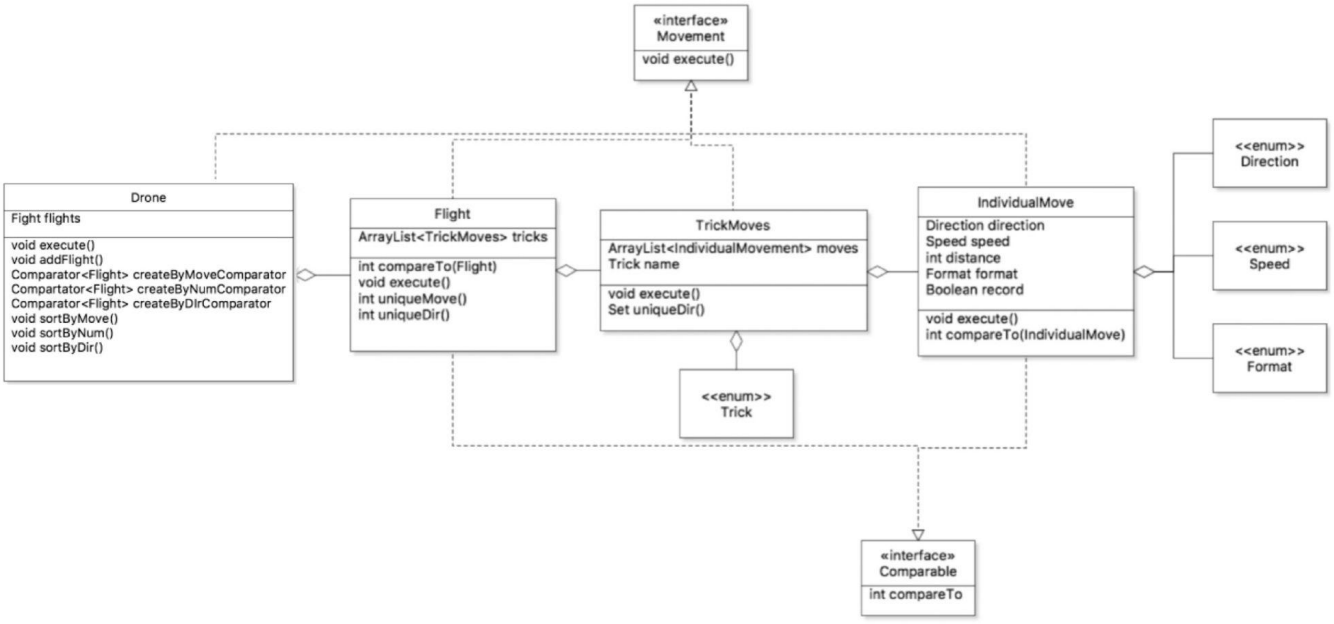

General design

- The `Drone` class consists of a list of `Flight` objects it can execute; The `Flight` class consists of a list of `TrickMoves` objects with name in Enum `Trick`; Each trick consists of a list of `IndividualMoves` with fields: *direction*, *speed*, *distance*, and a boolean *record* representing whether it is recorded with its *format*. All classes above implements the interface `Movements` as they are all made of individual moves. The `execute()` command will be passed down level by level so that every move will be executed.
- In order to compare the flights by different strategies, static factory methods (`createByMoveComparator()`, etc) are used:
    - To allow the client to switch between different comparison strategies without having to tweak the code of `compareTo()` method. It's more convenient as the client only needs to change the name of the comparator to achieve flexible comparison.
    - To give the comparators access to private members of the class they compare without breaking encapsulation.

Client

- The client is able to specify the *speed* and *distance* of each trick they add to the flight; However, for the realistic aspect of the functionalities, certain moves in the tricks are pre-programmed. (For example, the TAKEOFF trick always starts with LOW speed, but the client has the freedom to program the speed afterwards.)