**CS F213**

**Object Oriented Programming**

Helper Document

Project Number: 19

Project Title: Token Bucket Algorithm

Group Number: 147

Team Members:

Saksham Mahajan (2019B4A70627P)

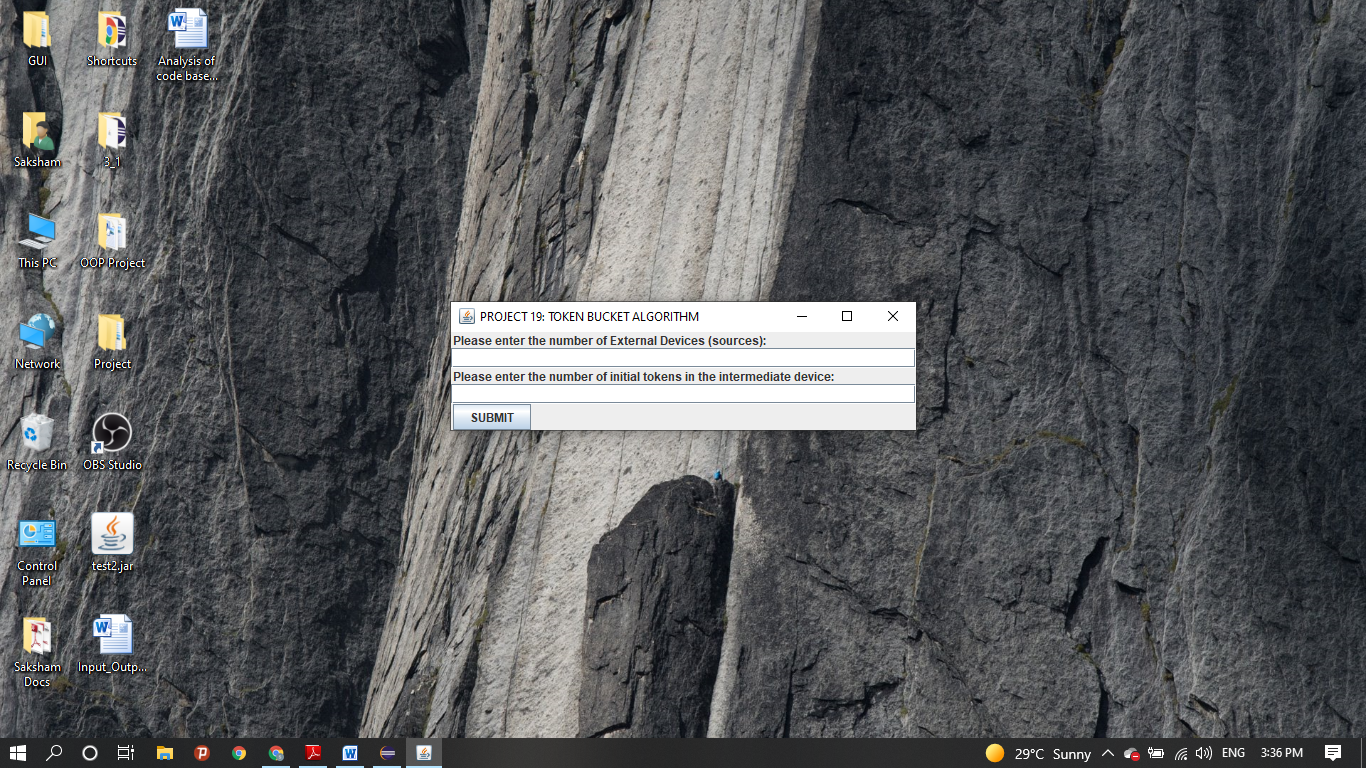
Ritik Thakur (2019B2A70878P)

**CONTENTS**

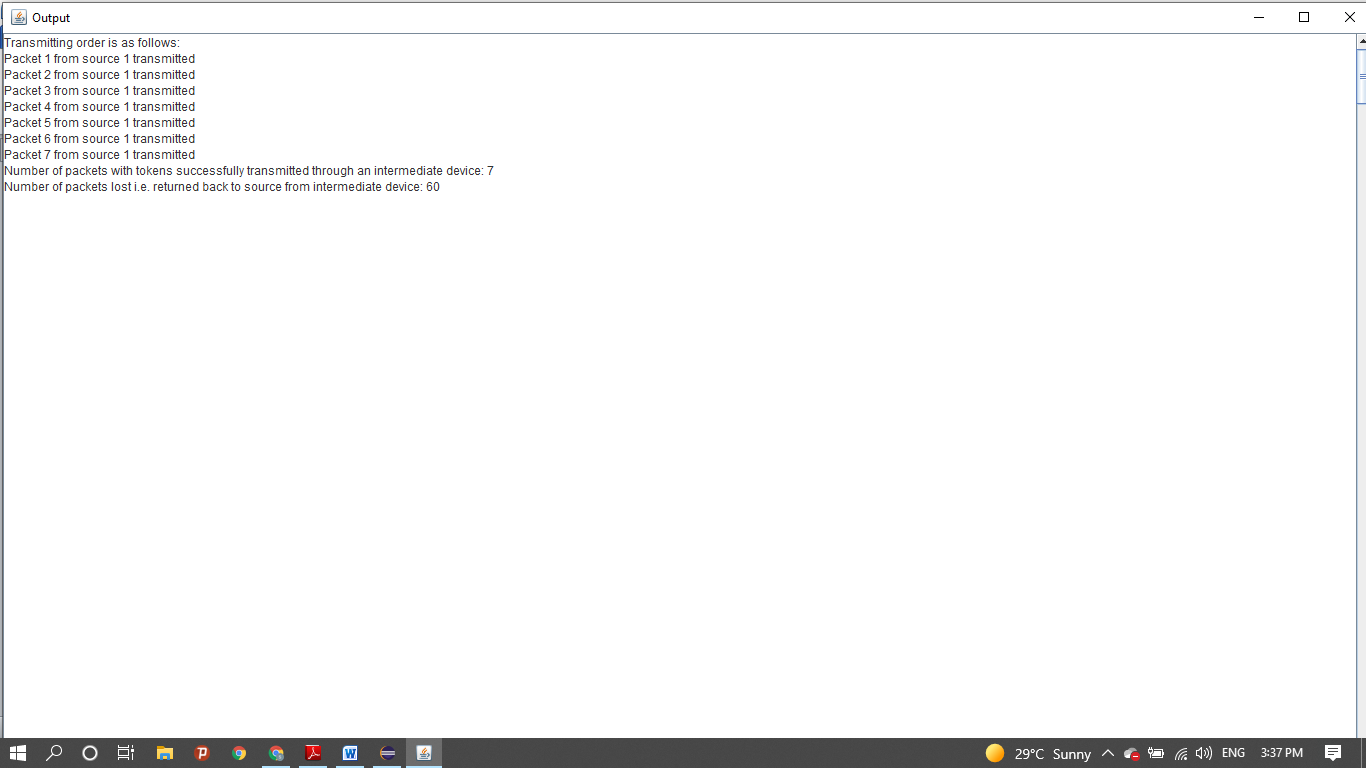
1. **Explanation of the code**
2. **Test cases that have been run**
3. **Additional information**
4. **UML diagrams**
5. **Explanation of the code**

The project assigned to us is based on Token Bucket Algorithm, which is used to average inflow rate of the traffic/network. Different devices send data(packets) to the network which can either be transmitted successfully or rejected depending on the availability of the tokens in the intermediate device. So the packets that are received in a burst are outputted in a specific transmitting order and the number of packets successfully transmitted and the number of packets lost are calculated.

