STARLINK ISL DOCUMENTATION

Problem Statement :-

We have to send message from one ground station to another ground station with the minimization of hops
The communication is done via GSOsatellite LEOsatellites and GroundStations.

CLASS DESCRIPTION OF THE PROJECT :-

1. Constellation
2. Satellite
3. GSOsatellite
4. LEOsatellite
5. GroundStation
6. Driver

- **CONSTELLATION** :-

Constellation in the original starlink project is set to be the collection of all the satellites (in our simulation it is LEOsatellite and GSOsatellite) .

Constellation is extended by Satellite Class

- **SATELLITE** :-

Satellite is a small part of the humongous starlink constellation and contains a few LEOsatellites and a GSOsatellite .

(in our simulation we have 1 GSOsatellite and 5 LEOsatellites)

Satellite class implements the communication interface and is extended by LEOsatellite and GSOsatellite class .

- **LEOSATELLITE**:-

LEO satellite is a low earth orbit satellite part of which receives and sends messages to the ground stations it is also capable of inter leo satellite communication.
Leo Satellite extends the Satellite class and implements the Runnable Interface .

- **GSOSATELLITE**:-

GSO satellite is a geostationary orbit satellite which receives messages from one LEO satellite and sends them to the other satellite .

GSO satellite extends the Satellite class and implements the Runnable Interface .

- **GROUNDSTATION** :-
  Ground Stations are various stations which can send messages to LEOsatellites and receive messages from the LEOsatellites.

  Groundstation implements the runnable interface and communication interface .

- **DRIVER :-**

  Initialises all objects and asks the user for source and destination of message.

INTERFACE DESCRIPTION :-

- **COMMUNICATION INTERFACE**

  It is an interface which contains the sendMessage function and is implemented by the ground station and satellite class.

Additional Classes :-

- **COLORS** :

  Contains the ANSI ESCAPE CODES which are used to print the colored text in the console for the unix operating system .

  Reference [Print coloured output in the console]

```
package Colors;

public class Colors {

    public static final String RESET = "\033[0m";
    public static final String RED = "\033[0;31m";
    public static final String GREEN = "\033[0;32m";
    public static final String PURPLE = "\033[1;95m";
    public static final String YELLOW = "\u001B[33m";
    public static final String CYAN = "\u001B[36m";
}
```

- **WrongInput :**

  Helps in printing the wrong input detected by the user (if any)

CODE EXPLANATION : -

1. First we are taking the number of messages the user wants to send .

```
System.out.println("\n"+Colors.PURPLE+"Enter the number of messages you want to send from one
groundstation station to another"+Colors.RESET+"\n");
Scanner scr = new Scanner(System.in)  ;
numberOfMessages = scr.nextInt() ;
```

2. Then we ask the user for the source and destination of the message and also check if the input entered by the user is valid .

   Since we are given that groundstations are indexed from 0 to 9 any input beyond this would be invalid.

```
System.out.println(Colors.YELLOW+"Enter the intial ground station and final ground station"
+Colors.RESET);
int initialGroundStation = scr.nextInt() ;
int finalGroundStation = scr.nextInt() ;


try {
    if(initialGroundStation>9||initialGroundStation<0||finalGroundStation>9||
    finalGroundStation<0){
        throw new WrongInput(Colors.CYAN+"Invalid Input Entered By User"+Colors.RESET);
    }
} catch (WrongInput e) {
    System.out.println(e);
    continue;
}
```

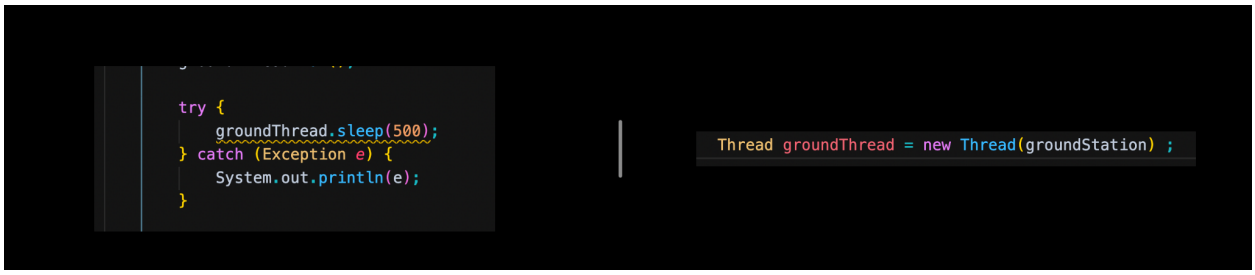
3. Then we are creating the object for the groundstation and passing the initial and final ground station number in the ground station constructor .

```
GroundStation groundStation = new GroundStation(initialGroundStation , finalGroundStation);
```

```
package Classes;
import Colors.Colors;

public class GroundStation implements CommunicationInterface , Runnable {

    private int intialPosition ;
    private int finalPosition ;

    public GroundStation(int intialPosition , int finalPosition){
        this.intialPosition = intialPosition ;
        this.finalPosition = finalPosition ;
    }
```

4. After this we are making a thread of the ground station which is implementing the runnable interface and the run method of the thread calls the sendmessage function .

