Flight Gear Client

```
                    ┌─────────────────────────┐
                    │      <<Interface>>      │
                    │      IFGClientApi       │
                    ├─────────────────────────┤
                    │ + setAileron()          │
                    │ + setElevators()        │
                    │ + setRudder()           │
                    │ + setThrottle()         │
                    │ + setBreaks()           │
                    └─────────────────────────┘
                               △
                               ┊
                               ┊
```

FGClient
- server : ServerSocket - stop : volatile boolean - FlightGear : Client - map : HashMap - es : ExecutorsService
+ runServer() + setAileron() + setElevators() + setRudder() + setThrottle() + setBreaks() + finalize()

⊕ ─────────────▶

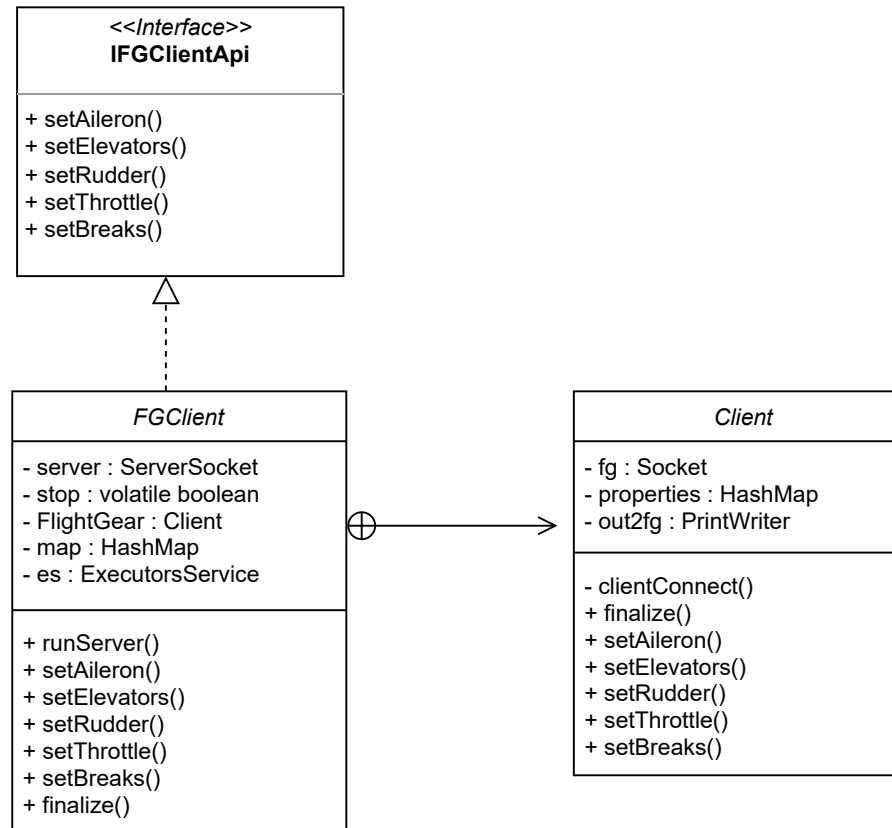| Client |
| --- |
| - fg : Socket
- properties : HashMap
- out2fg : PrintWriter |
| - clientConnect()
+ finalize()
+ setAileron()
+ setElevators()
+ setRudder()
+ setThrottle()
+ setBreaks() |

