new class of JobCompleted that notifys the observer of any new job done or completed by the robot.
Battery	Not in agreement with lab 4 report.	There was a need for new class of Battery that notifys the observer of Battery status and if there is a need for battery charging.

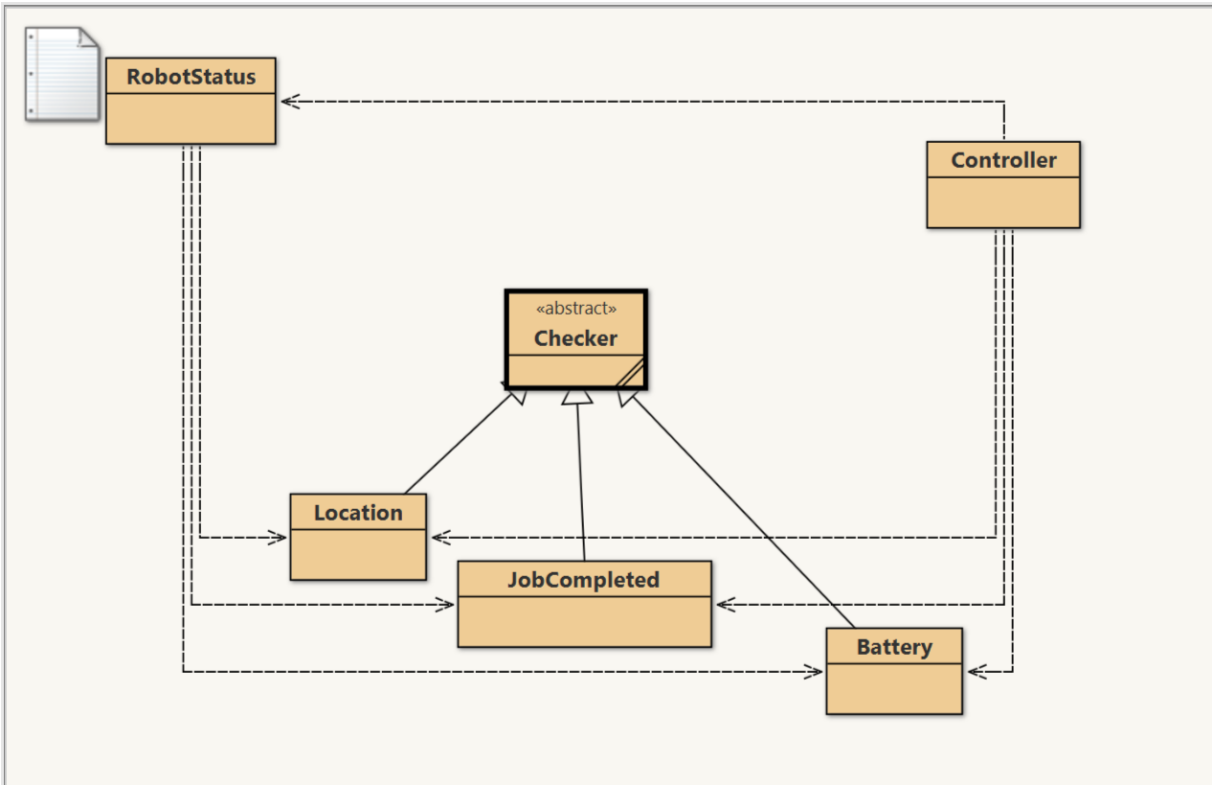
3 Demonstrator (Partner B)

<Prep task #2: short documentation of your vertical prototype application>

<Prep task #3: comparison: name agreements and deviations in comparison with the corresponding sections of your lab 4 report>

4 Distributed Prototype Description

<Problem Overview Lab task: Which demonstrator will fit to which of your lab 2 detailed UC descriptions? Update the two (or more) corresponding detailed UC descriptions to form your distributed prototype>



I decided to go with DisplayStatus demonstrator which fits in for the lab 2 detailed UC description. Considering the classes and in relation with the Observer pattern. Which gets update of all the activities in every individual classes. This demonstrates how the UC in question showing how the activities and how the communication of the classes works.

5 Client-Server RMI Design

< Lab task #2 and #3 first part: Describe the tasks of your prototype client and server resp.>

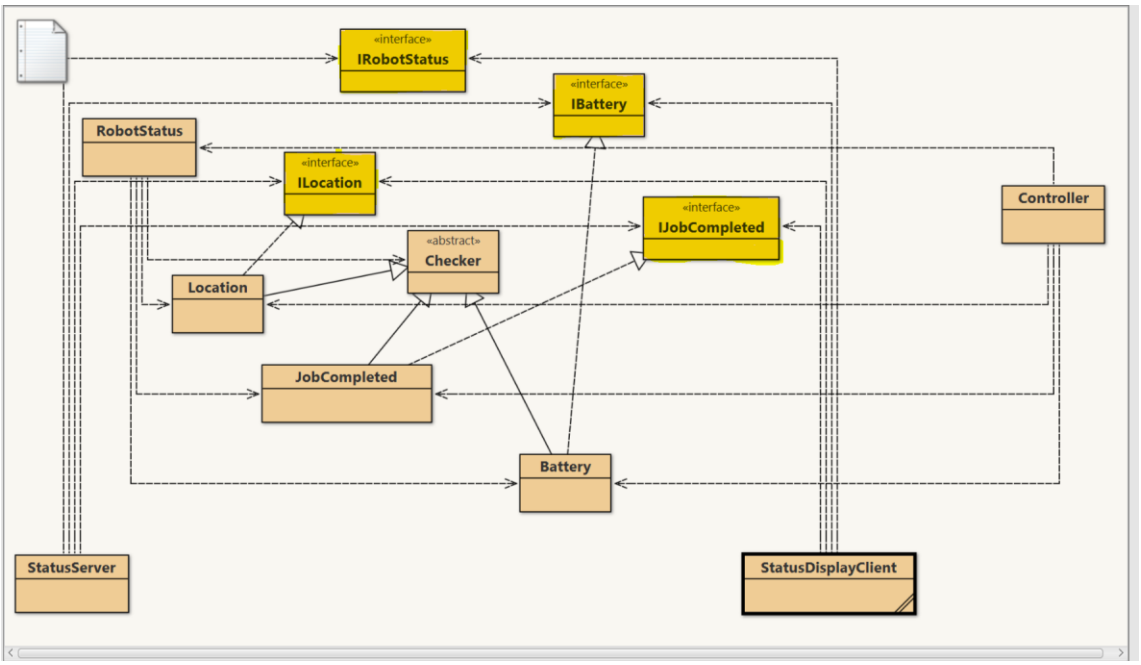
<Lab task #3 part 2: Table of classes and interfaces with a column indicating if it is a remote interface, a client-side or server-side class and a column if it needs a stub (i.e. is of UnicastRemoteObject type) or if objects of it's type are communicated (i.e. implements Serializable): Class Diagram; optional: with table information color-coded>