   We also sleep the thread for 500 milliseconds to show latency in the transfer .



```
try {
    groundThread.sleep(500);
} catch (Exception e) {
    System.out.println(e);
}
```

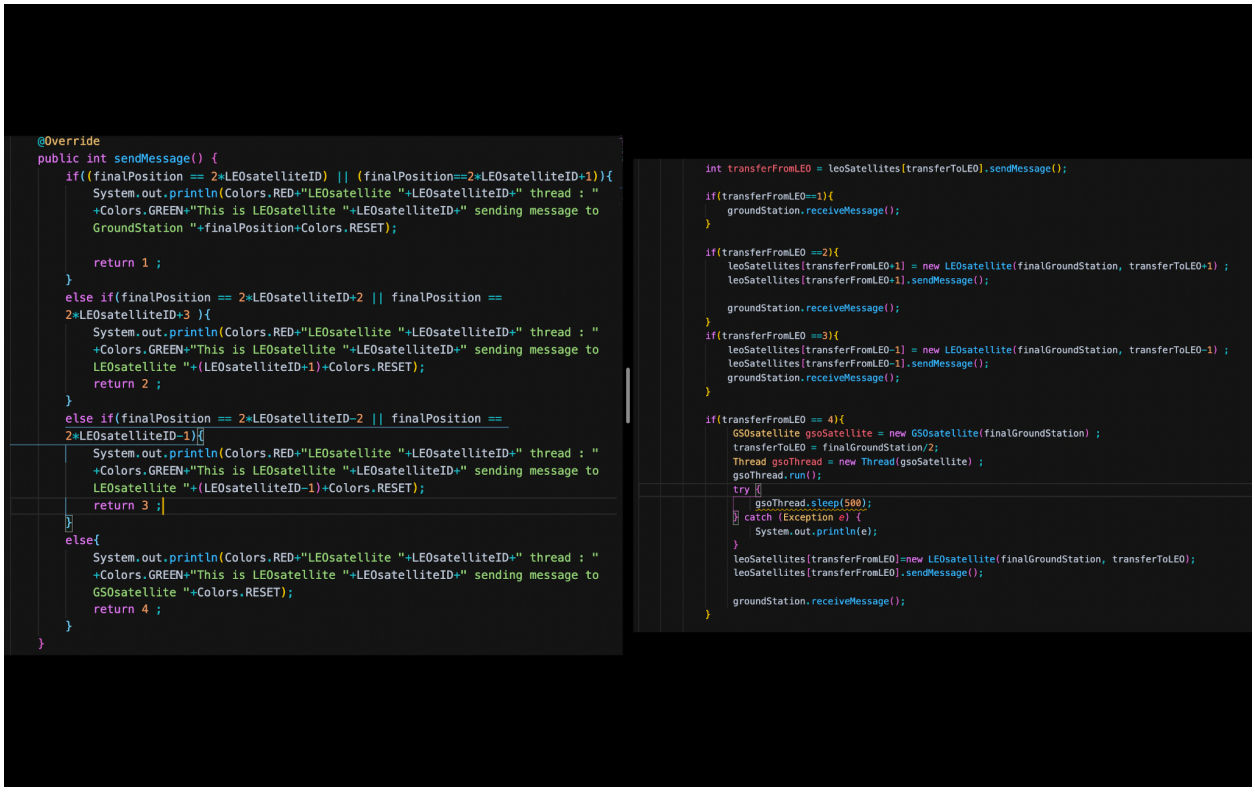
```
Thread groundThread = new Thread(groundStation) ;
```

5. After this we are making an array of objects of LEO satellites and we are running the ground thread after it .

```
LEOsatellite [] leoSatellites = new LEOsatellite[5] ;
groundThread.run();
```

6. Now we are sending messages from the LEOsatellite and checking whether it should be a (LEO to ground station transfer) or (LEO to LEO) or (LEO to GSO) transfer .