Description

this process is going to connect to flight gear simulator as client and execute commands that will actively fly the plain and also to be a server for the Agent's model and execute commands by his reauest.
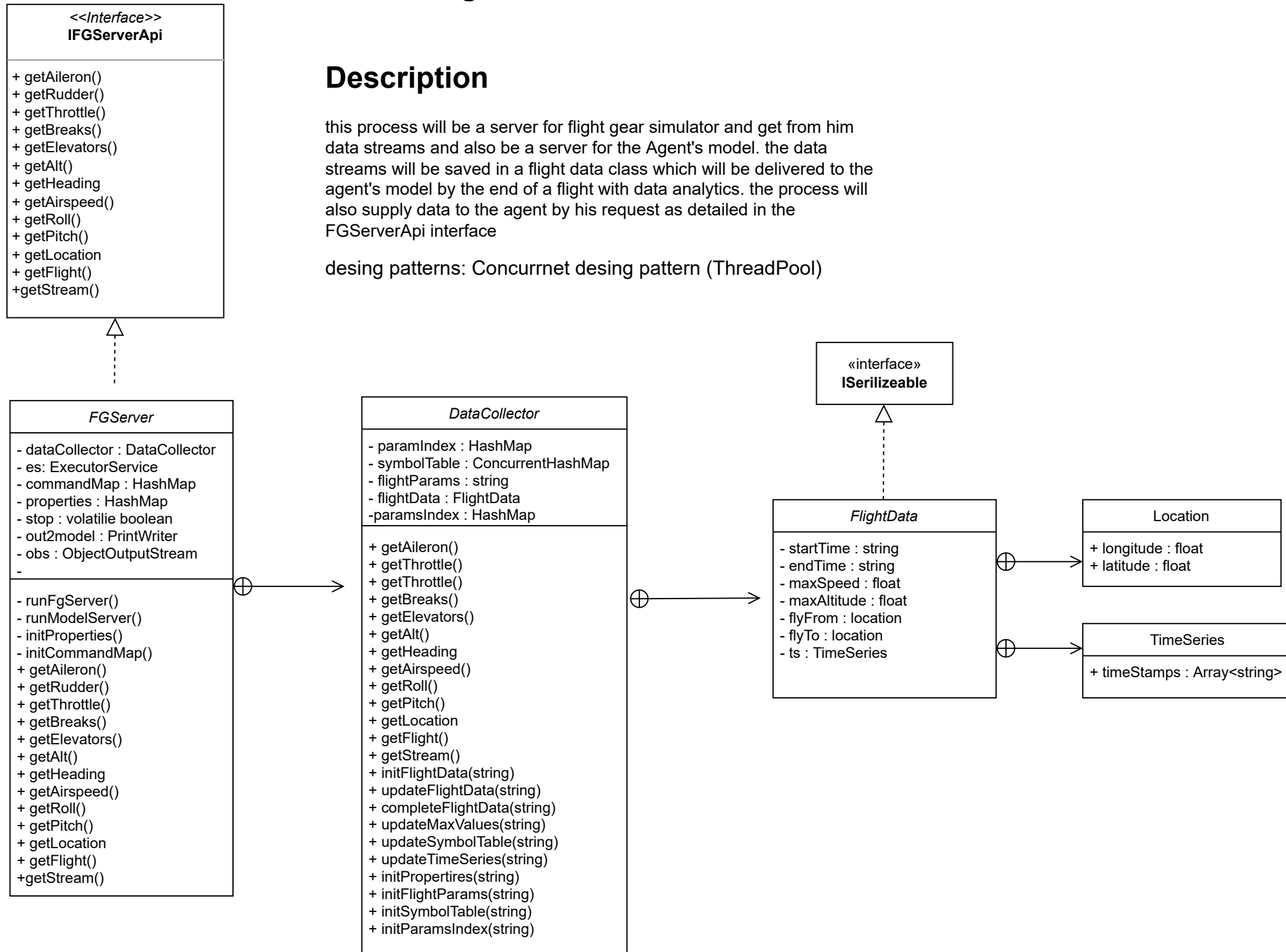
desing patterns: Concurrnet desing pattern (active object)

Flight Gear Server

<<Interface>>
IFGServerApi

+ getAileron()
+ getRudder()
+ getThrottle()
+ getBreaks()
+ getElevators()
+ getAlt()
+ getHeading
+ getAirspeed()
+ getRoll()
+ getPitch()
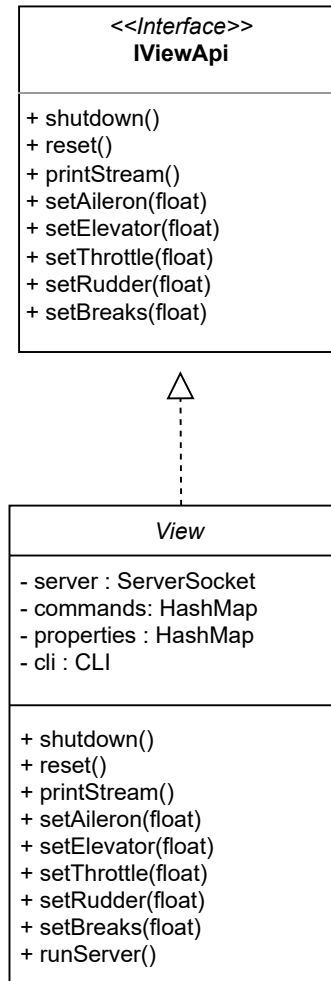+ getLocation
+ getFlight()
+getStream()

