# HW2- Encapsulation, Observer and Template Method

## Introduction

In this project, we are improving the drone flyer by implementing different design patterns. Also, we are developing a simulator for substituting the drone.

## Aim

Our aim to is to become familiar with Encapsulation and applications of more patterns like Observer pattern, Template method or Factory method.

## UML Diagrams

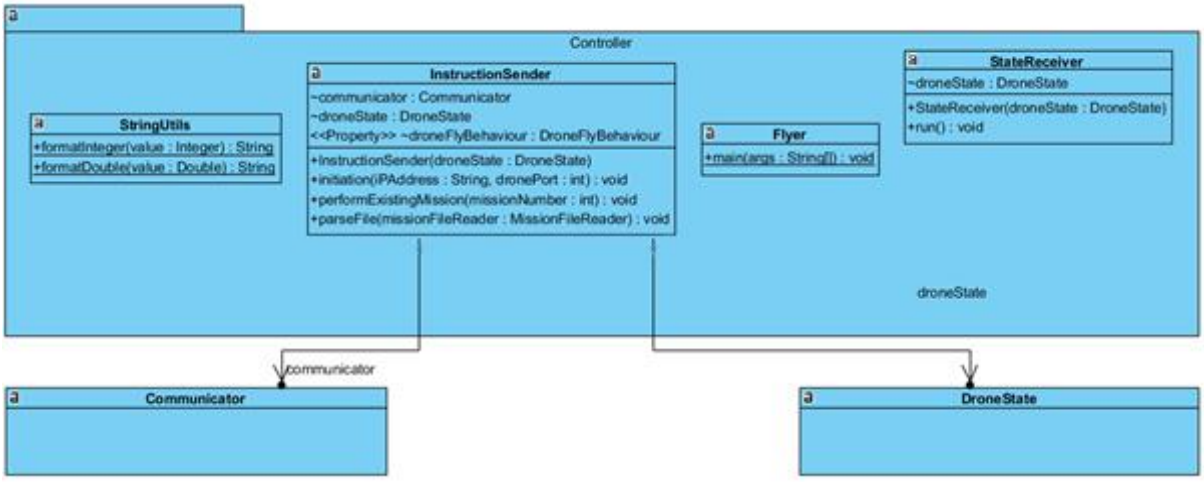


Fig. 1.1 Class Diagram of package Controller

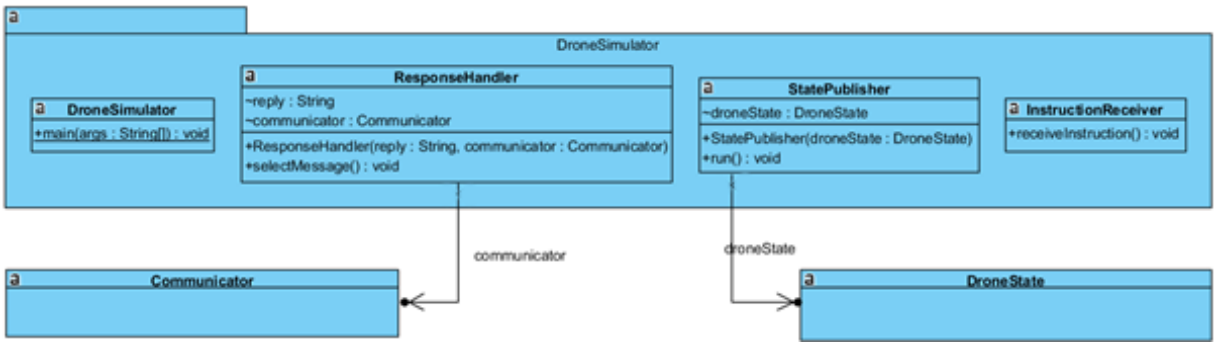


Fig. 1.2 Class Diagram of package Drone Simulator

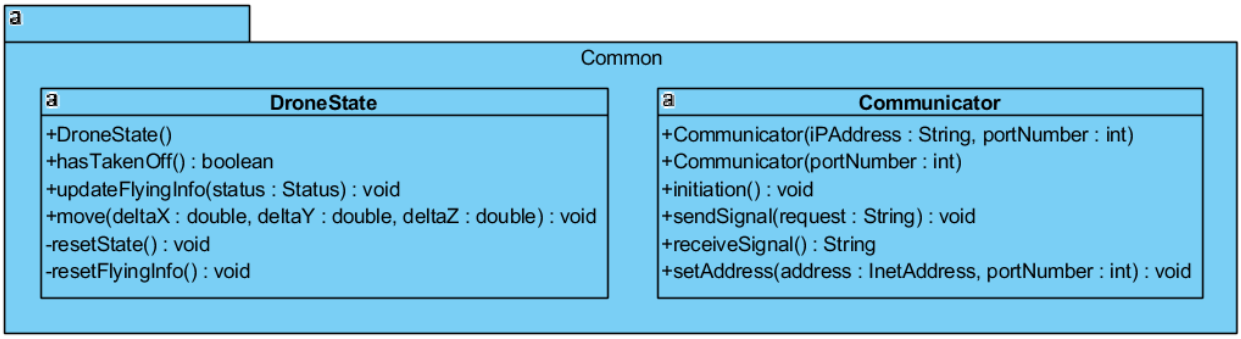


Fig 1.3 Class Diagram of package Common

