OOP Project Documentation

Project Name: Starlink Inter-Satellite Link (ISL) for Communication Satellites (Project 7).
Group No.: 134.
Team Member 1: Anish Atul Kulkarni (2020A7PS0975P).
Team Member 2: Shivam Abhay Pande (2020A7PS0124P).
Github Link: https://github.com/shiworm/oop-project-134

1. Required Software and Executing the file

Required Software: OpenJDK 17+. All imports are done from the standard Java library. Preferably run in Linux-based OS due to a bug (mentioned in section 6). For Windows, either run the code in VS Code or ConEmu Terminal installed from :

ConEmu | Downloads
'Preview' builds are recommended for most users. Read more about release stages. These mirrors are not maintained by project owners! Searching for some old (exact) ConEmu build? All builds
https://conemu.github.io/en/Downloads.html

As Windows cmd prompt does not support ANSI colored outputs.

Execution

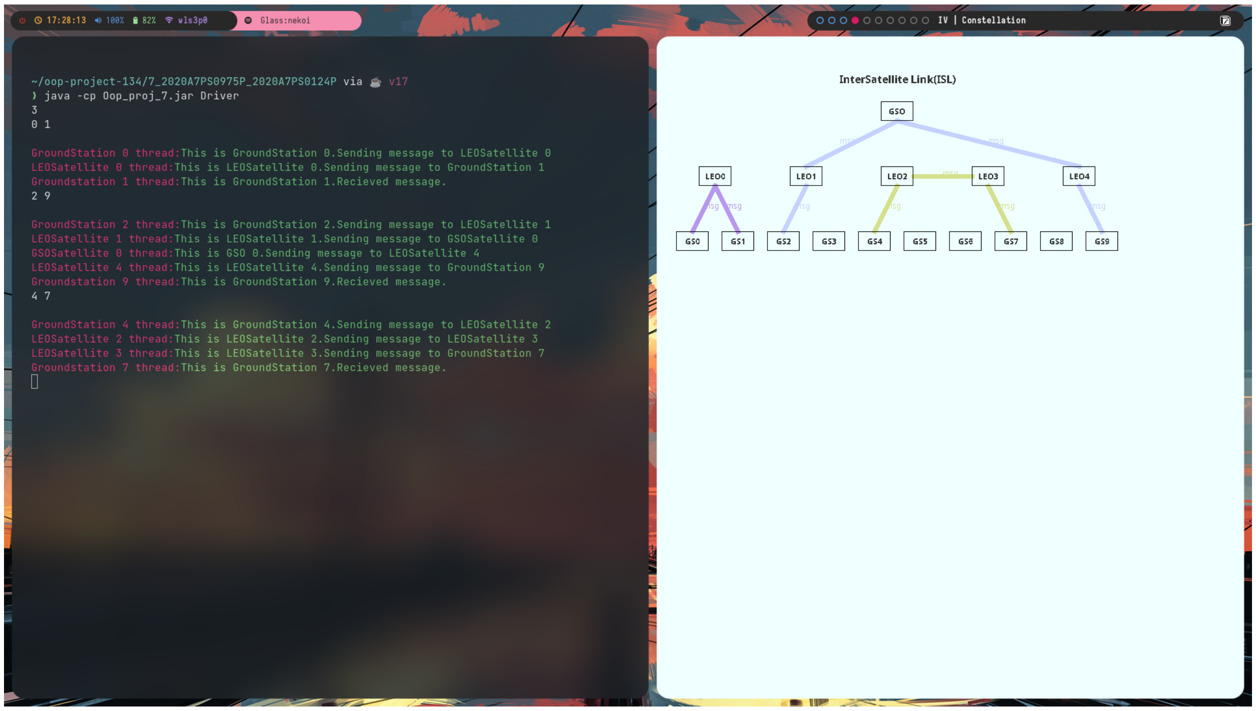
Download the ZIP file and extract it's contents in a folder. After extracting the folder run the following commands:

```
$ cd 7_2020A7PS0975P_2020A7PS0124P
$ ls /*Listing function. For Windows enter "dir"*/
```

If the contents of the ZIP file are shown then to run the project run the command:

```
$ java -cp Oop_proj_7.jar Driver
$ /*Sample Input*/
```

After this enter the testcase in place of Sample Input and run. The output should be as follows:



