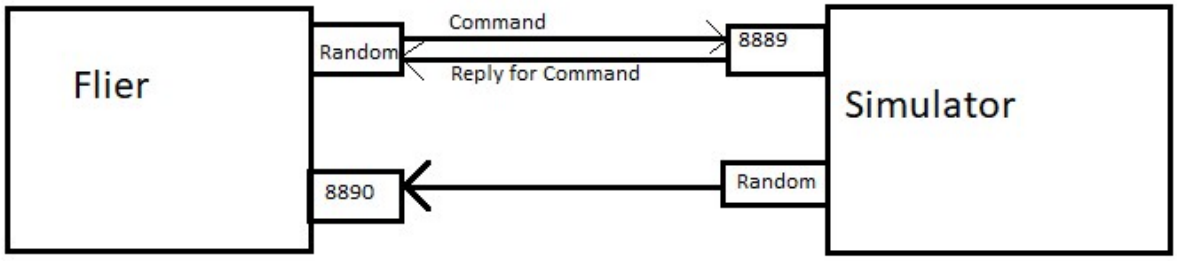
Encapsulation, Observer and Template Method Assignment Report -Princy Jain

Introduction:

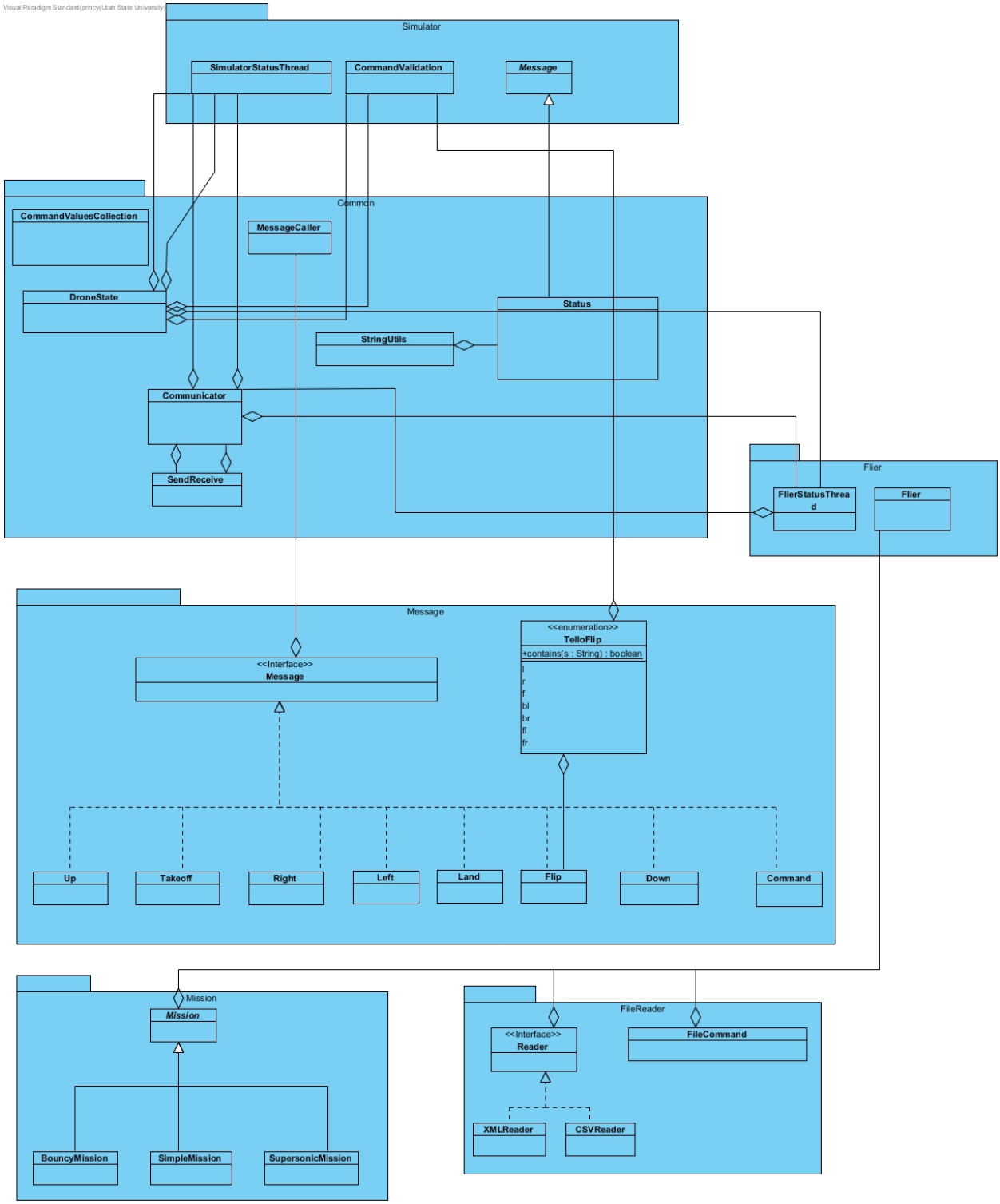
This assignment mainly focuses on learning **Observer Pattern** along with **Encapsulation** and **Template method pattern.** According to Observer Pattern, the observer will not ask for the current state of observable (one who is observed), instead the observable will update the observer every time there is a change in state. Here, in this project the observer is **Flier** and the observable is **Drone/Simulator.**

