

# Practica 2: Implementación de un Token-ring

Alumno: Alejandro Franco Mireles

Grupo: 4CV14

Materia: Desarrollo de Sistemas Distribuidos

Profesor: Pineda Guerrero Carlos

Sep 2021

## 1. Compilación

Imágenes de la compilación

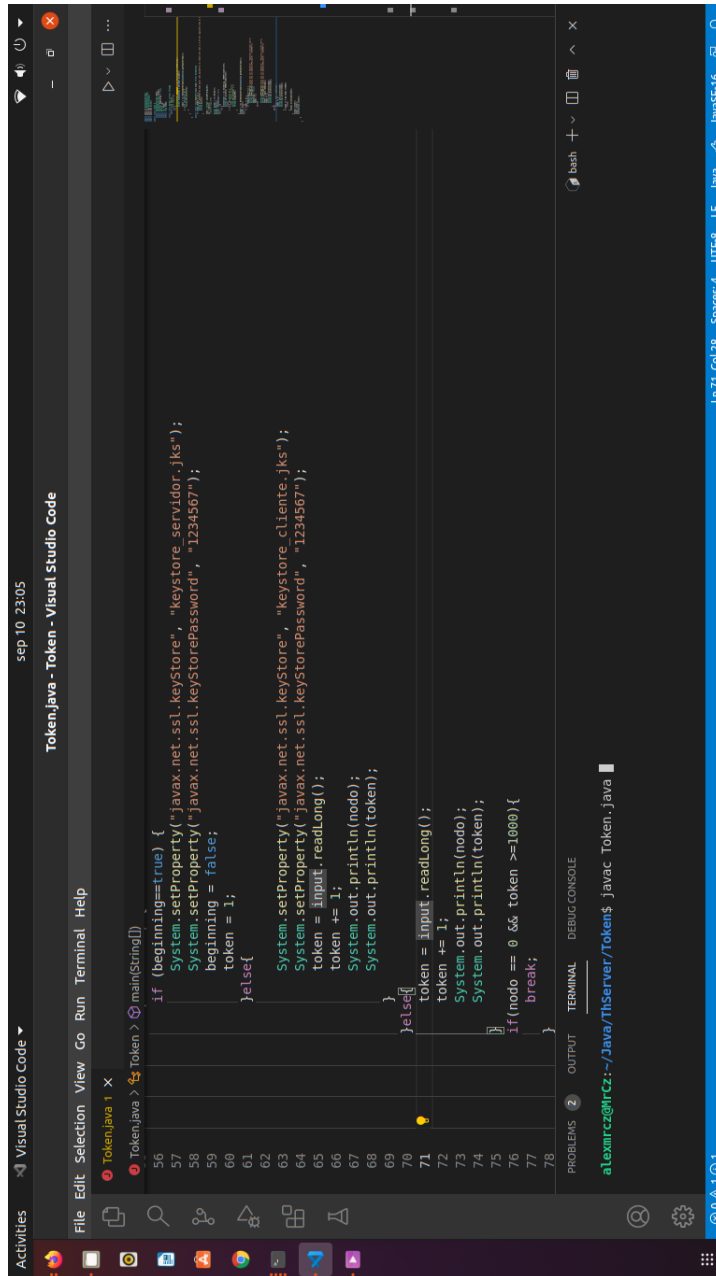


Figura 1: Antes de la compilación

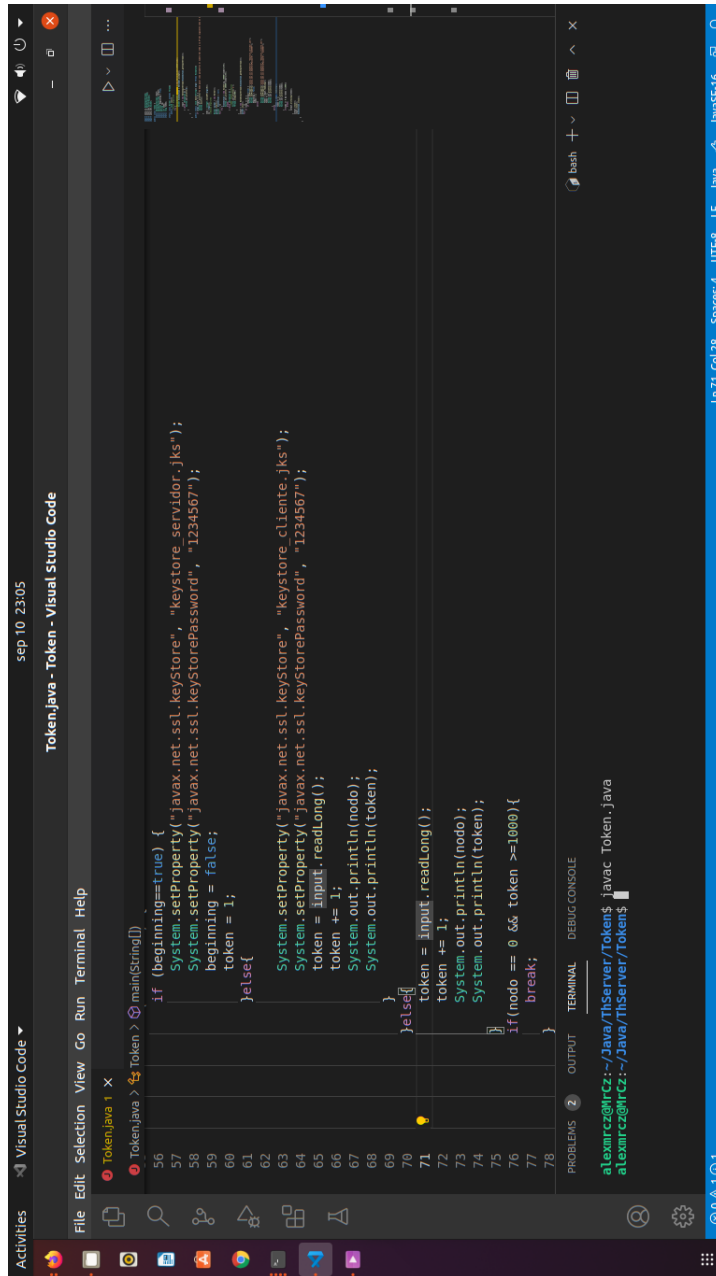
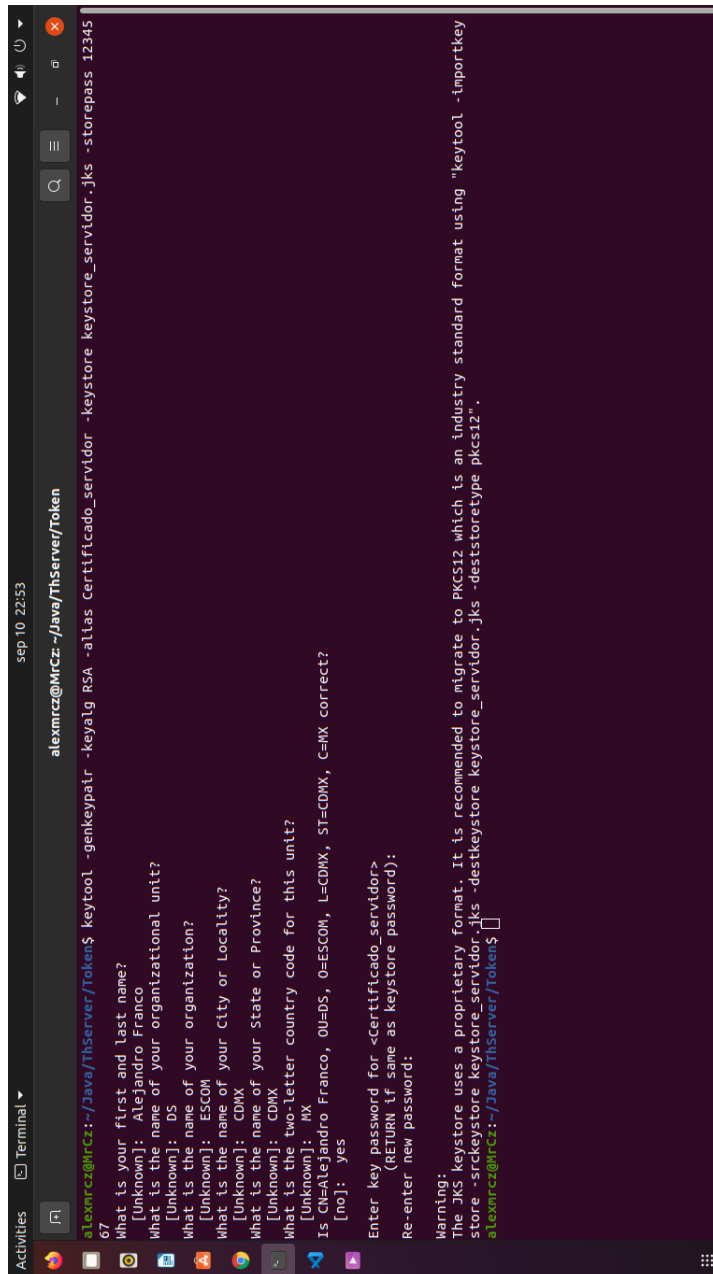