Description

this process will be a server for flight gear simulator and get from him data streams and also be a server for the Agent's model. the data streams will be saved in a flight data class which will be delivered to the agent's model by the end of a flight with data analytics. the process will also supply data to the agent by his request as detailed in the FGServerApi interface

desing patterns: Concurrnet desing pattern (ThreadPool)

FGServer

- dataCollector : DataCollector
- es: ExecutorService
- commandMap : HashMap
- properties : HashMap
- stop : volatilie boolean
- out2model : PrintWriter
- obs : ObjectOutputStream
-

- runFgServer()
- runModelServer()
- initProperties()
- initCommandMap()
+ getAileron()
+ getRudder()
+ getThrottle()
+ getBreaks()
+ getElevators()
+ getAlt()
+ getHeading
+ getAirspeed()
+ getRoll()
+ getPitch()
+ getLocation
+ getFlight()
+getStream()

DataCollector

- paramIndex : HashMap
- symbolTable : ConcurrentHashMap
- flightParams : string
- flightData : FlightData
-paramsIndex : HashMap

+ getAileron()
+ getThrottle()
+ getThrottle()
+ getBreaks()
+ getElevators()
+ getAlt()
+ getHeading
+ getAirspeed()
+ getRoll()
+ getPitch()
+ getLocation
+ getFlight()
+ getStream()
+ initFlightData(string)
+ updateFlightData(string)
+ completeFlightData(string)
+ updateMaxValues(string)
+ updateSymbolTable(string)
+ updateTimeSeries(string)
+ initPropertires(string)
+ initFlightParams(string)
+ initSymbolTable(string)
+ initParamsIndex(string)

«interface»
ISerilizeable

FlightData

- startTime : string
- endTime : string
- maxSpeed : float
- maxAltitude : float
- flyFrom : location
- flyTo : location
- ts : TimeSeries

Location

+ longitude : float
+ latitude : float

TimeSeries

+ timeStamps : Array<string>

View

```
        <<Interface>>
          IViewApi

+ shutdown()
+ reset()
+ printStream()
+ setAileron(float)
+ setElevator(float)
+ setThrottle(float)
+ setRudder(float)
+ setBreaks(float)
```

```
            View

- server : ServerSocket
- commands: HashMap
- properties : HashMap
- cli : CLI

+ shutdown()
+ reset()
+ printStream()
+ setAileron(float)
+ setElevator(float)
+ setThrottle(float)
+ setRudder(float)
+ setBreaks(float)
+ runServer()
```
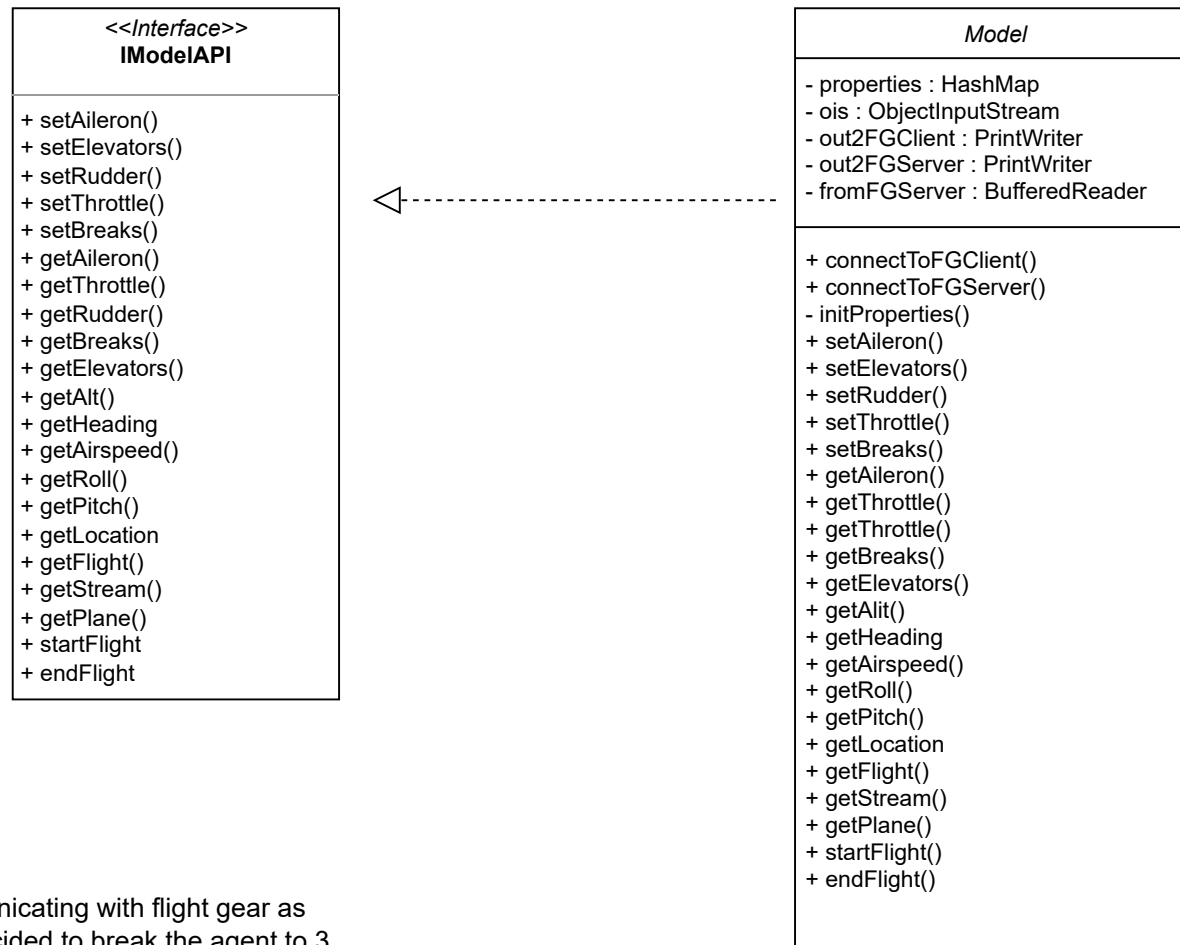
Description