Project Design:

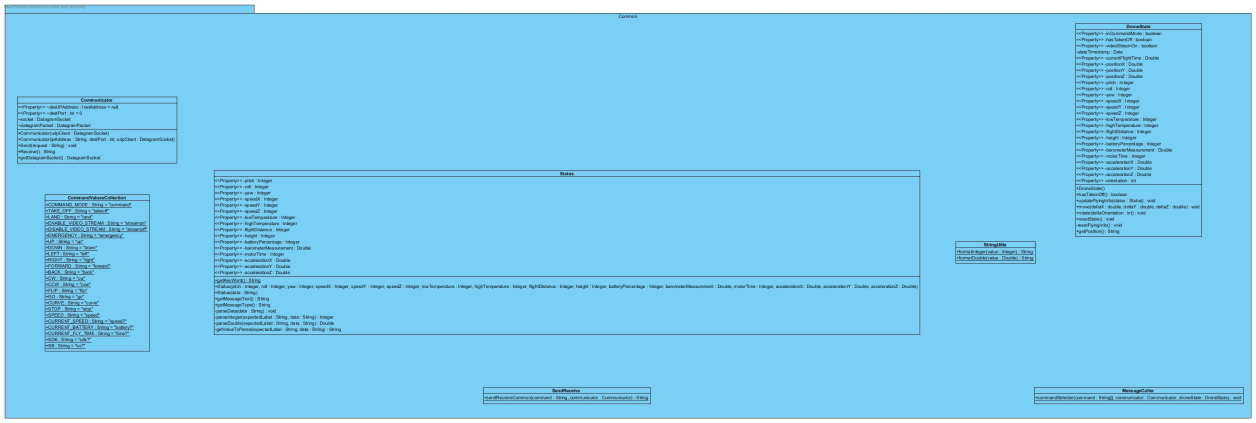
I started my design by making the communication diagram to have a clear understanding of how the communication is taking place between the flier and simulator.