Figura 2: Después de la compilación

## 2. Implementación

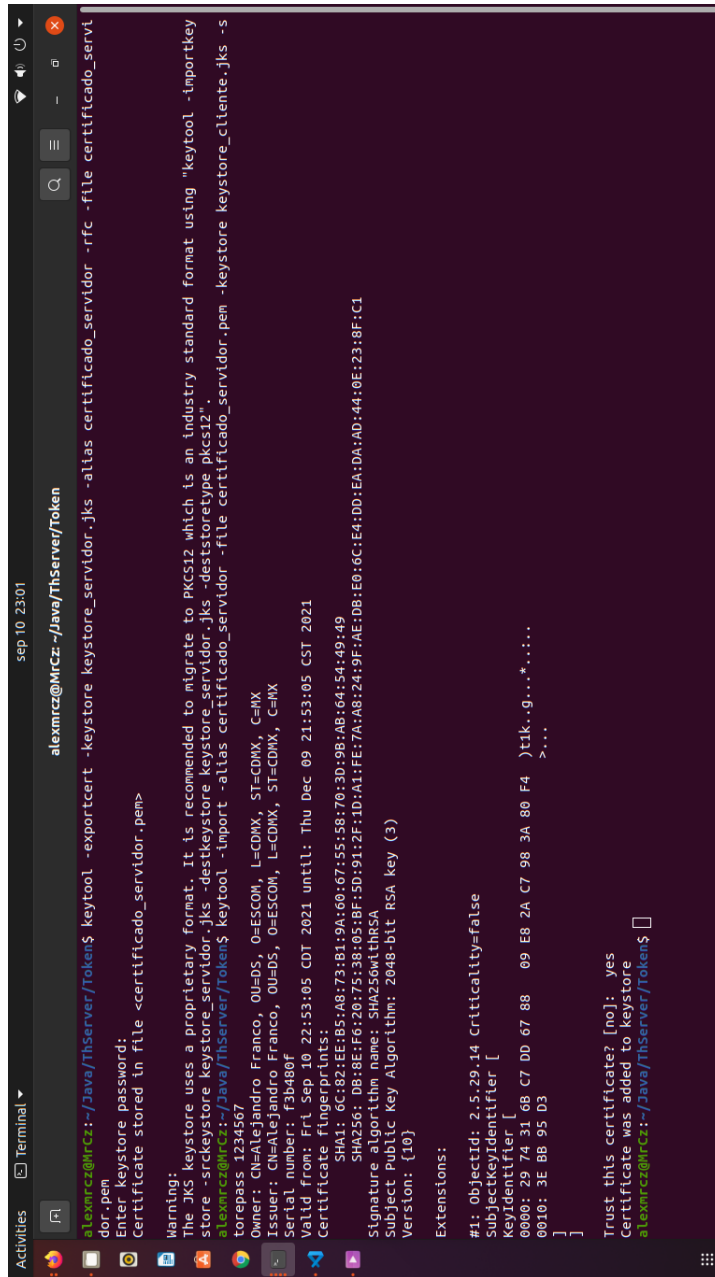
Generando Key Servidor



```
alexmrz@MrCz: ~/Java/ThisServer/Token
alexmrz@MrCz: ~/Java/ThisServer/Token$ keytool -genkeypair -keyalg RSA -alias Certificado_servidor -keystore keystore_servidor.jks -storepass 12345
67
What is your first and last name?
[Unknown]: Alejandro Franco
What is the name of your organizational unit?
[Unknown]: DS
What is the name of your organization?
[Unknown]: ESCOM
What is the name of your City or Locality?
[Unknown]: CDMX
What is the name of your State or Province?
[Unknown]: CDMX
What is the two-letter country code for this unit?
[Unknown]: MX
Is CN=Alejandro Franco, OU=DS, O=ESCOM, L=CDMX, ST=CDMX, C=MX correct?
[no]: yes
Enter key password for <Certificado_servidor>
(RETURN if same as keystore password):
Re-enter new password:
Warning:
The JKS keystore uses a proprietary format. It is recommended to migrate to PKCS12 which is an industry standard format using 'keytool -importkey
store -srckeystore keystore_servidor.jks -destkeystore keystore_servidor.jks -deststoretype pkcs12'.
alexmrz@MrCz: ~/Java/ThisServer/Token$
```