the agent view will listen for external java program for direct commands
(instead of getting the from the backend). mainly for debug.

design patterns: observer design pattern.

Model

<<Interface>>
IModelAPI
+ setAileron()
+ setElevators()
+ setRudder()
+ setThrottle()
+ setBreaks()
+ getAileron()
+ getThrottle()
+ getRudder()
+ getBreaks()
+ getElevators()
+ getAlt()
+ getHeading
+ getAirspeed()
+ getRoll()
+ getPitch()
+ getLocation
+ getFlight()
+ getStream()
+ getPlane()
+ startFlight
+ endFlight

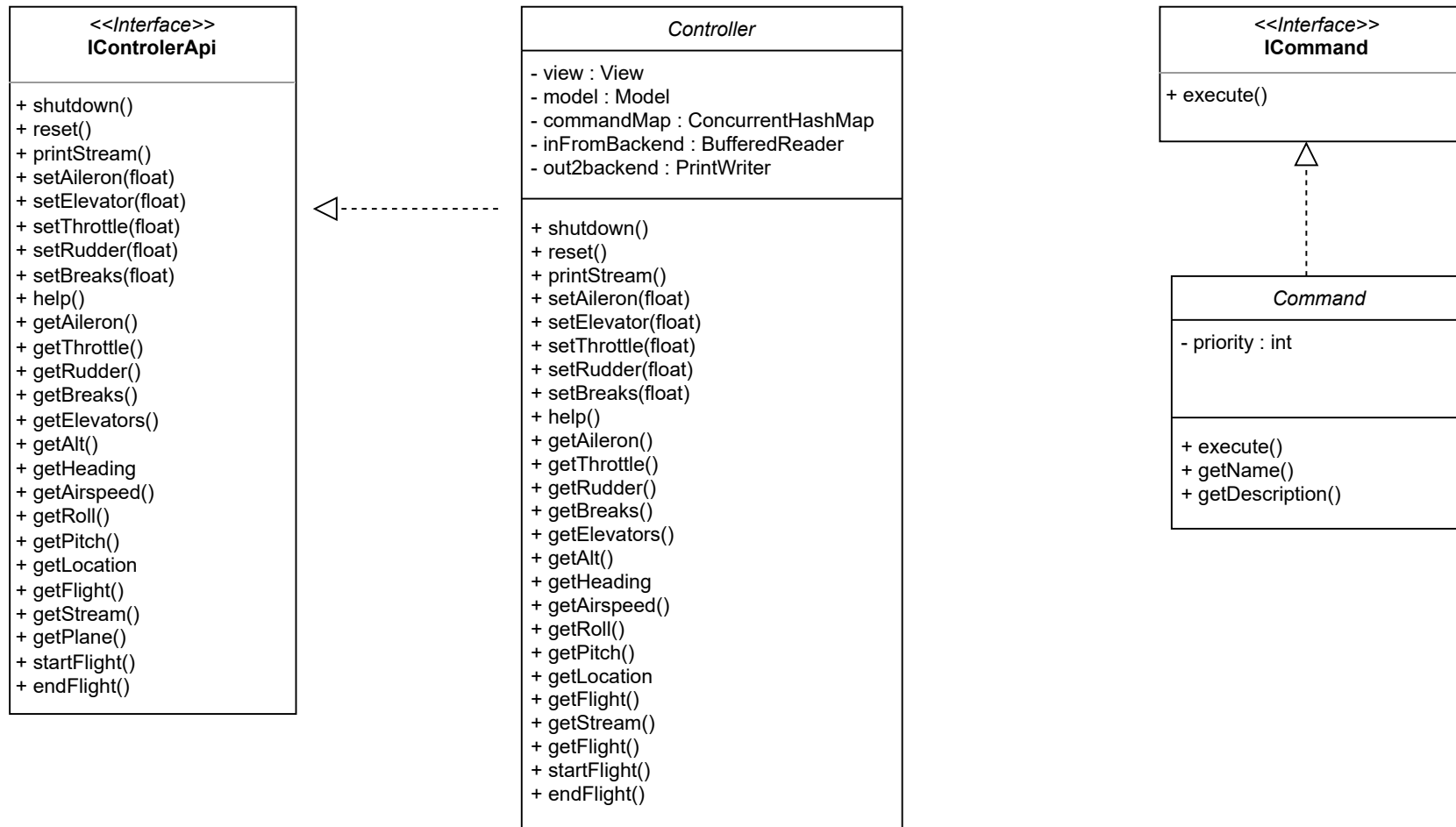| *Model* |
|---|
| - properties : HashMap |
| - ois : ObjectInputStream |
| - out2FGClient : PrintWriter |
| - out2FGServer : PrintWriter |
| - fromFGServer : BufferedReader |
| |
| + connectToFGClient() |
| + connectToFGServer() |
| - initProperties() |
| + setAileron() |
| + setElevators() |
| + setRudder() |
| + setThrottle() |
| + setBreaks() |
| + getAileron() |
| + getThrottle() |
| + getThrottle() |
| + getBreaks() |
| + getElevators() |
| + getAlit() |
| + getHeading |
| + getAirspeed() |
| + getRoll() |
| + getPitch() |
| + getLocation |
| + getFlight() |
| + getStream() |
| + getPlane() |
| + startFlight() |
| + endFlight() |

Description

the agent model is responsible for communicating with flight gear as server and the client, in our project we decided to break the agent to 3 differnet processes (along with FGClient and FGServer) so now the model's job will be to manage the comunication with the controller alongside the 2 other processes, instead of be responsible for executing.