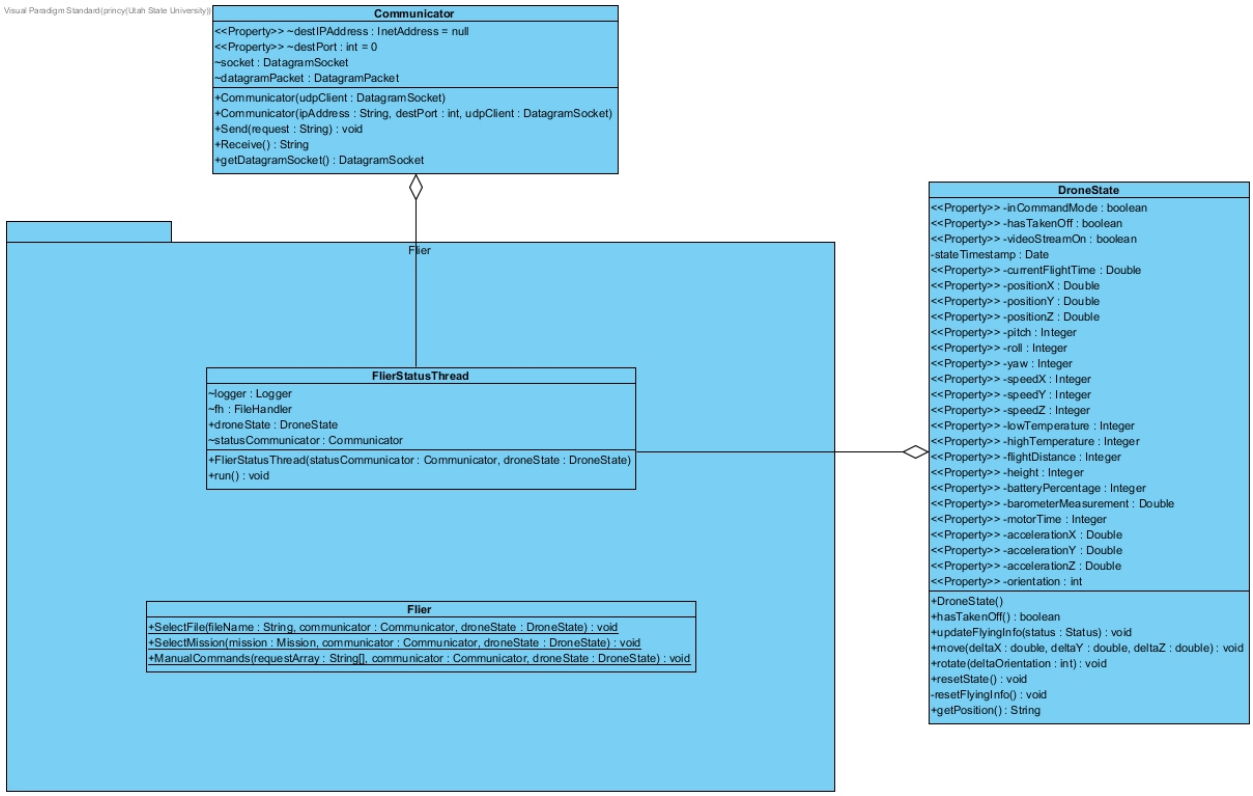
Class Diagrams:



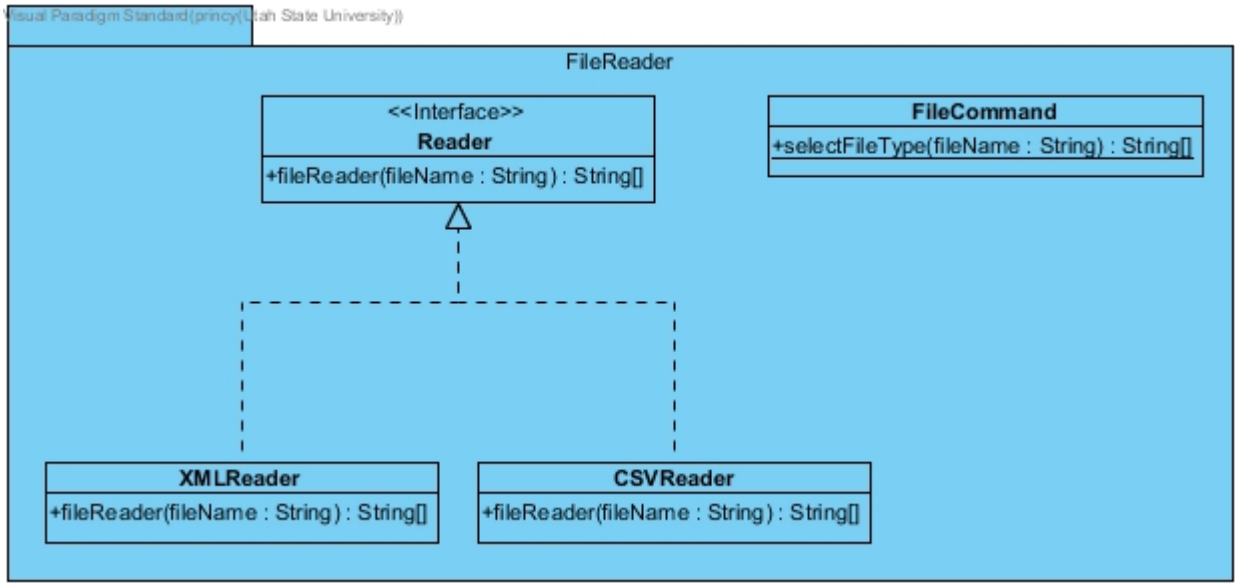
Common Package:



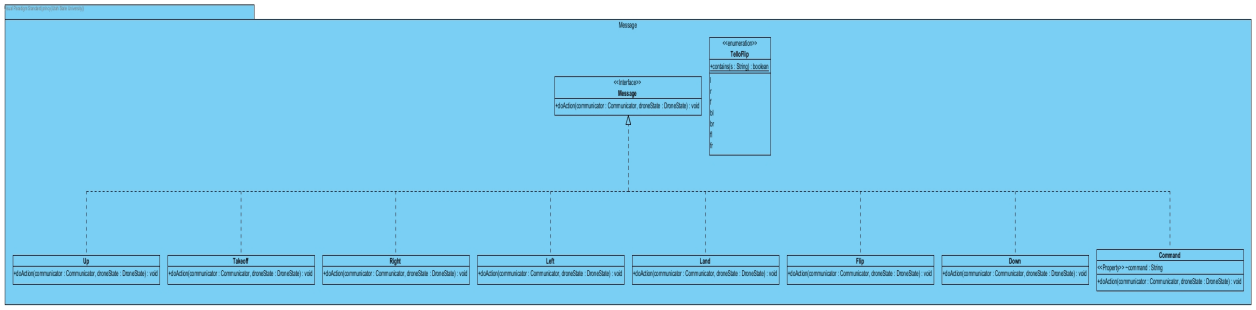
Flier Package:



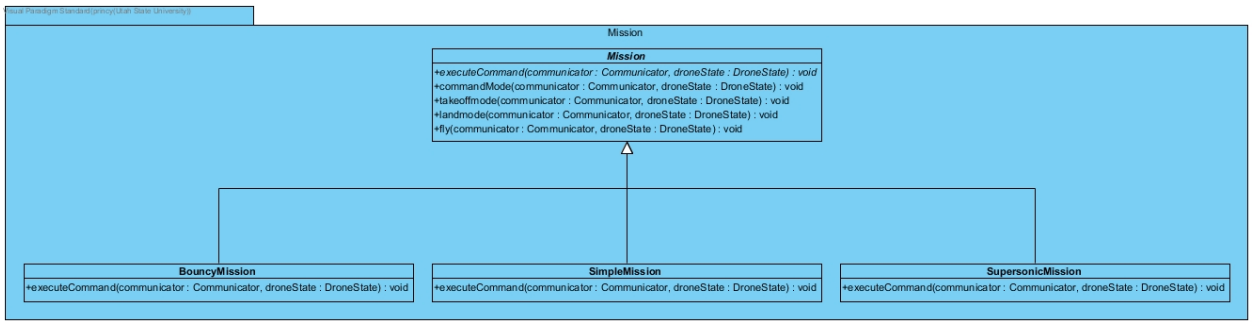
File Reader Package:



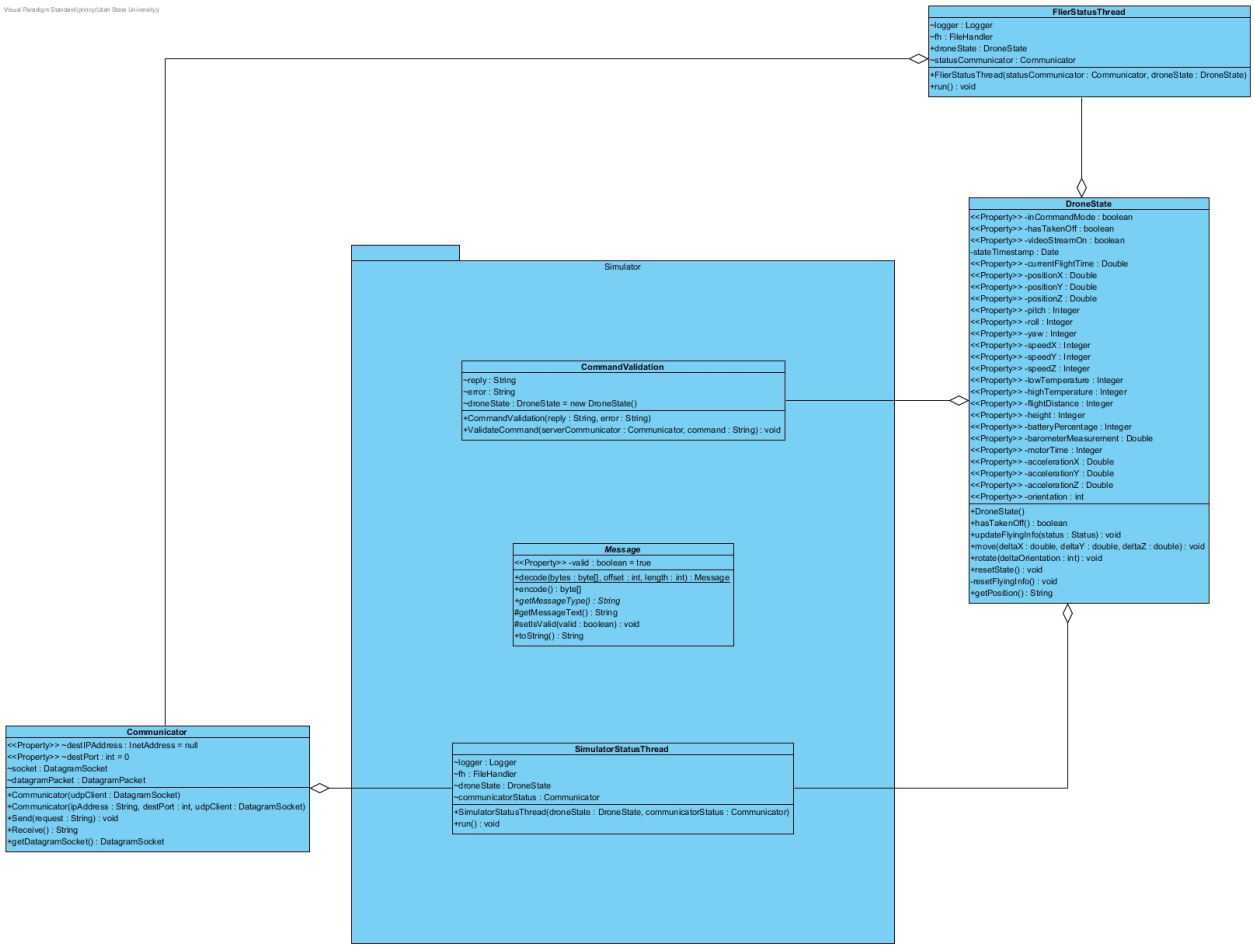
Message Package:



Mission Package:

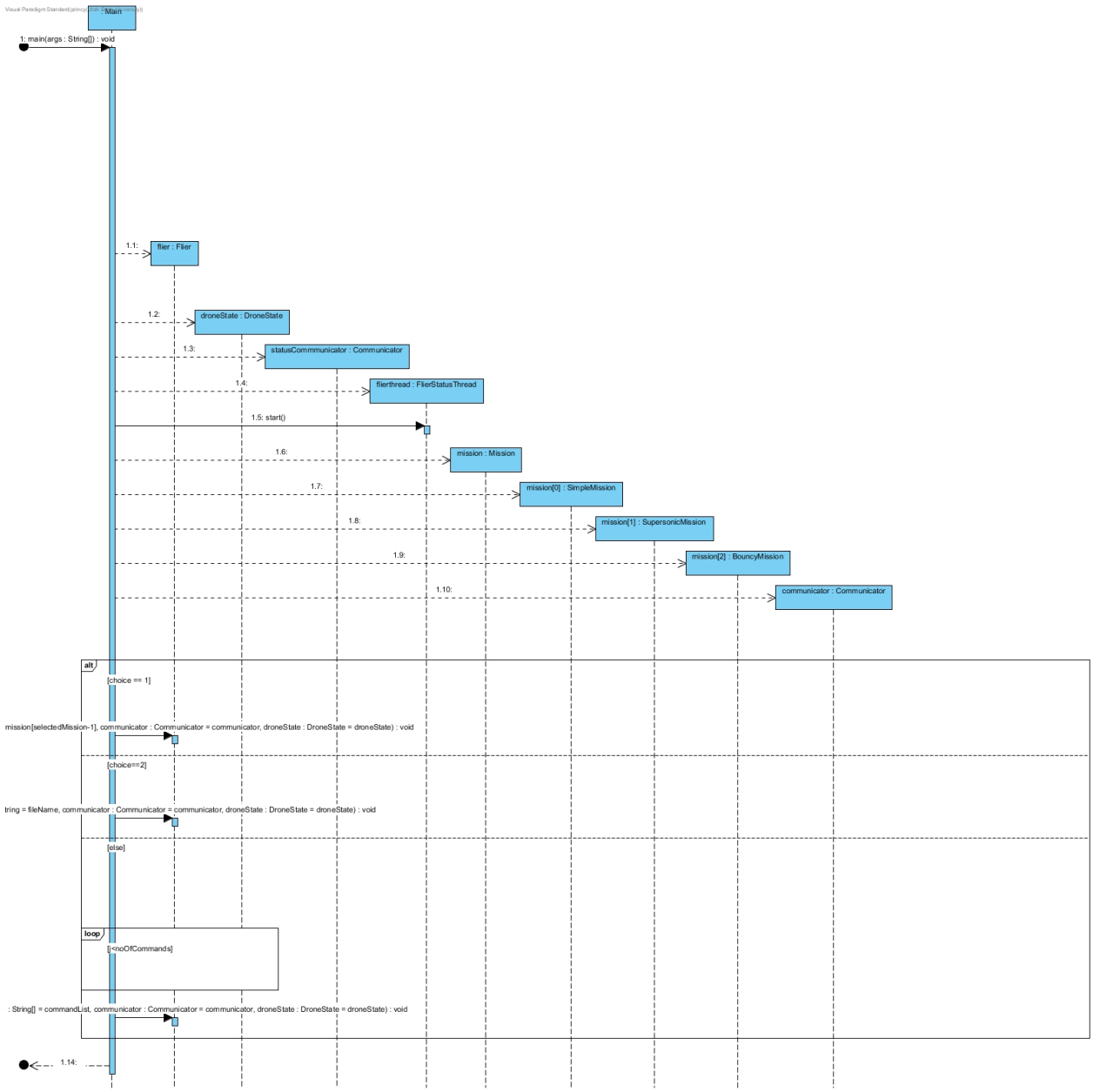


Simulator Package:

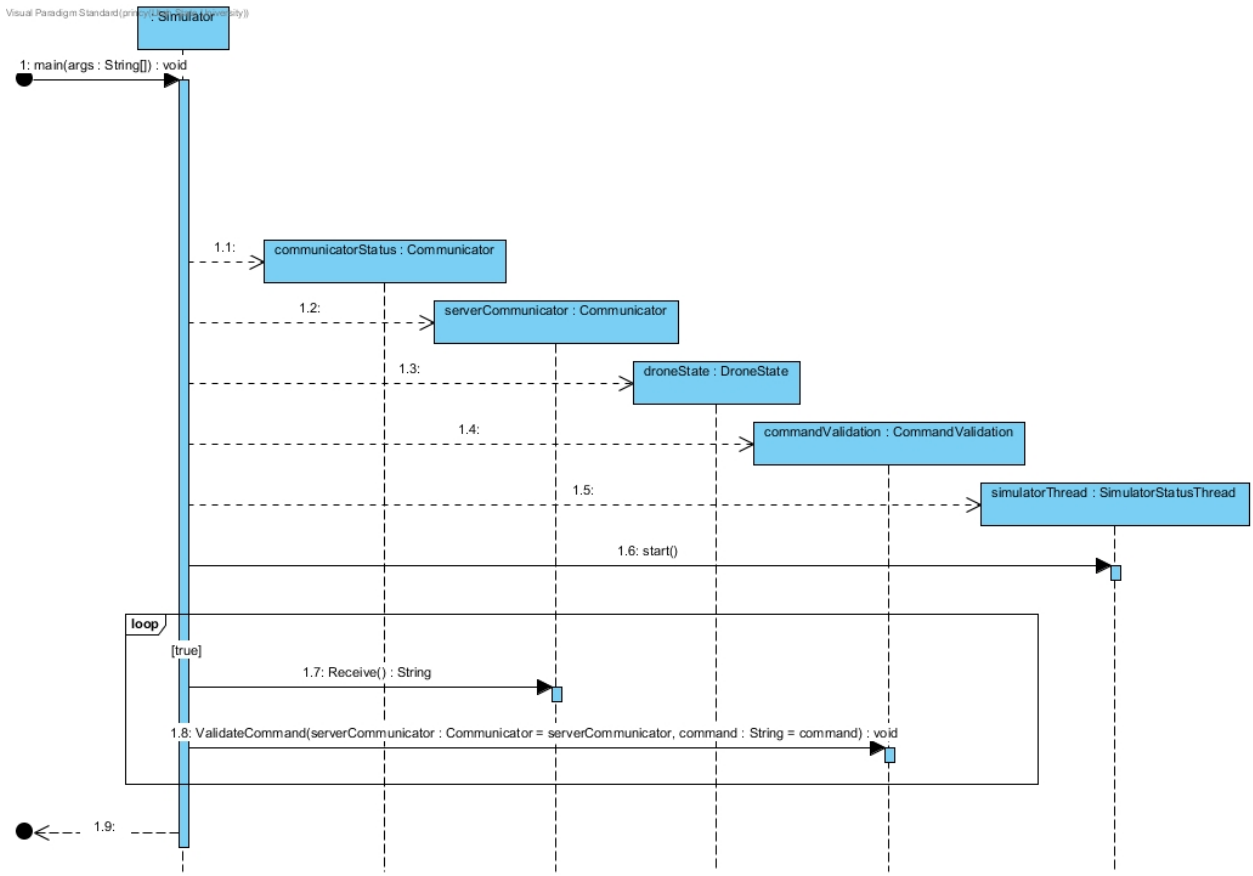


Interaction Diagram (Sequence Diagram):

Flier Main:



Simulator Main:



Points Covered:

1. **Mandatory**- Both the flier and the simulator checks if the drone is in command mode or not. If the drone/simulator is in command mode, then only the simulator will send status messages to the flier and also the flier will only receive the status messages when the drone is in command mode. The port used at the flier end to receive status message is 8890. This is executed at Flier side in **FlierStatusThread** class and on the simulator side in **SimulatorStatusThread** class.
   The Flier irrespective of the ongoing Mission will use this status message to get a clear picture of the drone's state.
2. **Mandatory**- If the Battery drops below 20%, then the drone will land safely instead of executing the further commands. This condition can be found in Common\SendReceive class.
   The flip maneuver will be converted into left maneuver if the battery drops below 20%. This condition is implemented in Message\Flip class.
   If the high temperature gets above 60 then the drone will land safely instead of executing the further commands. This condition can be found in Common\SendReceive class.
3. **Mandatory**- To do
4. Not covered
5. My project is capable of loading missions from two different file formats, i.e., CSV and XML.
   This is done in package FileReader. I have created a Reader interface which is the implemented by both CSV and XML Reader (Applied Strategy Pattern).
6. The user is asked to input the port number of the drone and also the number of retries they want perform while sending and receiving messages.
7. Every time a command is executed, the position of the drone is updated and is displayed to the user (7.1).
   Every time a command which requires parameters like distance to move, degree to rotate or type of flip is executed, the user will be asked to enter that specific parameter (7.2).

If the battery drops below 20%, the flip is converted into left (7.5).

8. Every time user executes the code, he/she will be asked to select if they want to select mission from existing missions, from csv or xml file, or they want to create their own mission which covers open close principle.
9. Common\Communicator class is used by both Flier and Simulator to send and receive messages.
10. Message serialization and deserialization is done in Common\Communicator which is shared by both Flier and Simulator.
11. Common\DroneState and Common\Status classes which are responsible for drone state is used by both Flier and Simulator to update the state of drone at their ends.

Insights Uncovered:

- Socket Programming which was quite a new concept for me.
- Template Pattern
- Observer Pattern