* The program execution starts with the MainClass which creates an instance of CreateGUI class using constructor, thus creating the input screen.
* When the user enters the number of external devices and the number of tokens assigned to intermediate device and presses the submit button, the input screen is automatically closed and the input values are used to create instances of the ExternalDevice class and initialize the number of initial tokens in the IntermediateClass.
* Based on the static attribute ‘memory’ the number of packets that are passed to the IntermediateClass are decided and are added to an ArrayList while the rest of the packets are dropped immediately.
* Depending on the ArrayList the threads are created and started in succession using for loop.
* The next for loop that operates join() on each thread has been implemented to ensure that these threads run before the execution of the thread that follows this snippet which has been created using abstract runnable class.
* The run() is executed in the IntermediateDevice class and executes the transmit() and setTokens() methods that hold the logic for transmitting/rejecting and updating the number of tokens respectively.
* Inside the run block the time that passes has also been calculated so as to add tokens to the token bucket after each second using the setTokens() method.
* When all the threads have been executed an instance of the OutputScreen is created which renders the output screen.
* The constructor of the OutputScreen takes an ArrayList containing the strings that have to be printed on the output screen using the textArea object of the JFrame.



Input Screen



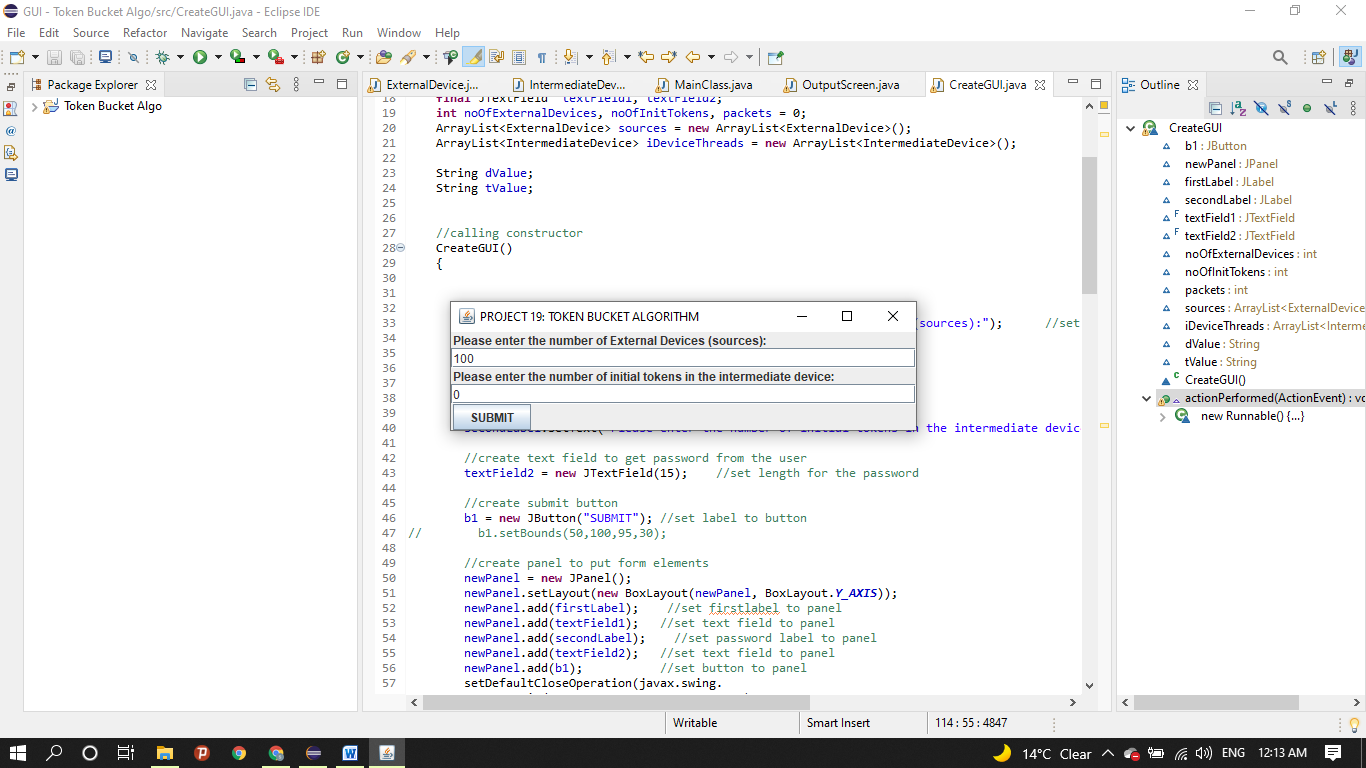
Output Screen

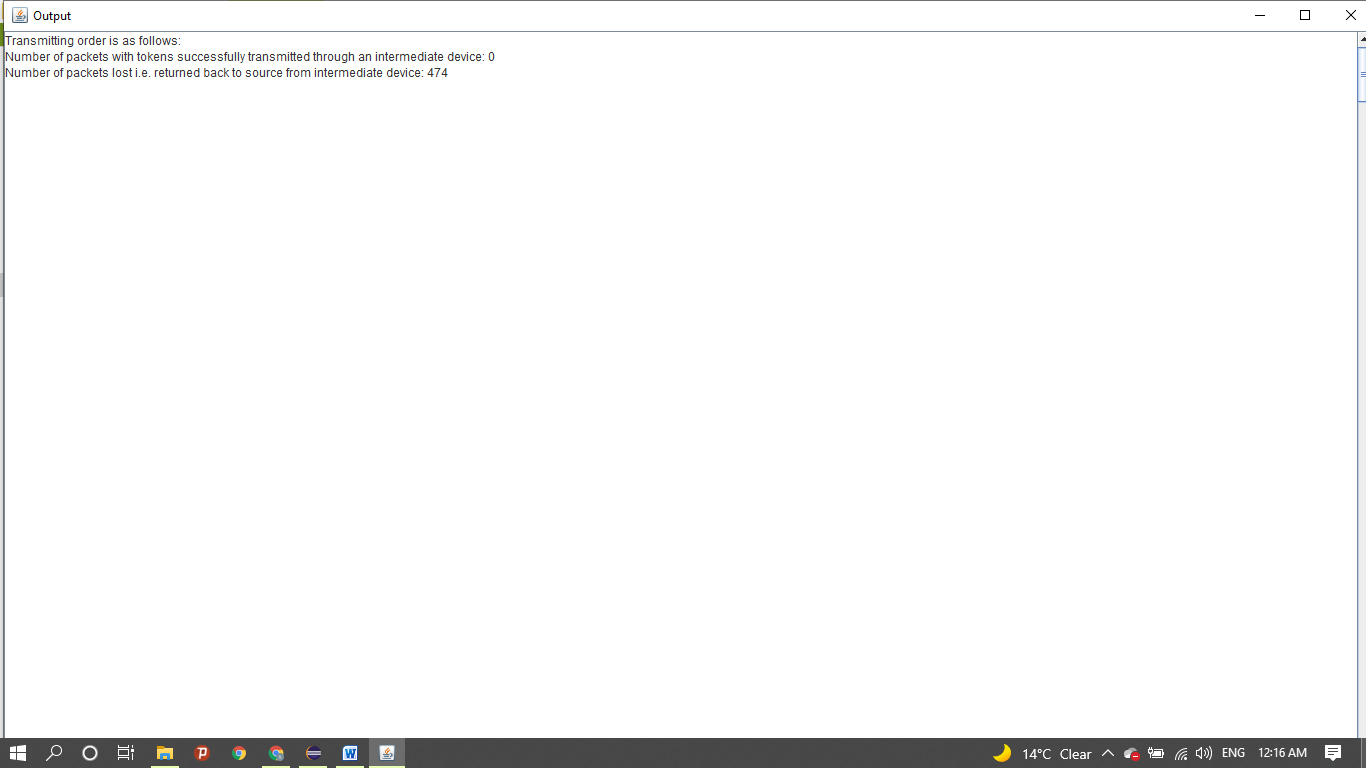
**Note:** GUI has been implemented using Java Swing.

1. **Test Cases that have been run**

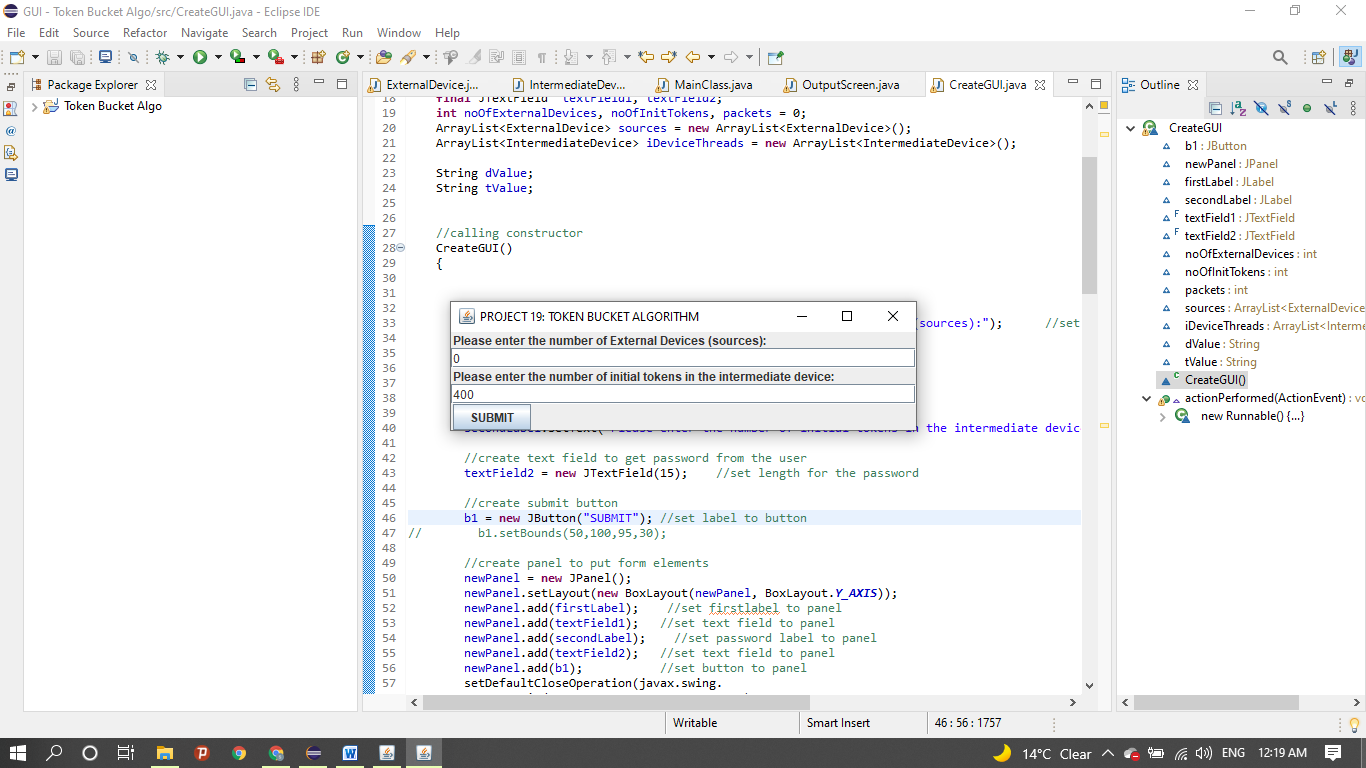
The following values have been used to obtain the outputs