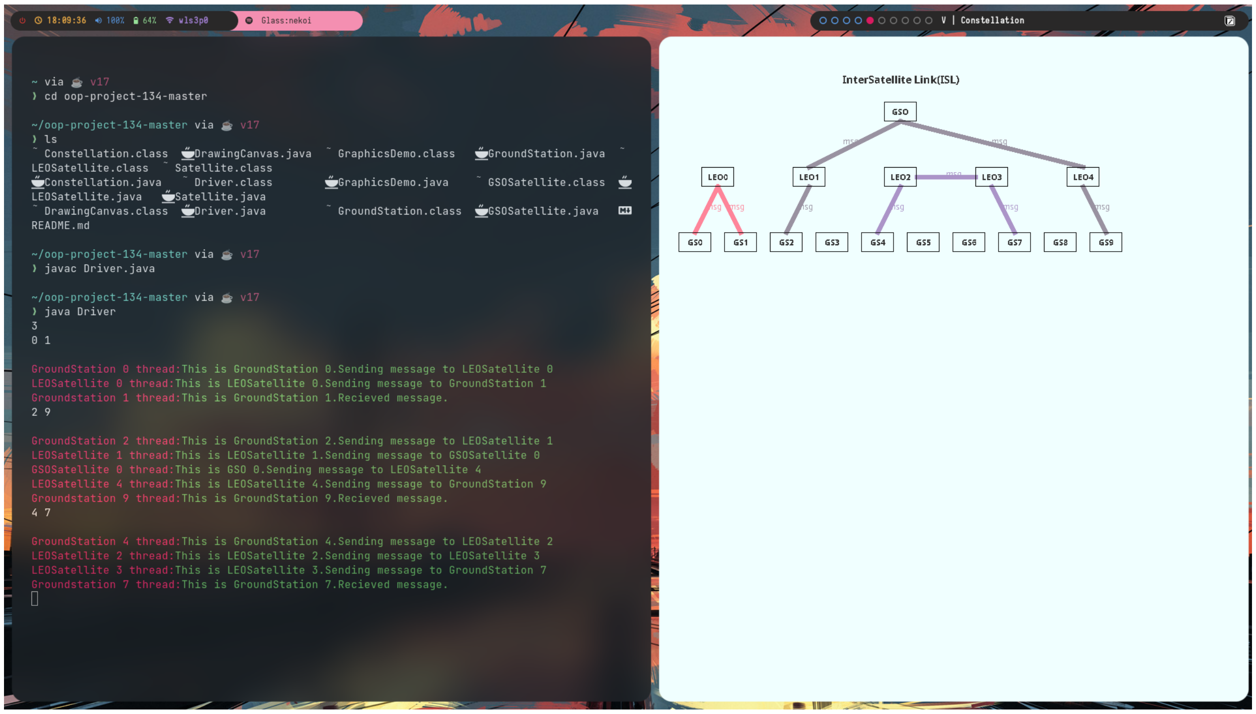
```
~/oop-project-134/7_2020A7PS0975P_2020A7PS0124P via ☕ v17
❯ java -cp Oop_proj_7.jar Driver
3
0 1

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.
2 9

GroundStation 2 thread:This is GroundStation 2.Sending message to LEOSatellite 1
LEOSatellite 1 thread:This is LEOSatellite 1.Sending message to GSOSatellite 0
GSOSatellite 0 thread:This is GSO 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.
4 7

GroundStation 4 thread:This is GroundStation 4.Sending message to LEOSatellite 2
LEOSatellite 2 thread:This is LEOSatellite 2.Sending message to LEOSatellite 3
LEOSatellite 3 thread:This is LEOSatellite 3.Sending message to GroundStation 7
Groundstation 7 thread:This is GroundStation 7.Recieved message.
```

In case of any problems in running the code using above method(.jar file execution), run the following commands :

https://github.com/shiworm/oop-project-134

```
$ git clone https://github.com/shiworm/oop-project-134
$ cd oop-project-134 /*Or whatever file name gets clones*/
$ ls /*This should print all the files from github. For windows enter "dir"*/
$ javac Driver.java
$ java Driver
/*Sample Input*/
```

The output would be as follows:



```
~ via ☕ v17
❯ cd oop-project-134-master

~/oop-project-134-master via ☕ v17
❯ ls
 Constellation.class  DrawingCanvas.java   GraphicsDemo.class   GroundStation.java  
LEOSatellite.class   Satellite.class
Constellation.java    Driver.class         GraphicsDemo.java    GSOSatellite.class  
LEOSatellite.java    Satellite.java
 DrawingCanvas.class  Driver.java          GroundStation.class  GSOSatellite.java   
README.md

~/oop-project-134-master via ☕ v17
❯ javac Driver.java

~/oop-project-134-master via ☕ v17
❯ java Driver
3
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.
2 9

GroundStation 2 thread:This is GroundStation 2.Sending message to LEOSatellite 1
LEOSatellite 1 thread:This is LEOSatellite 1.Sending message to GSOSatellite 0
GSOSatellite 0 thread:This is GSO 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.
4 7

GroundStation 4 thread:This is GroundStation 4.Sending message to LEOSatellite 2
LEOSatellite 2 thread:This is LEOSatellite 2.Sending message to LEOSatellite 3
LEOSatellite 3 thread:This is LEOSatellite 3.Sending message to GroundStation 7
Groundstation 7 thread:This is GroundStation 7.Recieved message.
```