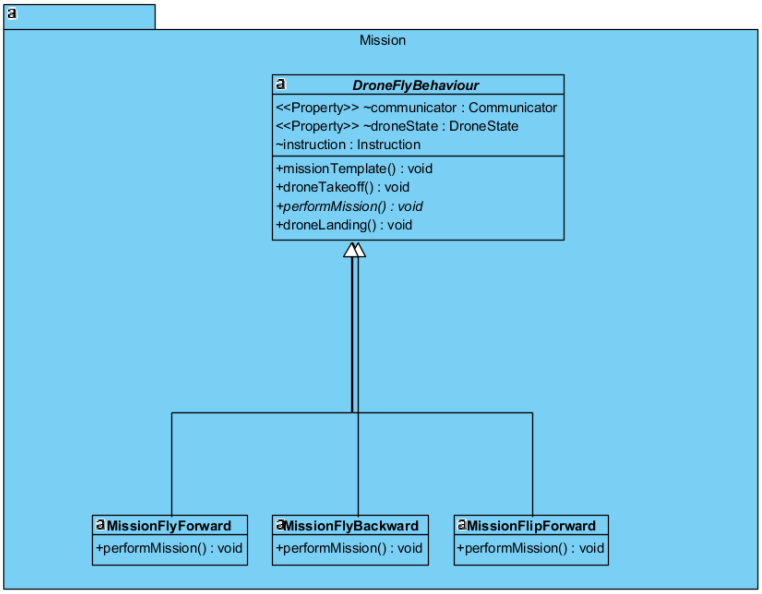


Fig 1.4 Class Diagram of Package Mission

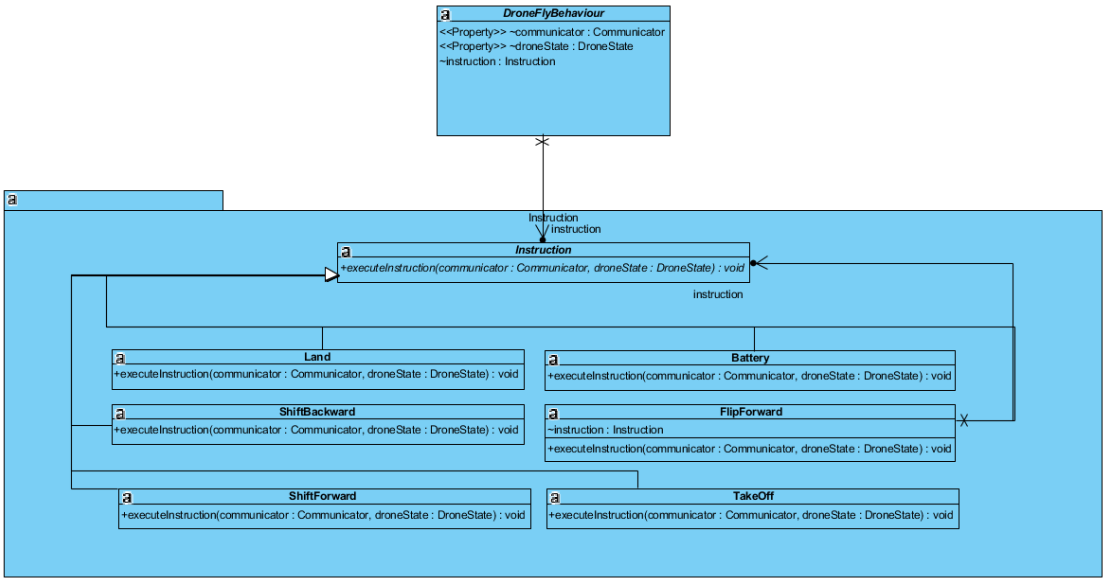


Fig 1.5 Class Diagram of Package Instruction

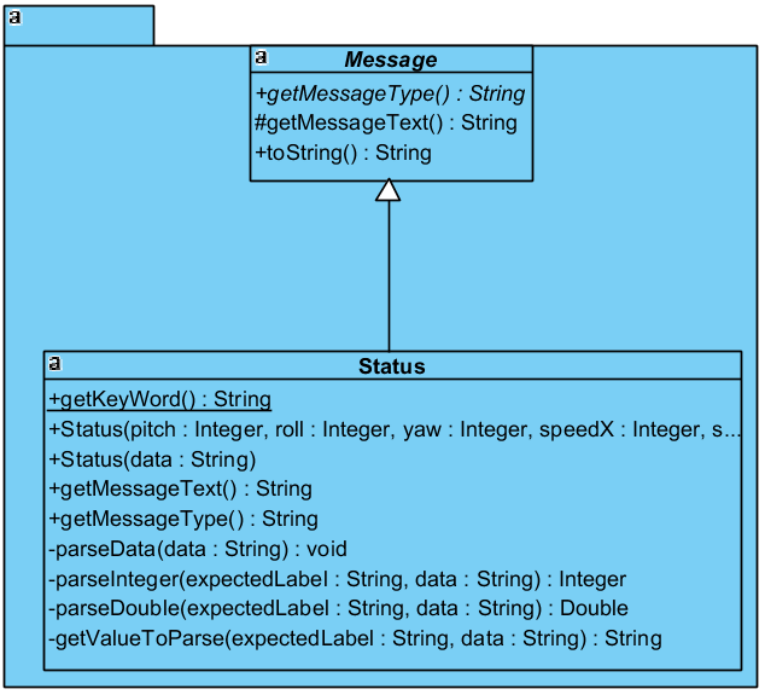


Fig 1.6 Class Diagram of Package Message

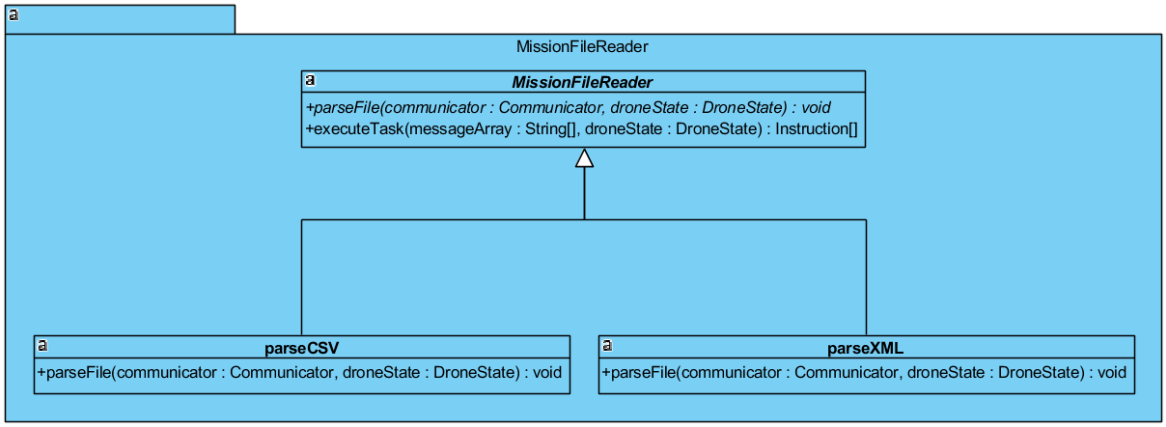


Fig 1.7 Class Diagram of Package MissionFileReader

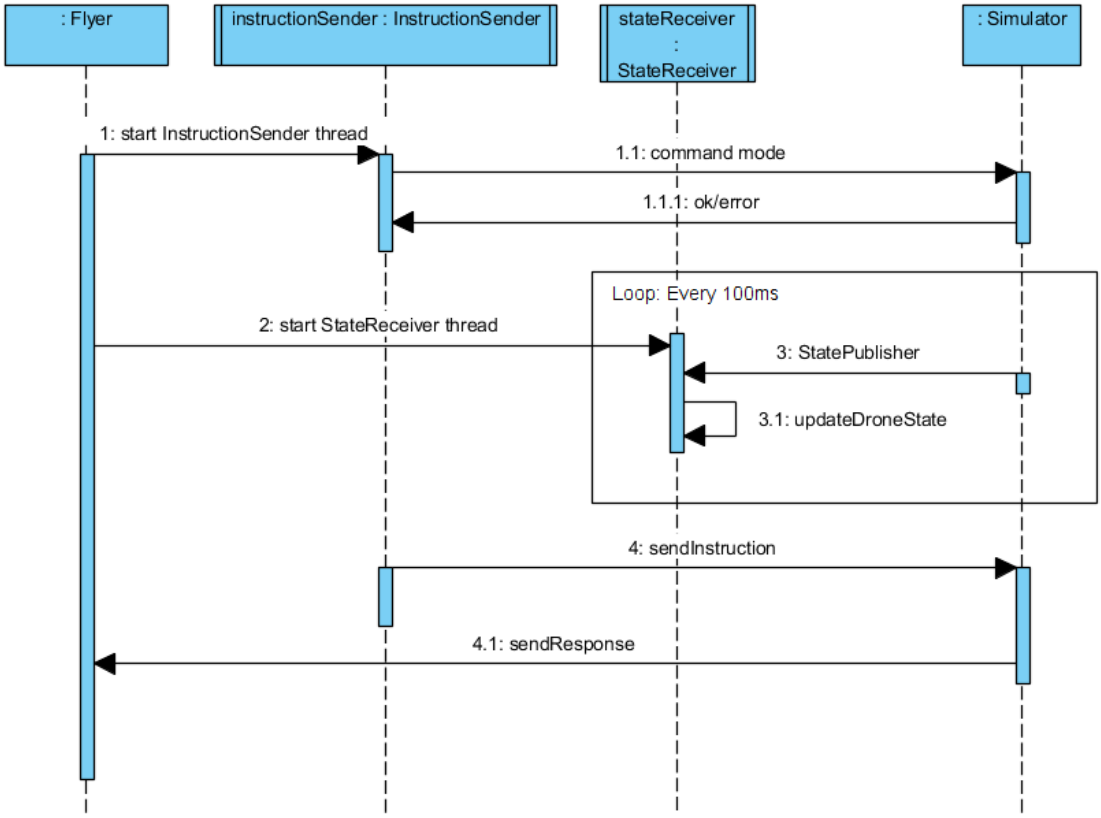


Fig 1.8 Sequence Diagram of Flyer

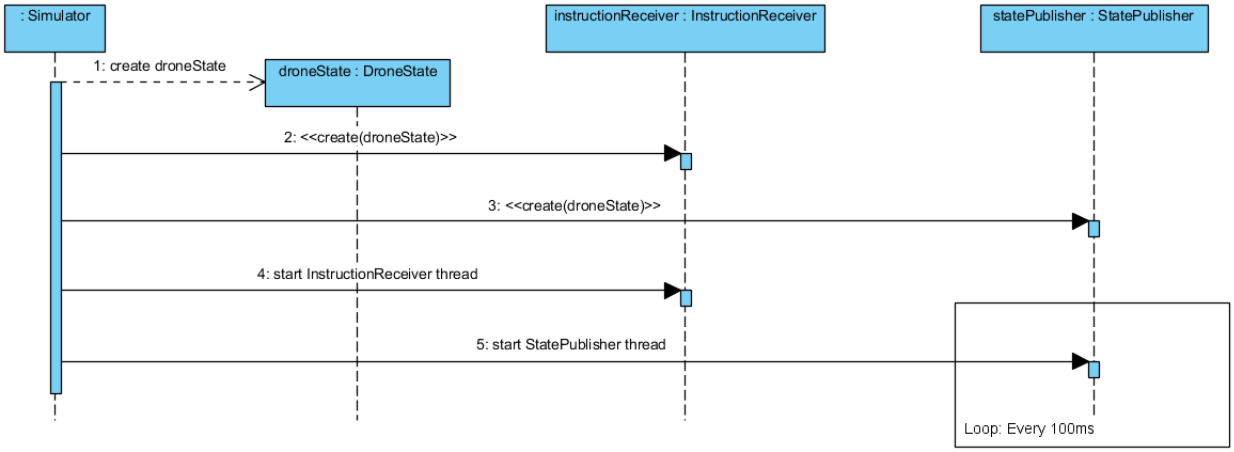


Fig 1.9 Sequence Diagram of Simulator

**Insights Uncovered**

-While doing the project I have learned a very important part of programming that is Socket Programming. I created a Drone Simulator and connected it to the Flyer using Socket Programming.

