**How To Run**

1. Import the zip file in NetBeans IDE.
2. Run the application.
3. Select Source Node where to start from.
4. Click on Start to fly the Drone
5. Run the Drone again and again between same source and destination to make the machine learn the path
6. Check the console logs to understand the process more clearly

**Expected Behavior:**

After few runs, the drone will get the stored path instead of finding the new one. If drone is traversing on stored path, you will see a path colored white, else if it is a new path, traversed path will be blue colored.

On the screen you will see red and orange colored nodes. These are the anomalies. Red being the anomaly which is of high intensity and node should change the path once it occurs. Second one is the orange anomaly which is of low intensity, but the drone can go on the node with this anomaly, it’s just it will consume more fuel.

The current position is represented by the green color.

The traversed path is represented by blue color.

If the drone gets stored path, the whole path is colored with white color, and drone follows the same path until it faces another anomaly.

**Working**:

Code Inside Timer() method in MainFrame:

First Drone gets the Source and Destination from user input. Then using the source as key it check the HashTable to see if path exists for this Source, Destination pair. If yes, then it uses the path and start traversing through that path. If no path available, it will use the **shortestDistance** algorithm to determine the next node. And at next node, again check if the path exists of not.

Once the drone reaches destination, all the traversed paths are stored in the HashTable.

**Class Description**

To show the node (points on map which the drone will use to traverse, I have used labels on JPanel). Each label is associated with a Node. Let’s call this as a maze.

**Business Classes:**

**Drone**:

This is the class which simulate/represents a drone object. It has properties of a drone, like its fuel, the position information, record of nodes traversed, list of nodes from the learned/stored path.

It has method “getAdjacentNodes(Node n)” which takes current node as argument and return a list of nodes surrounding the current node.

It also as “shortestDistancePathAlgo()” which takes the list of adjacent nodes and returns a node which is closest to the destination node using the distance formula.

**Node:**

This class represents each point in the maze. Each node has properties like its coordinates and type of anomaly on it.

**PathNode:**

PathNode is individual node in the LikedList {LinkedList is the value of (key, value) pair of the Machine Learned HashMap} which consist of an ArrayList which stores the nodes from a source to a destination. This basically represents path from source to destination and properties of this path like, distance, path confidence and fuel consumes. These factors are used to determine which the better path is.

**PathLinkedList:**

This is the Value part of the Key, Value pair. It consist of the head node which always points to the best route available. The best route is determined based on the confidence factor. The number of times drone follows the same path, the confidence factor increases and thus each time all the nodes in LinkedList are shuffled to make the best node at the root.

**MachineLearning:**

This class consist of the HashMap, which is used to store the data of learned paths. Path from source to destination is fetched from this HashMap.

**UtilityD:**

This class stores the utility methods like the one to color the nodes based on the anomalies, get the nodes from JPanel etc.

**DB4OUtil:**

This class is a utility to store object. Everytime the application is ran, the object data is stored in the file system. Next time we start the application, our learned path information is loaded along with the application. So we make objects persistent.

**UserInterface:**

**MainFrame:**

This class consist of the main method. This user interface class represents the Maze. Also, it has the run method which has the logic to fly the drone.

Timer() is the method which has the main logic to perform node search and do machine learning. It works like, first Drone gets the Source and Destination from user input. Then it check the HashTable to see if path exists for this Source, Destination pair, if yes, then it uses the path and start traversing through that path. If no path available, it will use the **shortestDistance** algorithm to determine the next node. And at next node, again check if the path exists of not.

Once the drone reaches destination, all the traversed paths are stored in the HashTable.