2. Description of Classes

a. `Constellation` Class

This class is the parent class of each component class of a constellation. It saves `static` variables from the input and is later used by the children classes to function.

b. `Satellite` Class

This class stores a `static ArrayList` of the ID's of the GroundStations and LEOSatellites which is later used to determine which Satellite or GroudStation is to receive the message.

c. `GroundStation` Class

This class contains two important methods:

1. `sendMsg()`: If a GroundStation thread is initialized for the first time, `countergs` will be 0 and hence would executed. This further initlaizes the first LEOSatellite thread and increments `countergs` so that the next time GroundStation thread is initalized, it will not send it back to another LEOSatellite.
2. `receivedMsg()`: If a GroundStation thread is initialized for the second time (i.e. `countergs` is 1), the GroundStation would receive the message and would finish the communication.

d. `LEOSatellite` Class

This class contains three important methods:

1. `sendMsgtoGSO()`: If the LEOSatellites that are able to communicate with the two GroundStations cannot communicate with each other (i.e. not adjacent), a thread for the GSOSatellite is initialized and mesage is passed on. `counter` is incremented so that if and when the LEOSatellite thread is called again, it would not run the same method again.

2. `sendMsgtoGS()`: If the initial and final GroundStations can communicate through a common LEOSatellite, we can send the message directly back to the final GroundStation. Here a GroundStation thread is initialized and message is passed on.

3. `sendMsgtoLEo()`: If the LEOSatellites that are able to communicate with the two GroundStations can communicate with each other (i.e. are adjacent), a thread for the second LEOSatellite is initialized and message is passed on. `counter` is incremented so that the next LEOSatellite thread called will definitely run the method to send the message to final GroundStation.

e. `GSOSatellite` Class

Once a GSOSatellite thread is running, it will run the `sendMsgs()` method and send the message to the LEOSatellite that can communicate with the final GroundStation and initializes the LEOSatellite thread for the same.

f. `GraphicsDemo` Class

This class initializes a `DrawingCanvas` object to draw the lines for the GUI Output of the code.

g. `DrawingCanvas` Class

This class draws lines in the Constellation diagram in the output as per the input GroundStations.

h. `Driver` Class

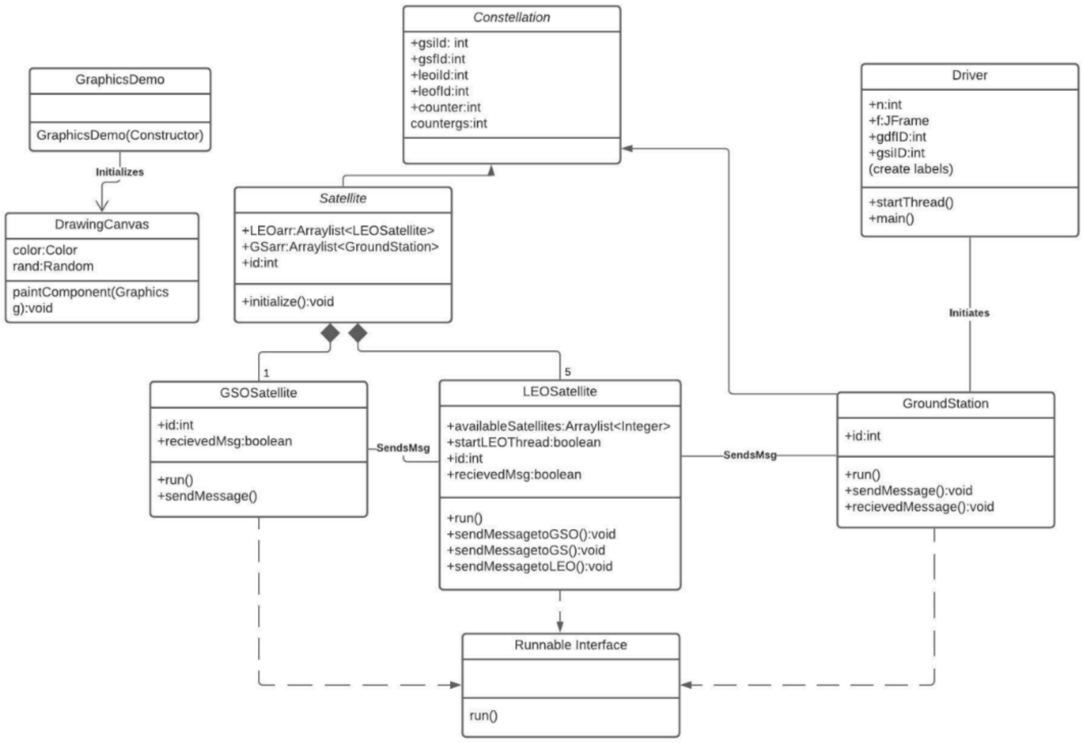
This class contains:

1. `initialize()`: It initializes the `ArrayList` inside the `Satellite` class and provides suitable objects to it.