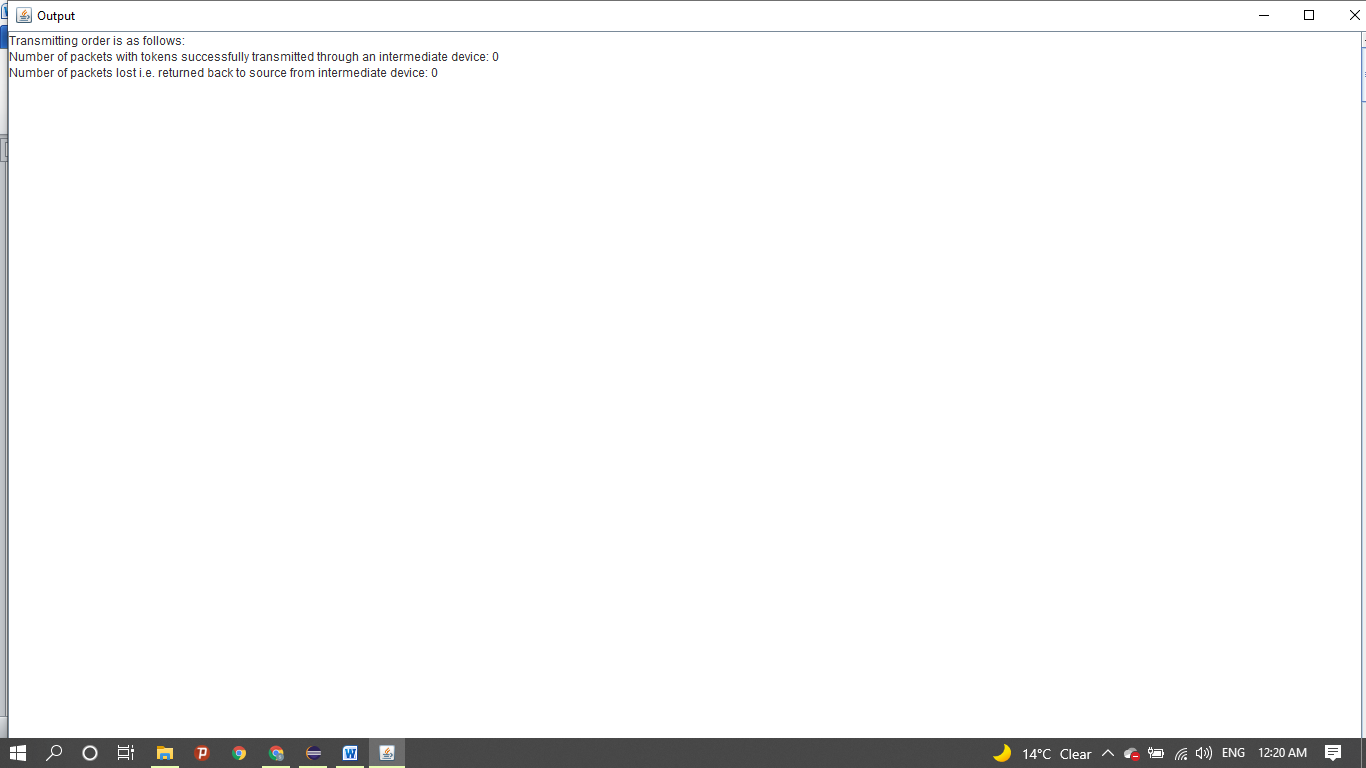
* The **memory** attribute i.e. the number of packets that can be stored in the intermediate device = **1000**
* The **capacity** attribute i.e. the number of tokens that can be stored in the intermediate device = **500**
* The **rate** attribute i.e. the rate at which tokens are replenished in the token bucket = **250**
* The **number of packets** in each external device/source has been randomized to a value between **1 and 10 (both inclusive)**
* **Case I: number of initial tokens = 0**



