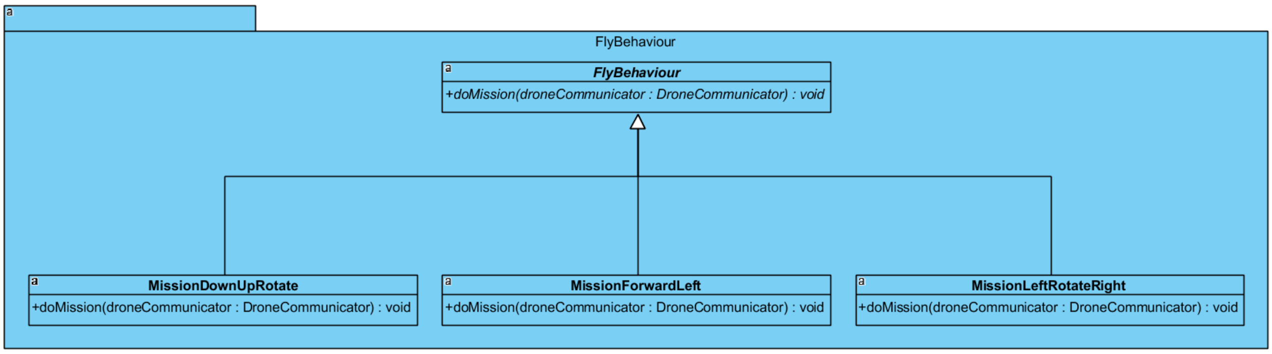
Kanak Tenguria A02310652

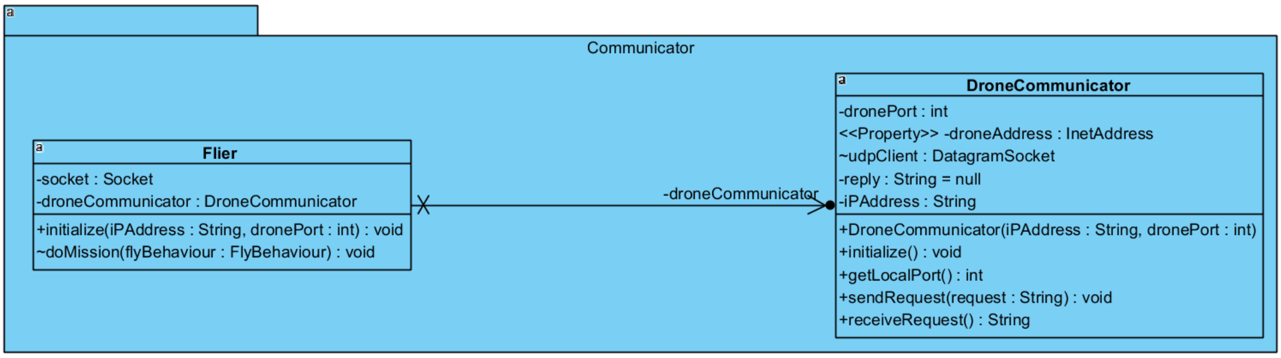
HW1 Report
Submitted on: 11 September 2019

UML Diagrams

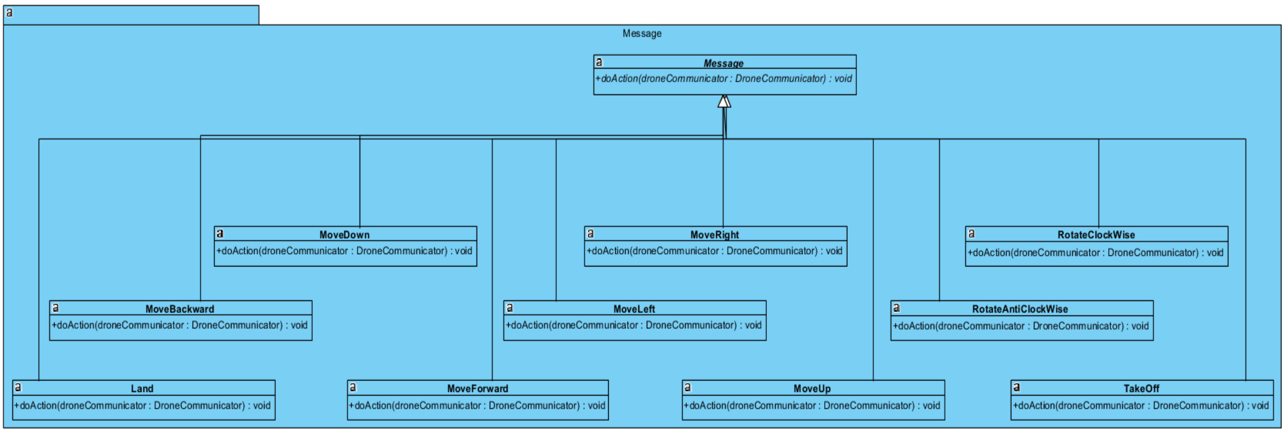
1. Class diagram of package FlyBehaviour:


FlyBehaviour
FlyBehaviour
+doMission(droneCommunicator : DroneCommunicator) : void
MissionDownUpRotate
+doMission(droneCommunicator : DroneCommunicator) : void
MissionForwardLeft
+doMission(droneCommunicator : DroneCommunicator) : void
MissionLeftRotateRight
+doMission(droneCommunicator : DroneCommunicator) : void

2. Class diagram of package Communicator:


Communicator
Flier
-socket : Socket
-droneCommunicator : DroneCommunicator
+initialize(iPAddress : String, dronePort : int) : void
~doMission(flyBehaviour : FlyBehaviour) : void
-droneCommunicator
DroneCommunicator
-dronePort : int
<<Property>> -droneAddress : InetAddress
~udpClient : DatagramSocket
-reply : String = null
-iPAddress : String
+DroneCommunicator(iPAddress : String, dronePort : int)
+initialize() : void
+getLocalPort() : int
+sendRequest(request : String) : void
+receiveRequest() : String

3. Class diagram of package Message:


Message
Message
+doAction(droneCommunicator : DroneCommunicator) : void
MoveDown
+doAction(droneCommunicator : DroneCommunicator) : void
MoveRight
+doAction(droneCommunicator : DroneCommunicator) : void
RotateClockWise
+doAction(droneCommunicator : DroneCommunicator) : void
MoveBackward
+doAction(droneCommunicator : DroneCommunicator) : void
MoveLeft
+doAction(droneCommunicator : DroneCommunicator) : void
RotateAntiClockWise
+doAction(droneCommunicator : DroneCommunicator) : void
Land
+doAction(droneCommunicator : DroneCommunicator) : void
MoveForward
+doAction(droneCommunicator : DroneCommunicator) : void
MoveUp
+doAction(droneCommunicator : DroneCommunicator) : void
TakeOff
+doAction(droneCommunicator : DroneCommunicator) : void

Insights uncovered during the project

So, when I started this project, I had no idea about practical implementation of interface or abstract class. I only had theoretical knowledge about it. Coming from a background where theoretical knowledge is given preference, this kind of education approach made me learn new things. Thinking about a code where we need to apply all the concepts proved very hard for me. Initially, I just copied the sample code of socket to a communicator class. For controlling the flow, I made Control class and did all the coding there. In the beginning, it was an example of very poor abstraction. I was not able to figure out how to even separate send and receive methods. This assignment also taught me how to use constructors as well.

More importantly, I learned how to apply abstraction in the project. The difference between public and private and also the significance of declaring the data members in different places was something new. I never understood the importance of abstraction before this assignment since I had no practical exposure so that is something very important I learned from this project. I realized that this type of coding approach is very useful in achieving modularity as well. While in the development phase, I even witnessed the use of this approach while changing some of the commands given to the drone.

Talking about the strategy pattern, I first created one interface named take off rather than creating a separate interface for all the commands and implementing it, which shows that there was no hierarchy in my mind for the project. This project made me understand the value of UML diagrams. After many failed attempts of directly coding everything, I sketched a proper class diagram and followed it to achieve the strategy pattern. In fact, it gave me such an understanding that I applied double strategy to the project.

One more thing worth sharing is the UML diagram itself. I didn’t know anything about how to develop class diagrams and so this project gave me motivation to learn that as well. Also, just like everything else, I didn’t know how to do unit testing as well. Although that’s the only thing which is incomplete in this submission but considering the fact that there was too much to learn during this assignment, I am very satisfied that I made such efforts and learned so many things. I will definitely complete the unit testing in upcoming submissions.

Overall, after starting this project and applying strategy pattern in this assignment, I realized the value of modularity and abstraction and learned many new things in the process as well.