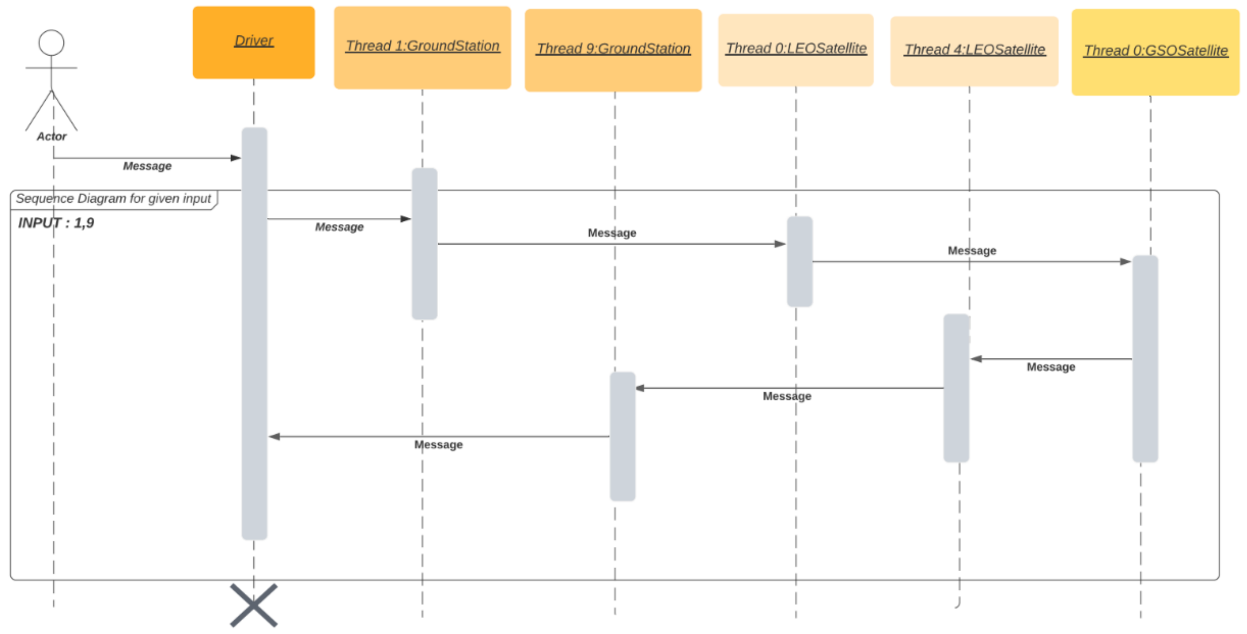
2. `main()`: The `main()` function initializes the JFrame and inside a `while` loop, takes inputs from the user. For each iteration it initializes the `counter`, `countergs` and the thread for the initial GroundStation.

3. UML Diagrams

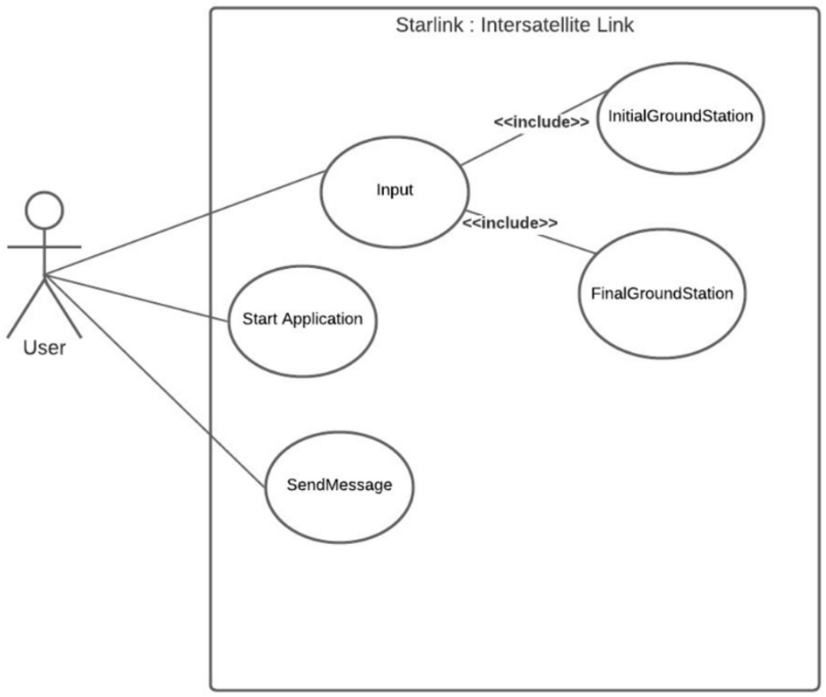
UML Class Diagram



UML Sequence Diagram (next page)