Lab task #2:

prototype client and server description	
Client	**Server**
The client calls the Registry which have have already the server- stub and this why the client can directly connect with the server. When a method call is made at the Client-side, the sends message through the interface to the server	The Server in the other hand implements the interface with the method call so when the client calls this method, the server gets the message and feeds back the Client with the right value in response to the method call. In this case, the

which then in return feeds back the client with the right value. Here the Client represent our application while the server represents the Robot itself.	server represents the Robot and the Client the application.

Lab task #3:



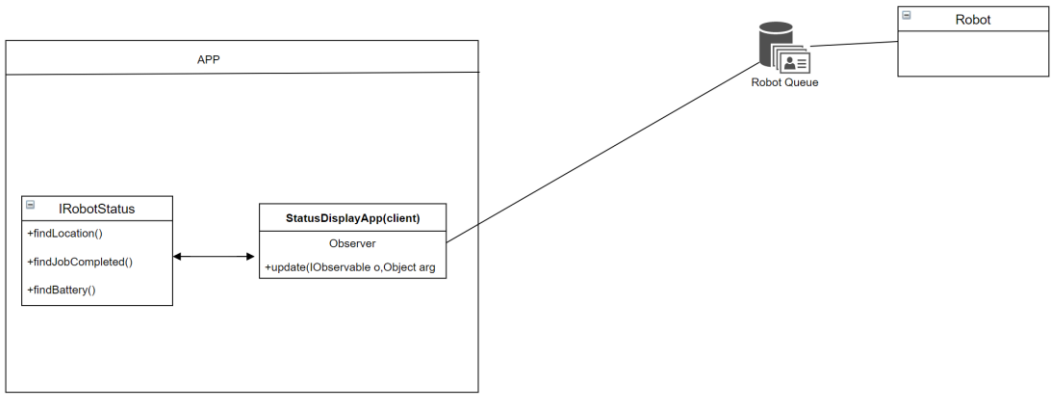
Classes	interfaces	Client- side	Server-side	stub	Description
RobotStatus	IRobotStatus	IRobotStatus	Remote IRobotStatus	getUpdate()	The Server feeds the Registry with a server stub which when the client calls the registry and sends the stub to the client.
Location	ILocation		Remote ILocation	getLocation()	The Server gets the method of the interface and when the client makes request with the same method call. The server then sends the stub to the client
JobCompleted	IJobCompleted		Remote	getJobCompleted	Same process like previously with the

			IJobCompleted		Location class.
Battery	IBattery		Remote IBattery	getBattery	Here is also same process going on because the server communicate with the interfaces and sends the client the stub when the method call is made.

6 MOM-based Design

To realize the MOM-based design, there are a small number of changes that must take place to change the RMI-based design into a MOM-based design. Since there are several design choices that would alter the implementation, a significant design choice must be explicitly stated prior to moving forward: Since multiple apps can pair to a single robot, and multiple robots can be paired to a single app, so I designed the app and the robot acting simultaneously as both server and a client fully asynchronously. In other words, both the app and the robot will send messages to the message queue, but neither will require a response from the other before moving forward.

Robot Queue



In this regard, the Robot has its own message queue, and pairing an app to the robot will mean subscribing to that robot queue.

The reason of this design concept is to facilitate message transmission from a robot to all apps paired with the robot considering that a robot could be paired to more than one app.

Topic: requesting Location from the Robot

Subscriber class:	StatusDisplayApp
Publisher Class:	RobotServer
Payload Description (send):	Observer- the observer will carry as payload an IRobotStatus object containing the getLocation method from the update.
Payload Description (receive):	findLocation- the IRobotStatus that sent the payload to the RobotClient will wait to get a response that the robot received the payload or notification that the message is in a queue.
Topic: requesting for JobCompleted	
Subscriber Class:	StatusDisplayApp
Publisher Class:	RobotServer
Payload Description (send):	Observer- the observer will carry as payload an IRobotStatus object containing the getJobCompleted method from the update.
Payload Description (receive):	findJobCompleted- the IRobotStatus that sent the payload to the RobotClient will wait to get a response that the robot received the payload or notification that the message is in a queue.

7 Shortcomings and drawbacks

I got a better understanding of what I'm required of and now I tried to improve it better and I got a clearer picture of how the how communication and connections works. The other thing is that my Team mate cannot continue and it's a pity because I wished to offer more help but she had other family problems that affected her.

References

<all used ressources>