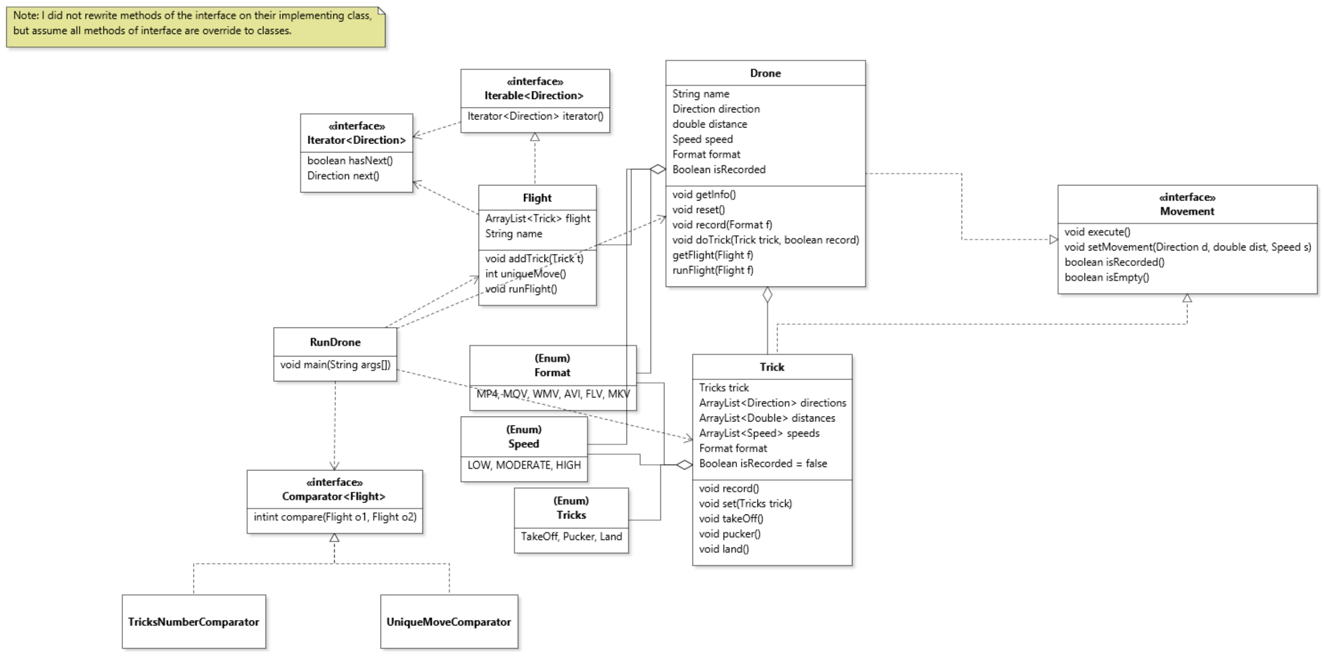
The solution starts off with the Drone class, which represents a single drone entity that contains methods allowing clients to perform various actions, such as set a movement, execute it, record, and perform tricks. The solution also includes an interface Movement, which was implemented in Drone, as well as in Trick as both classes could be defined through Movement.

Apart from implementing an interface, other new techniques were also used, such as implementing an iterator in order to check our private fields without making a deep copy like last assignment (while keeping encapsulation), and creating comparator classes that will allow us to compare and sort our flight under different conditions.

Multiple assumptions that lead to various design decisions and trade-offs were made: first, even though Flight is simply a list of tricks, I did not implement a proper recording function for it as it was not specified in the instruction. To that end, a client must iterate himself through the flight and call the record method implemented in Trick to do so. Additionally, as per a reply of TA, Drone can only see movements in flight and execute movements. To that end, if a client wishes to edit a flight, it must go through using methods directly from the Flight class. Lastly, I have assumed that the “compare by unique movements “f in Q4 meant “unique movement directions”, thus allowing me to use the method I have created for that end in Q3.

Below is a class diagram that illustrate the key decisions made during the design of the solution:


Note: I did not rewrite methods of the interface on their implementing class, but assume all methods of interface are override to classes.
«interface» Iterable<Direction>
Iterator<Direction> iterator()
«interface» Iterator<Direction>
boolean hasNext()
Direction next()
Drone
String name
Direction direction
double distance
Speed speed
Format format
Boolean isRecorded
void getInfo()
void reset()
void record(Format f)
void doTrick(Trick trick, boolean record)
getFlight(Flight f)
runFlight(Flight f)
«interface» Movement
void execute()
void setMovement(Direction d, double dist, Speed s)
boolean isRecorded()
boolean isEmpty()
Flight
ArrayList<Trick> flight
String name
void addTrick(Trick t)
int uniqueMove()
void runFlight()
RunDrone
void main(String args[])
(Enum) Format
MP4, MQV, WMV, AVI, FLV, MKV
(Enum) Speed
LOW, MODERATE, HIGH
(Enum) Tricks
TakeOff, Pucker, Land
Trick
Tricks trick
ArrayList<Direction> directions
ArrayList<Double> distances
ArrayList<Speed> speeds
Format format
Boolean isRecorded = false
void record()
void set(Tricks trick)
void takeOff()
void pucker()
void land()
«interface» Comparator<Flight>
intint compare(Flight o1, Flight o2)
TricksNumberComparator
UniqueMoveComparator