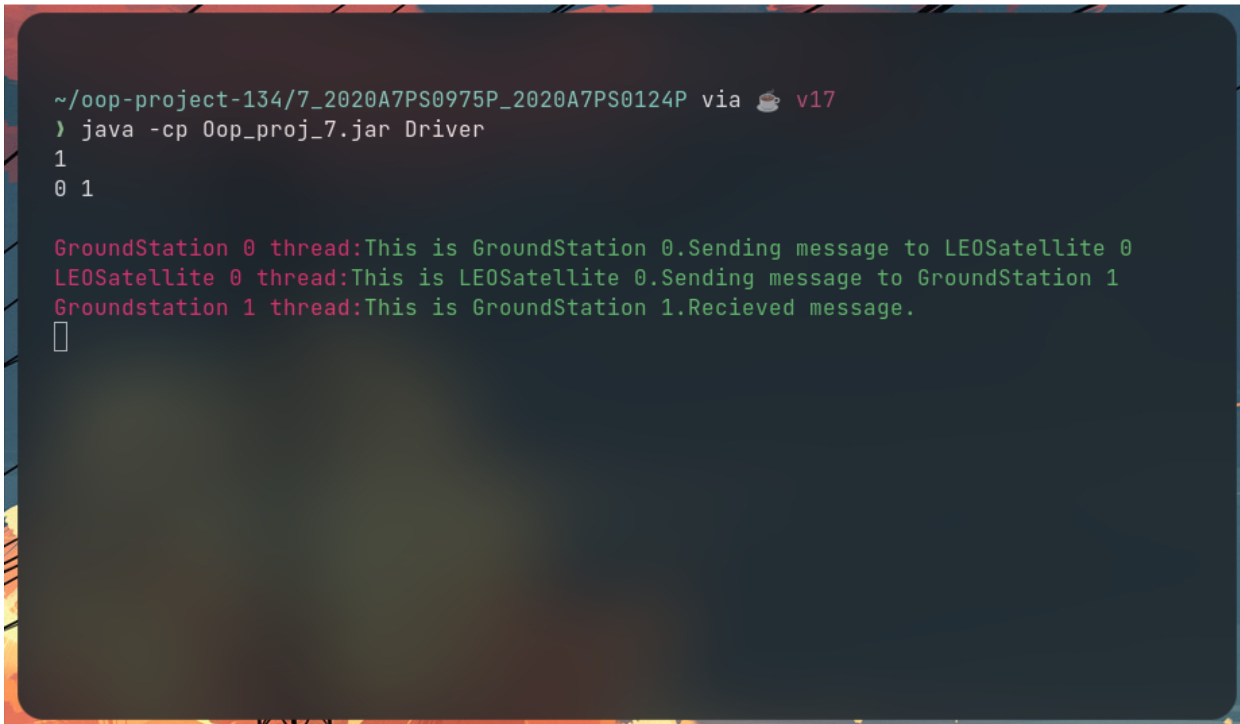


UML Usecase Diagram

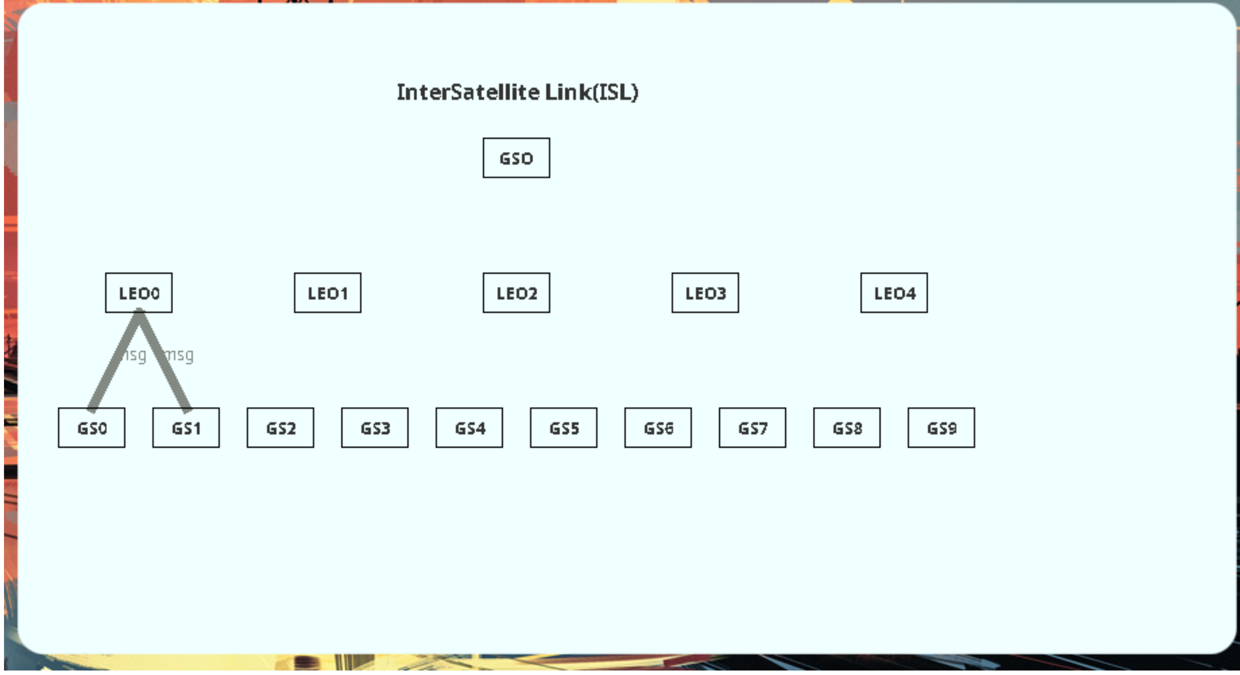


4. Analysis of Testcases

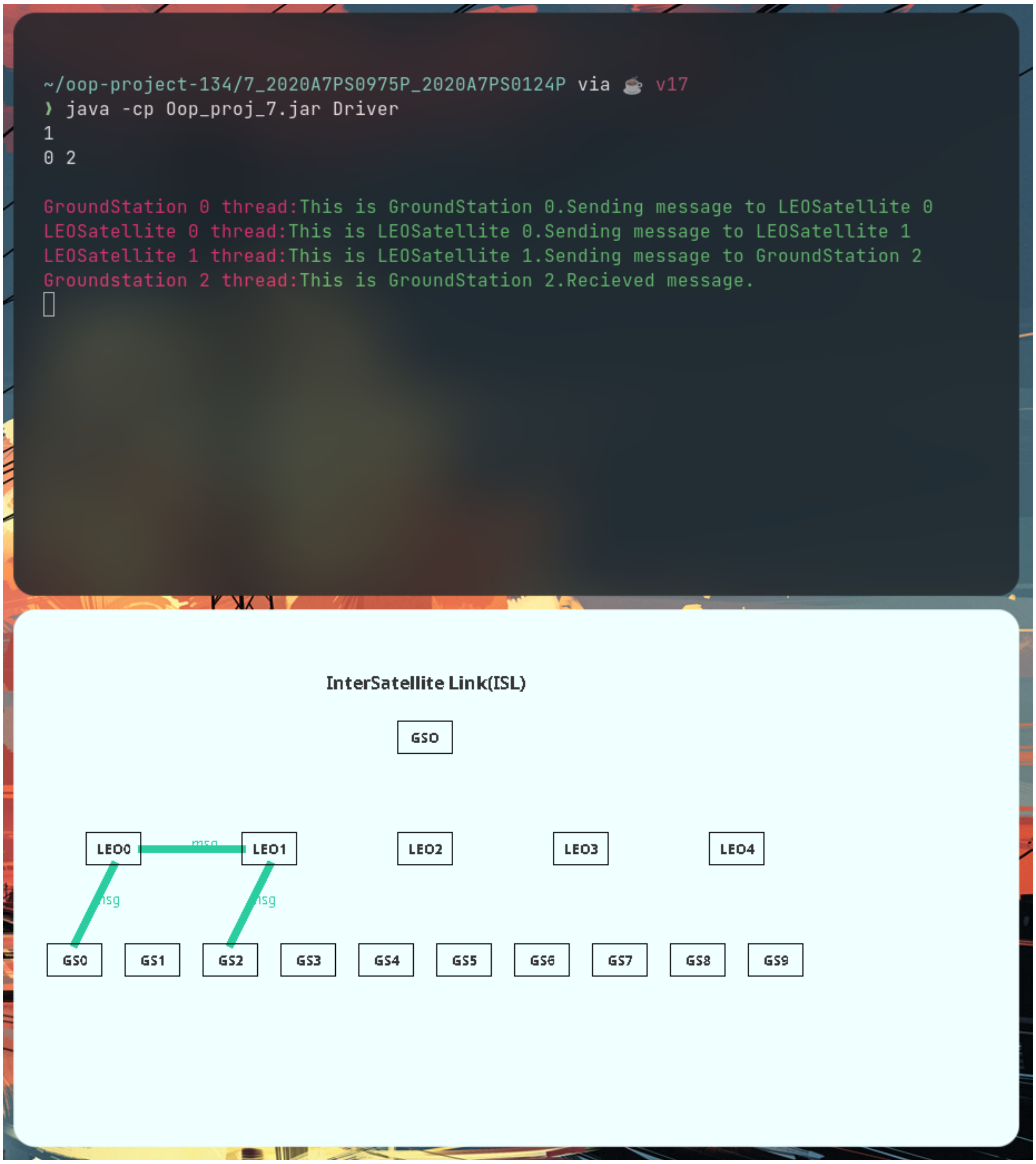
a. Case where the two GroundStations can communicate with a common LEOSatellite



```
~/oop-project-134/7_2020A7PS0975P_2020A7PS0124P via ☕ v17
❯ java -cp Oop_proj_7.jar Driver
1
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.
```

