In my design, the client creates a RemoteControl object which takes in a Drone object (the clients drone), and configures the moves for this drone: they can add a move, remove the last move in the sequence, or remove all moves. If the moves are invalid, I throw Exceptions and indicate why they are invalid. Once the moves are configured, the client can execute the flyDrone command on the remote and the drone will begin to fly. The moves in the sequence are of the type DroneMoves, where each move has either a **final** filename, distance, or neither. Some trade-offs with this design is that each move is assigned a filename/distance/move, but whether or not each field is used depends on the move itself (some assigned fields are never used).

I also considered the Drone object (the client's drone being passed to the RemoteControl) as having a 2D array to store flight history. This information could be important to the client. I have a public method that prints the history to the screen. I also have a getFlights method important for the RemoteControl that requires a password that is not accessible to the client.