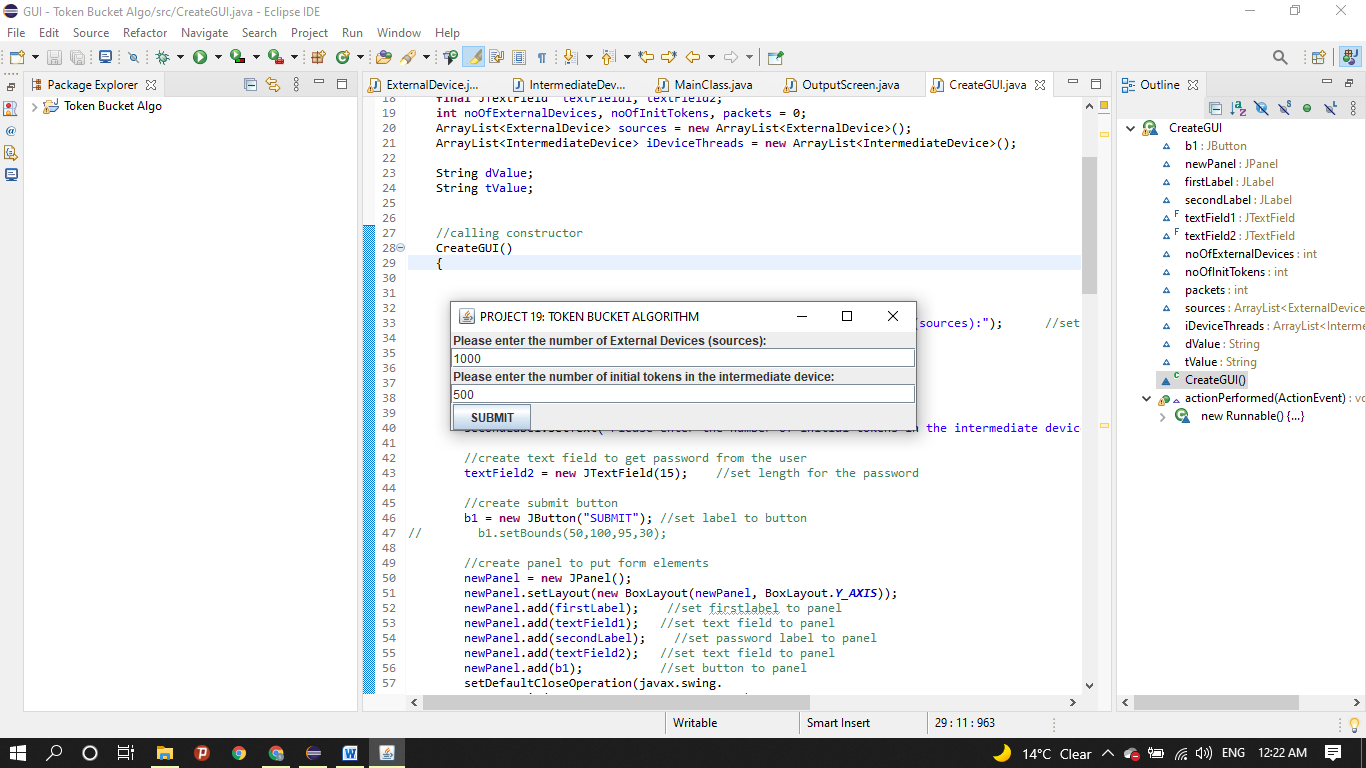


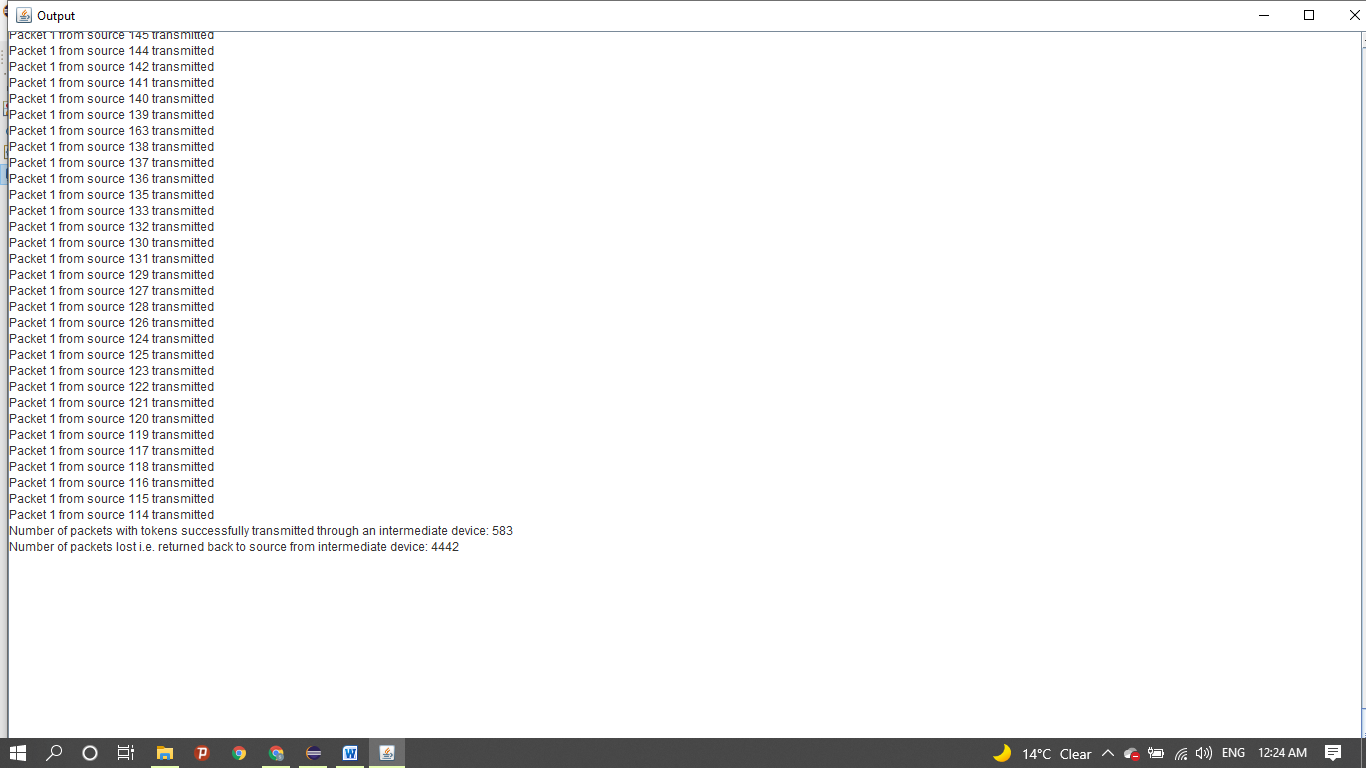
* **Case II: number of external devices = 0**