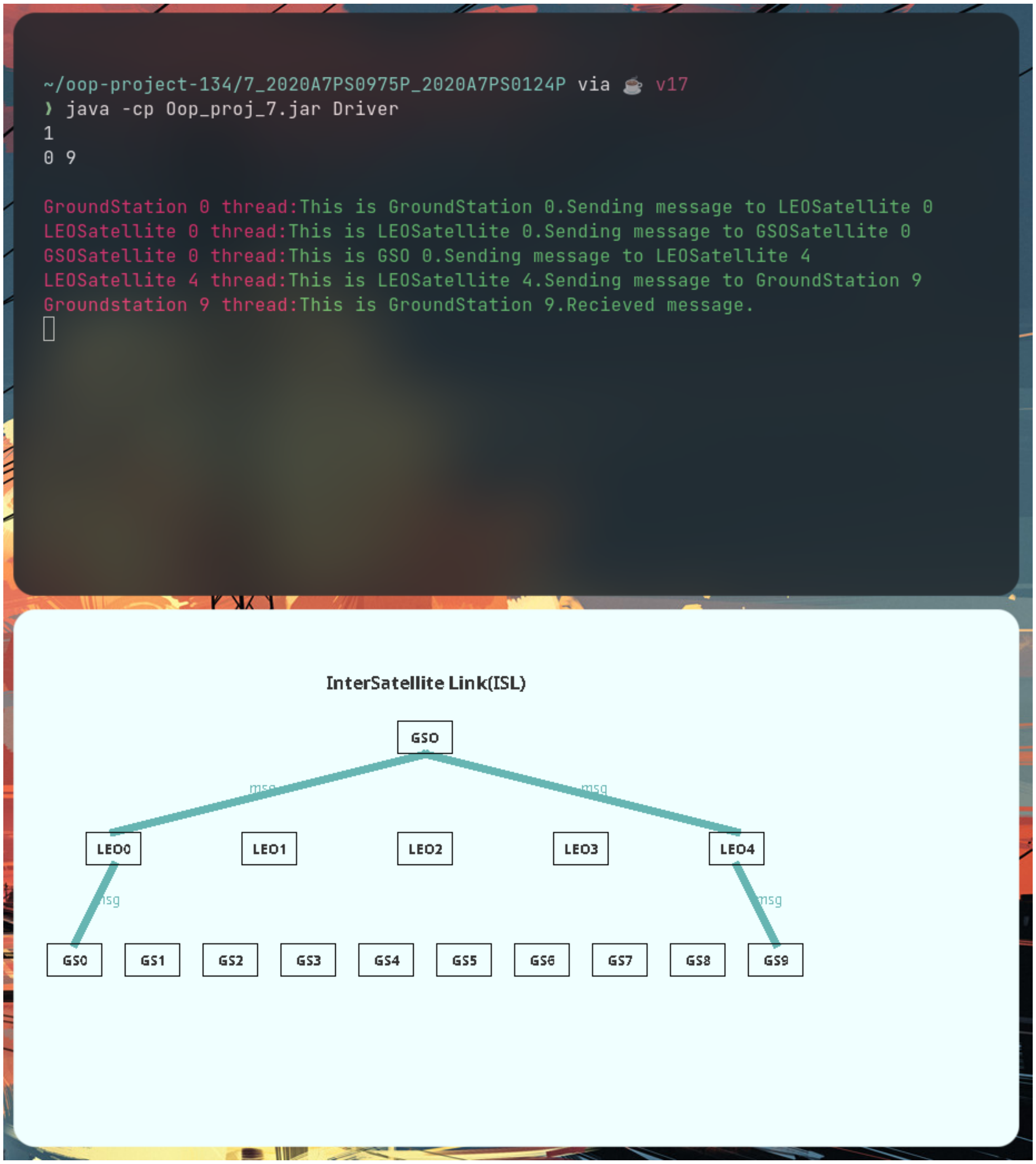


b. Case where the two LEOSatellites that can communicate with the GroundStations, can communicate.



```
~/oop-project-134/7_2020A7PS0975P_2020A7PS0124P via ☕ v17
❯ java -cp Oop_proj_7.jar Driver
1
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.
```