design patterns: observer design pattern.

Controller

<<Interface>> **IControlerApi**
+ shutdown()
+ reset()
+ printStream()
+ setAileron(float)
+ setElevator(float)
+ setThrottle(float)
+ setRudder(float)
+ setBreaks(float)
+ help()
+ getAileron()
+ getThrottle()
+ getRudder()
+ getBreaks()
+ getElevators()
+ getAlt()
+ getHeading
+ getAirspeed()
+ getRoll()
+ getPitch()
+ getLocation
+ getFlight()
+ getStream()
+ getPlane()
+ startFlight()
+ endFlight()

Controller
- view : View
- model : Model
- commandMap : ConcurrentHashMap
- inFromBackend : BufferedReader
- out2backend : PrintWriter
+ shutdown()
+ reset()
+ printStream()
+ setAileron(float)
+ setElevator(float)
+ setThrottle(float)
+ setRudder(float)
+ setBreaks(float)
+ help()
+ getAileron()
+ getThrottle()
+ getRudder()
+ getBreaks()
+ getElevators()
+ getAlt()
+ getHeading
+ getAirspeed()
+ getRoll()
+ getPitch()
+ getLocation
+ getFlight()
+ getStream()
+ getFlight()
+ startFlight()
+ endFlight()

<<Interface>> **ICommand**
+ execute()

Command
- priority : int
+ execute()
+ getName()
+ getDescription()

Description

the agent controller will get requests from the view and from the
backend, make validation checks and than transform them to the model
in the right order (and than back to the view/back end).

desing patterns: Concurrnet desing pattern (thread pool),
Command design pattern and observer design pattern