-While trying to figure out all the patterns that the project demands, I have appended a lot into my knowledge domain of java. Also I have faced the fact that every time we should try to figure out new methods and ways rather than coding just with the pre-existing knowledge: ”Never stop learning”.

-I got introduced to Template Pattern and it did clear a lot of concepts and it is also a very efficient way to avoid code duplication.

-When I was trying to implement the Strategy pattern I learned the value of project structure which until now I have not paid a lot of attention to. Implementing the pattern actually gave me a generalized understanding of what kind of situation needs to follow strategy pattern. Designing a flow for the project before coding gave me a kick start and also was the key to implement good abstraction ad modularity.

-I have improved the way of passing parameters and also the way of passing objects is a lot efficient than how I used to do it before. These small things are the key to implement good abstraction and modularity.

-one of my bad practice that I tackled while doing this project is that we should always be open to new ideas and strategy rather than thinking in one direction. Because sometimes the only solution is to change the whole plan and get started again from a new point. I have wasted a lot of time trying to figure out things where all I needed was to think differently about a new solution.

-I have also figured out that the naming conventions are not to be taken lightly as it could make the code look messy and also it won’t be readable for someone else. Also it is time wasting to debug a code which does not have proper names.

-Last point I learned is that we should always code in such a way that further add-ons and changes should not be a cumbersome task.

-Socket programming opened a whole new range of imagination of things that we can do with coding to which until now my mind was oblivious to.