c. Case where the two LEOSatellites that can communicate with the GroundStations, cannot communicate.



```
~/oop-project-134/7_2020A7PS0975P_2020A7PS0124P via ☕ v17
❯ java -cp Oop_proj_7.jar Driver
1
0 9

GroundStation 0 thread:This is GroundStation 0.Sending message to LEOSatellite 0
LEOSatellite 0 thread:This is LEOSatellite 0.Sending message to GSOSatellite 0
GSOSatellite 0 thread:This is GSO 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.
```

5. OOP Principles and Design Pattern Analysis

a. Program to an Interface, not to an Implementation.

This project does not use any interface other than the runnable interface which is used for multithreading. Objects are only used for transmitting messages from one object to another. It might have been possible to make an interface for implementing the SendMessage function but we have not implemented that.

b. Favour Composition over Inheritance.

If the code does not contain many methods that need to be continuously reused there is not much use of inheritance.

Instead of GroundStation and LEOSatellite class inheriting Satellite class an ArrayList of objects of GroundStations and LEOSatellites is passed into the Satellite class. This is done because there are no other features in Satellite class that need to be accessed by these classes. Thus the Satellite class is composed of GroundStations and LEOSatellites.

c. Classes should be open for Extension, but closed for Modification.

If additional LEOSatellites and GroundStations are added such that $LEOSatellites = GroundStations * 2$ is still valid, the only thing we need to do is add the additional objects in the ArrayLists of GroundStations and LEOSatellites.

So, our code to some extent is open for extension. But if the above condition is not followed the code logic would have to be modified.

d. Strive for Loosely Coupled Designs between Objects.

In the Starlink project we have not been able to follow this guideline. For example,while calling threads of different classes we create objects of LEOSatellite in Groundstation and vice versa. Similarly, a LEOSatellite object is created in GSOSatellite. So any change in a class would affect other methods also. Therefore our project is tightly coupled to an extent.

e. Encapsulate what varies

There was no necessity to do encapsulation in our project. Our project does not contain any data that is object specific and that needs to be made private in classes.

Also, like In the example of “strategy design pattern”, each duck type had a unique fly behaviour associated with it. But in the case of LEOSatellites,any of its instances can either send a message to Groundstation or another LEOSatellite or a GSOSatellite. There is no fixed behaviour for a given satellite type. Due to this it would not be of much use to create a MessageBehaviour interface which is implemented by sendMessage classes in this case.

f. Depend on Abstraction,do not depend on Concrete Classes.

Our project does not have any abstract classes. There is not much need for inheritance in the code. Due to this there was no requirement of a method which is declared abstract in the parent class and later called by an object type of the child class. This point is related to the “favour composition over inheritance point”.

6. Limitations/Bugs

Limitations:

1. When a big number of inputs are provided at the same time, threads of multiple inputs were initialized, messing up the order of output and hence main thread is put to sleep for 100 ms which reduces efficiency.
2. Two types of GUI Outputs were developed by us:
   a. Only 1 Jframe is initilaized and all communications are shown together in one diagram. In this case, the overlapping of common communication lines were overlapped making it hard to understand the path of message for a large number of inputs.
   b. For n inputs, n Jframes were initialized in a previous version of our code. The communications were clear but for higher number of inputs, more Jframes were opened.

Bugs:

1. Whenever the Jframe is minimized and opened again, the color of communication lines change automatically.
2. In Windows for inputs that are pasted and happen to be more than 12, in the terminal it takes first 12 inputs and displays a few outputs and then takes the rest of the input. In Linux however, this issue does not occur.

7. Future Work Possible Extending this Project

Our project is a minor visualization of the Starlink Project initiated in 2016 by SpaceX. The major benefits of this constellation system is:

1. Fast and reliable communication assuring efficiency of message transmission. The existing broadbands have a latency of approx 600 ms while the Starlink aims to provide latencies between 25 to 35 ms which is a 20 times efficient system and without compromising the speed of data transfer at 1 Gbit/s, which is comparable with fiber or cable.
2. Broadband can also be provided to rural and remote places using this system. It is more cost effective as Starlink provides unlimited speed at $\$99$ while for the same amount you can only get 100Mbps down, 10Mbps up in cities and 10Mbps up, 1 Mbps down.
3. It has better disaster recovery and has superior portability.

Apart from just internet services, this project can be used in the future for disaster management by better communication. It has now been implemented only in a few states of the United States but in the future SpaceX intends to implement it globally, or as much as possible.