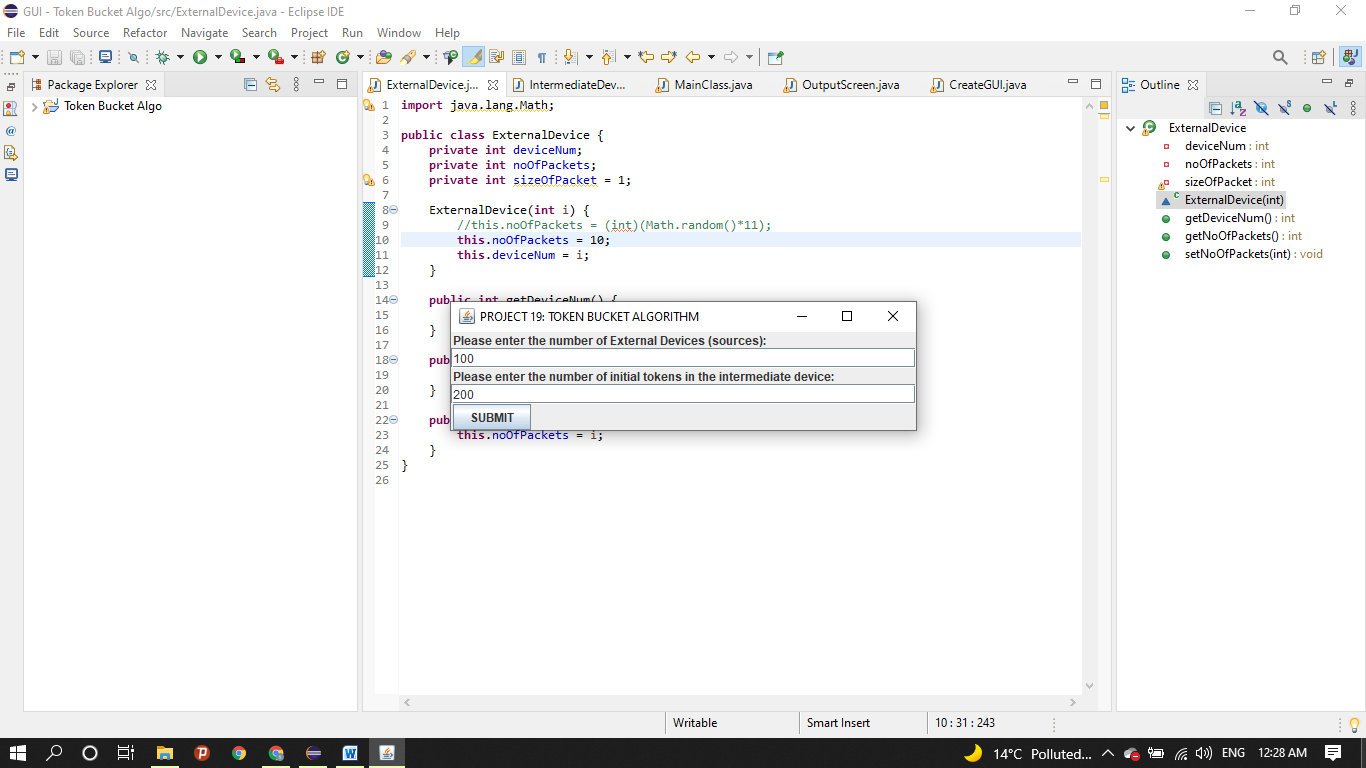


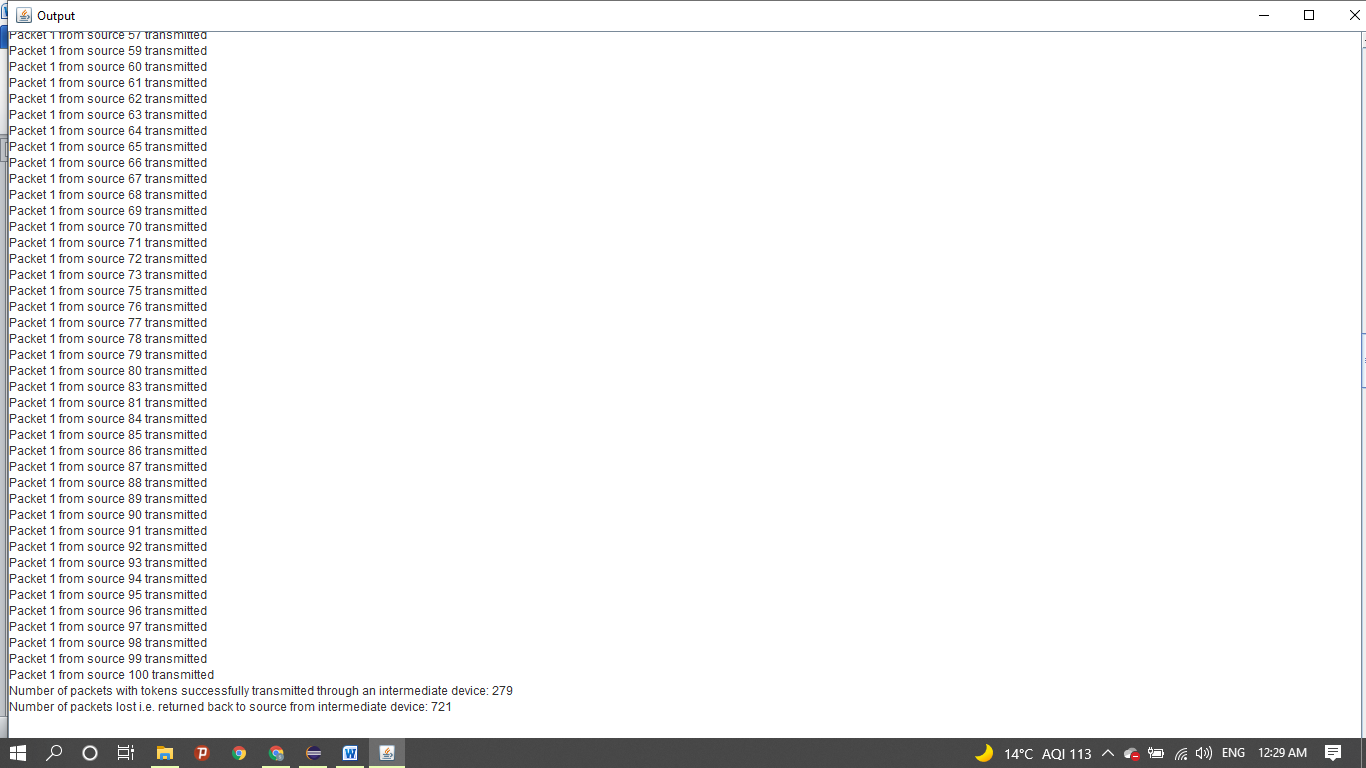
* **Case III: the number of initial tokens = capacity = 500**





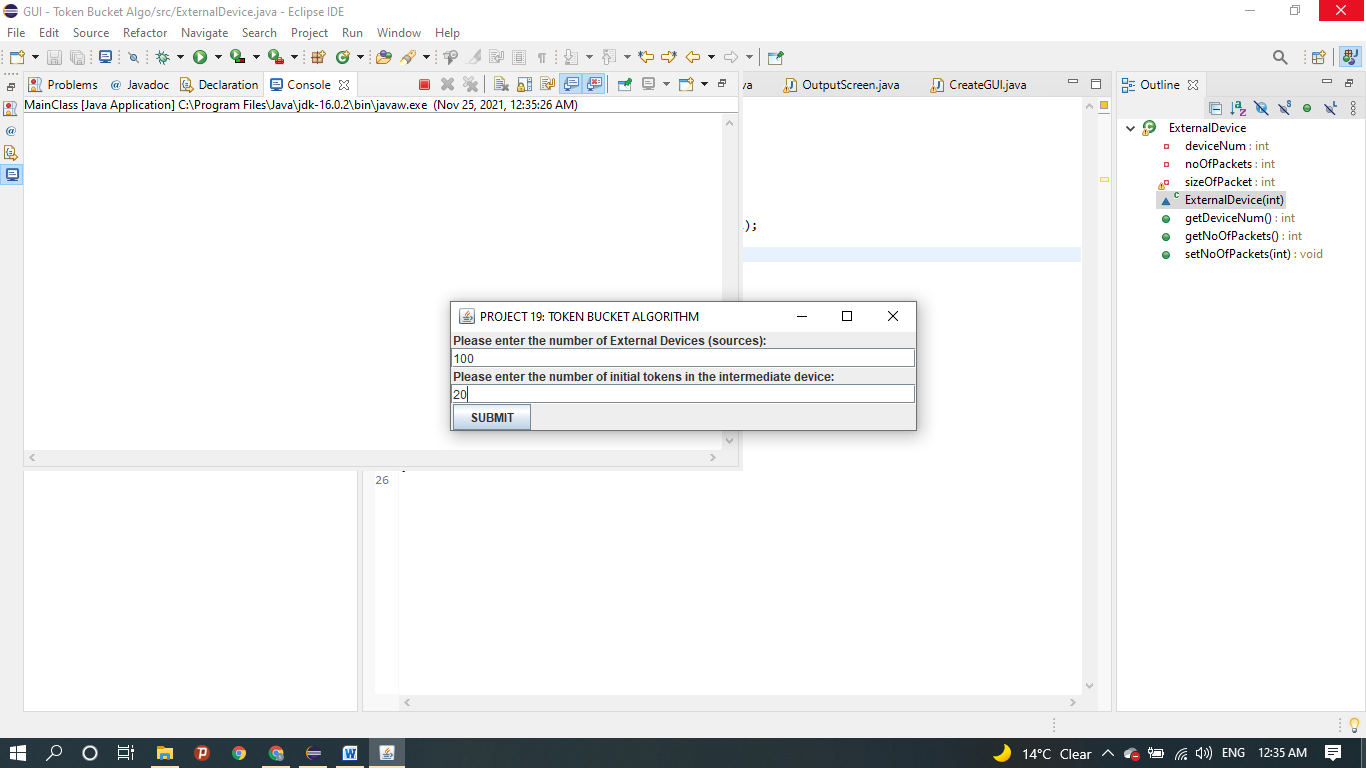
* **Case IV: the number of packets in each external device is set to 10 rather than randomizing them**

