Unit requirements:

LngLat:

- This class should be able to represent a coordinate in terms of its longitude and latitude values.
- This class should be able to calculate its Euclidean straight-line distance to another object of the same class.
- This class should be able to return a new object of the same class after moving in a certain direction.
- This class should be able to check whether an object of this class is close to (less than 0.00015 degrees) another object of the same class.

Client:

- JSON data should be able to be read from the REST server and be deserialised into corresponding objects.

JsonWriter

- A list of paths that the drone takes should be able to be written out in a GeoJSON format.
- There should be a way to find out if a point is within or on the boundary of an arbitrary polygon.
- There should be a way to portray each restaurant and no-fly zone vertices as a node in a graph.

Card:

- This class should store credit card details. (CVV, expiry date, card number)
- Objects of this class should be able to be validated against credit card conventions.

Direction:

- This enum class should contain 16 compass directions.
- Objects of this enum class should be able to return its compass direction in Double format.
- Objects of this enum class should be able to return its opposite direction.

Integration requirements:

Drone:

- This class should integrate with LngLat to display and reset its current coordinates.
- This class should integrate with Move to follow a series of Move objects and update its coordinates and battery levels.

Restaurant:

- This class should integrate with LngLat to show the restaurant's coordinates and menu.

Order:

- This class should integrate with Card for the following requirements:
    - This class should contain the following information about an order:
        - Order number

        - Credit card
        - Price in pence
        - Order items

ValidatedOrder:

- This class should integrate with Order to allow a delivery status enum to be attached.

OrderChecker:

- This class should integrate with Order, ValidatedOrder, and Restaurant to validate orders and attach relevant delivery enum values to each order.

Move:

- This class should integrate with LngLat and Order to show one single move that the drone takes, and which order it is currently carrying.

Polygon:

- This class should be able to represent an arbitrary sided polygon.
- This class should be able to check whether a point is inside a polygon by integrating with the LngLat class.
- This class should be able to check whether a line intersects a polygon by integrating with the LngLat class.

PathFinder:

- This class should integrate with LngLat, Polygon, and Move to find the shortest distance from the starting coordinate to the destination and return a list of Move objects.

Graph:

- This class should integrate with LngLat along with its inner classes Edge and Node to portray nodes and their respective undirected edges to each other.
- This class should integrate with Polygon to draw a visibility graph based on the no-fly zones provided.

JsonWriter:

- This class should integrate with ValidatedOrder and Move to satisfy the following requirements:
    - A list of paths that the drone takes should be able to be written out in GeoJSON format.
    - A list of orders for that day should be written out to a file, detailing its attributes as well as its delivery status.
    - A flightpath of the drone should be written out containing information about the drone's previous and current location, its bearing, and the order that it is carrying.

System requirements:

- The system should be able to produce a flightpath that maximises the sampled average number of validated pizza orders delivered before the drone's battery is exhausted.

- The system should be able to find the shortest distance for the drone to fly from one point to the other without entering the no fly zones
- The system should be able to output the flight path into a file that is in accordance with the file structure specified in the ILP documents.