   The sendmessage function of the leo satellite returns either of 4 values from which we check the above transfers



```
@Override
public int sendMessage() {
    if((finalPosition == 2*LEOsatelliteID) || (finalPosition==2*LEOsatelliteID+1)){
        System.out.println(Colors.RED+"LEOsatellite "+LEOsatelliteID+" thread : "
        +Colors.GREEN+"This is LEOsatellite "+LEOsatelliteID+" sending message to
        GroundStation "+finalPosition+Colors.RESET);

        return 1 ;
    }
    else if(finalPosition == 2*LEOsatelliteID+2 || finalPosition ==
    2*LEOsatelliteID+3 ){
        System.out.println(Colors.RED+"LEOsatellite "+LEOsatelliteID+" thread : "
        +Colors.GREEN+"This is LEOsatellite "+LEOsatelliteID+" sending message to
        LEOsatellite "+(LEOsatelliteID+1)+Colors.RESET);
        return 2 ;
    }
    else if(finalPosition == 2*LEOsatelliteID-2 || finalPosition ==
    2*LEOsatelliteID-1){
        System.out.println(Colors.RED+"LEOsatellite "+LEOsatelliteID+" thread : "
        +Colors.GREEN+"This is LEOsatellite "+LEOsatelliteID+" sending message to
        LEOsatellite "+(LEOsatelliteID-1)+Colors.RESET);
        return 3 ;
    }
    else{
        System.out.println(Colors.RED+"LEOsatellite "+LEOsatelliteID+" thread : "
        +Colors.GREEN+"This is LEOsatellite "+LEOsatelliteID+" sending message to
        GSOsatellite "+Colors.RESET);
        return 4 ;
    }
}
```

```
int transferFromLEO = leoSatellites[transferToLEO].sendMessage();

if(transferFromLEO==1){
    groundStation.receiveMessage();
}

if(transferFromLEO ==2){
    leoSatellites[transferFromLEO+1] = new LEOsatellite(finalGroundStation, transferToLEO+1) ;
    leoSatellites[transferFromLEO+1].sendMessage();

    groundStation.receiveMessage();
}
if(transferFromLEO ==3){
    leoSatellites[transferFromLEO-1] = new LEOsatellite(finalGroundStation, transferToLEO-1) ;
    leoSatellites[transferFromLEO-1].sendMessage();
    groundStation.receiveMessage();
}

if(transferFromLEO == 4){
    GSOsatellite gsoSatellite = new GSOsatellite(finalGroundStation) ;
    transferToLEO = finalGroundStation/2;
    Thread gsoThread = new Thread(gsoSatellite) ;
    gsoThread.run();
    try {
        gsoThread.sleep(500);
    } catch (Exception e) {
        System.out.println(e);
    }
    leoSatellites[transferFromLEO]=new LEOsatellite(finalGroundStation, transferToLEO);
    leoSatellites[transferFromLEO].sendMessage();

    groundStation.receiveMessage();
}
```

Explanation for all four return statements :-

Return 1
Ground station will receive message

Return 2
Leo satellite will send message to the right leo satellite which will send message back to ground station

Return 3
Leo satellite will send message to the left leo satellite which will send message back to ground station

Return 4
Leo satellite will send message to GSO satellite and GSO satellite thread will send message to another Leo satellite which will return message to final ground station

Send Message function for GSO satellite

```
public void receiveMessage() {
    System.out.println(Colors.RED+"GroundStation "+finalPosition+" thread :"
    +Colors.GREEN+" This is GroundStation "+finalPosition+". recieved message."
    +Colors.RESET);
}
```

Ground Station will Receive Message from

```
@Override
public int sendMessage() {
    System.out.println(Colors.RED+"GSOsatellite 0 thread : "+Colors.GREEN+"This is
    GSOsatellite 0. sending message to LEOSatellite "+finalPosition/2+Colors.RESET
    );
    return finalPosition/2 ;
}
```

SAMPLE TEST RUNNING

CASE 1 => 1 9
CASE 2 => 0 1
CASE 3 => 0 2

```
Enter the number of messages you want to send from one groundstation station to another

3
Enter the intial ground station and final ground station
1 9
GroundStation 1 thread : This is GroundStation 1 sending message to LEOsatellite 0
LEOsatellite 0 thread : This is LEOsatellite 0 sending message to GSOsatellite
GSOsatellite 0 thread : This is GSOsatellite 0. sending message to LEOSatellite 4
LEOsatellite 4 thread : This is LEOsatellite 4 sending message to GroundStation 9
GroundStation 9 thread : This is GroundStation 9. recieved message.

Enter the intial ground station and final ground station
0 1
GroundStation 0 thread : This is GroundStation 0 sending message to LEOsatellite 0
LEOsatellite 0 thread : This is LEOsatellite 0 sending message to GroundStation 1
GroundStation 1 thread : This is GroundStation 1. recieved message.

Enter the intial ground station and final ground station
0 2
GroundStation 0 thread : This is GroundStation 0 sending message to LEOsatellite 0
LEOsatellite 0 thread : This is LEOsatellite 0 sending message to LEOsatellite 1
LEOsatellite 1 thread : This is LEOsatellite 1 sending message to GroundStation 2
GroundStation 2 thread : This is GroundStation 2. recieved message.
```

Explanation of sample test cases =>

Refer to the path highlighted in the below diagram