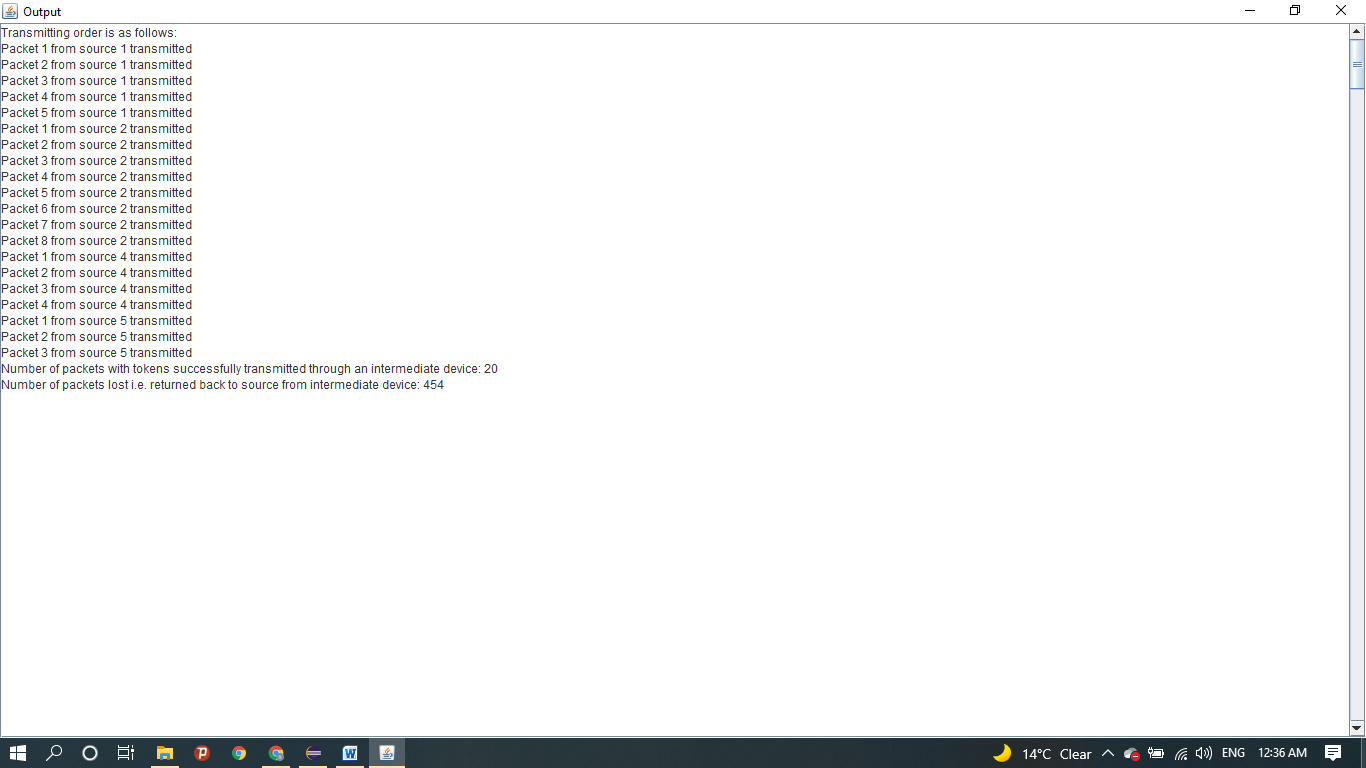




Clearly we can observe that in this case the number of lost packets + number of packets successfully transmitted = 721 + 279 = 1000. Which is the total number of packets that went into the program since number of packets per external device instance is 10 and the total number of external devices are 100.

**Case V: Random implementation**





1. **Additional Information**

* Note: The project was developed using the eclipse IDE Version: 2021-06 (4.20.0) and the src folder had JRE JavaSE 16 as its runtime environment. The project has been tested both in the IDE with the above specified JRE and in Windows 10 Pro using the cmd through javac and java commands.
* Double clicking on 19\_2019B4A70627P\_2019B2A70878.jar file will execute the program if the device has java installed.

Otherwise we can keep all java files included in 19\_2019B4A70627P\_2019B2A70878.jar file or 19\_2019B4A70627P\_2019B2A70878 folder, in a folder and through cmd type the following commands after reaching the folder:

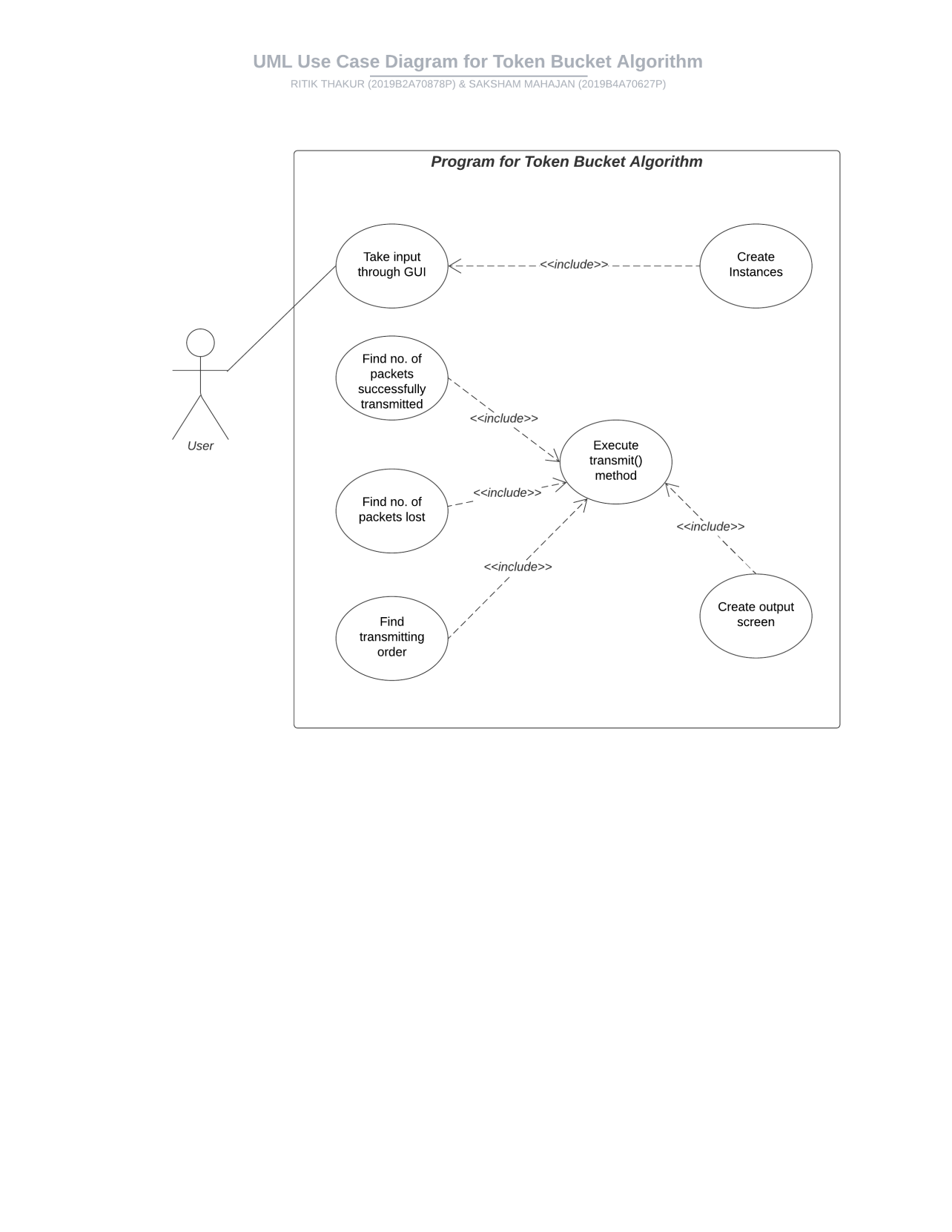
javac MainClass.java

java MainClass

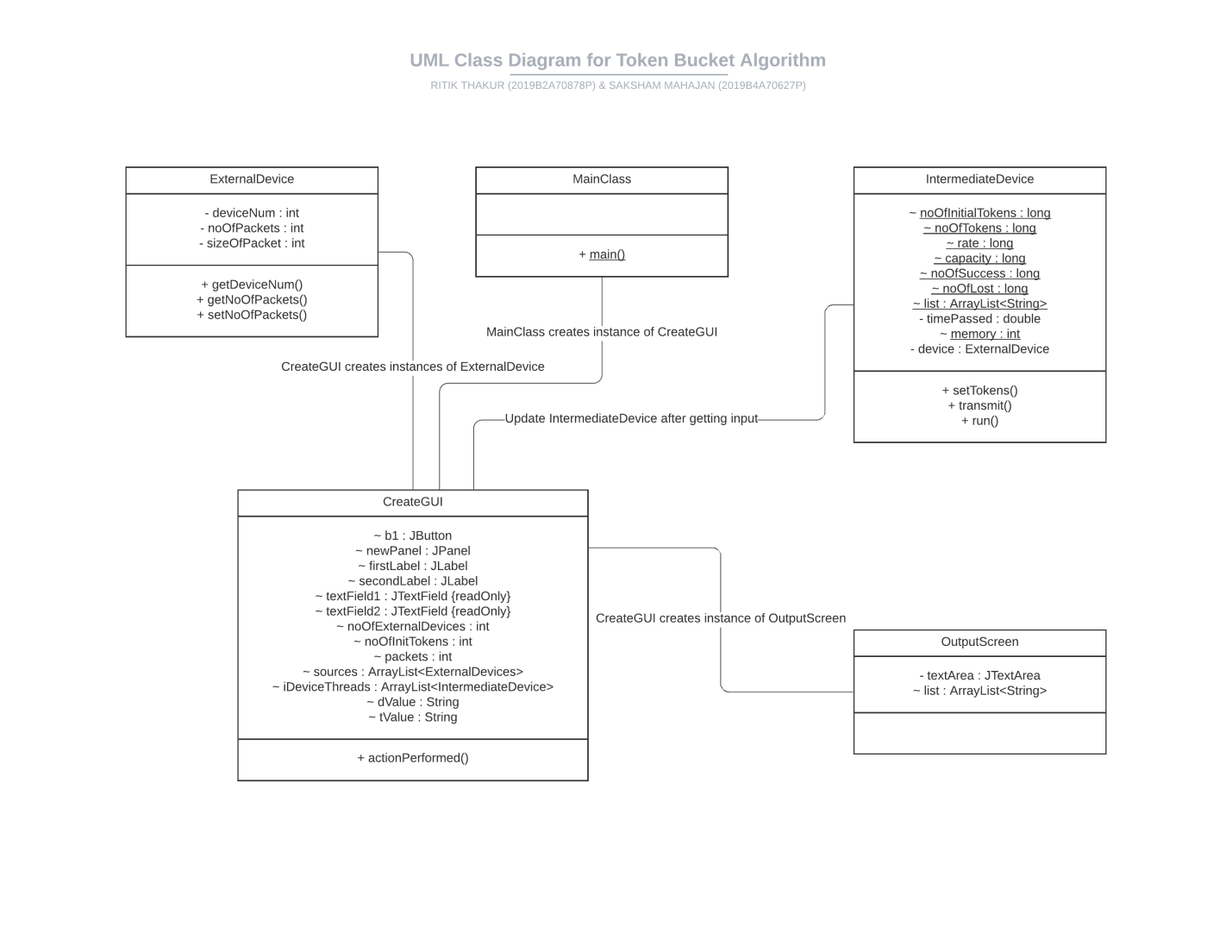
Else we can use an IDE and simply run the java application through 19\_2019B4A70627P\_2019B2A70878 folder.

* The GUI has been implemented using Java Swing and addition of extra libraries and inheriting classes like JFrame. These additional libraries were also available by default so no other installation or software has been used as such.
* Future work possible: The program can be extended further by taking the memory, capacity and rate as inputs from the user. Also we can further create varied instances of IntermediateDevice class and ExternalDevice as per the requirement and need. We can also take a string of inputs from a user to initialize the number of packets per source.
* Known Limitations: If submit button is pressed without typing the required inputs the program throws an error and terminates. There can be a lag between the closing of the input screen (input GUI) window and opening of the output screen, this time is the processing time. The working is dependent on JRE and can misbehave if JRE is changed.

1. **UML Diagrams**

****

UML USE CASE DIAGRAM

****

UML CLASS DIAGRAM