Figura 3: Servidor

## Generando Key Cliente



```
alexmcz@Mrcz: ~/Java/ThServer/Token
alexmcz@Mrcz:~/Java/ThServer/Token$ keytool -exportcert -alias certificado_servidor -rfc -file certificado_servi
dor.pem
Enter keystore password:
Certificate stored in file <certificado_servidor.pem>

Warning:
The JKS keystore uses a proprietary format. It is recommended to migrate to PKCS12 which is an industry standard format using "keytool -importkey
store -srckeystore keystore_servidor.jks -destkeystore keystore_servidor.jks -deststoretype pkcs12".
alexmcz@Mrcz:~/Java/ThServer/Token$ keytool -import -alias certificado_servidor -file certificado_servidor.pem -keystore keystore_cliente.jks -s
torepass 1234567
Owner: CN=Alejandro Franco, OU=DS, O=ESCOM, L=CDMX, ST=CDMX, C=MX
Issuer: CN=Alejandro Franco, OU=DS, O=ESCOM, L=CDMX, ST=CDMX, C=MX
Serial number: f3b48ef
Valid from: Fri Sep 10 22:53:05 CDT 2021 until: Thu Dec 09 21:53:05 CST 2021
Certificate fingerprints:
  SHA1: 0C:82:EE:B5:A8:73:B1:9A:60:67:55:58:70:3D:9B:AB:64:54:49:49
  SHA256: DB:8E:F6:20:75:38:05:BF:3D:91:2F:1D:1A:1F:FE:7A:AB:24:9F:AE:DB:E0:6C:E4:DD:EA:DA:AD:44:0E:23:8F:C1
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit RSA key (3)
Version: (10)

Extensions:
#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 29 74 31 68 C7 DD 67 88 09 E8 2A C7 98 3A 80 F4 )tk.g...*....
0010: 3E BB 95 D3 >...
]
Trust this certificate? [no]: yes
Certificate was added to keystore
alexmcz@Mrcz:~/Java/ThServer/Token$
```

Figura 4: Cliente

## Ejecución nodo 0

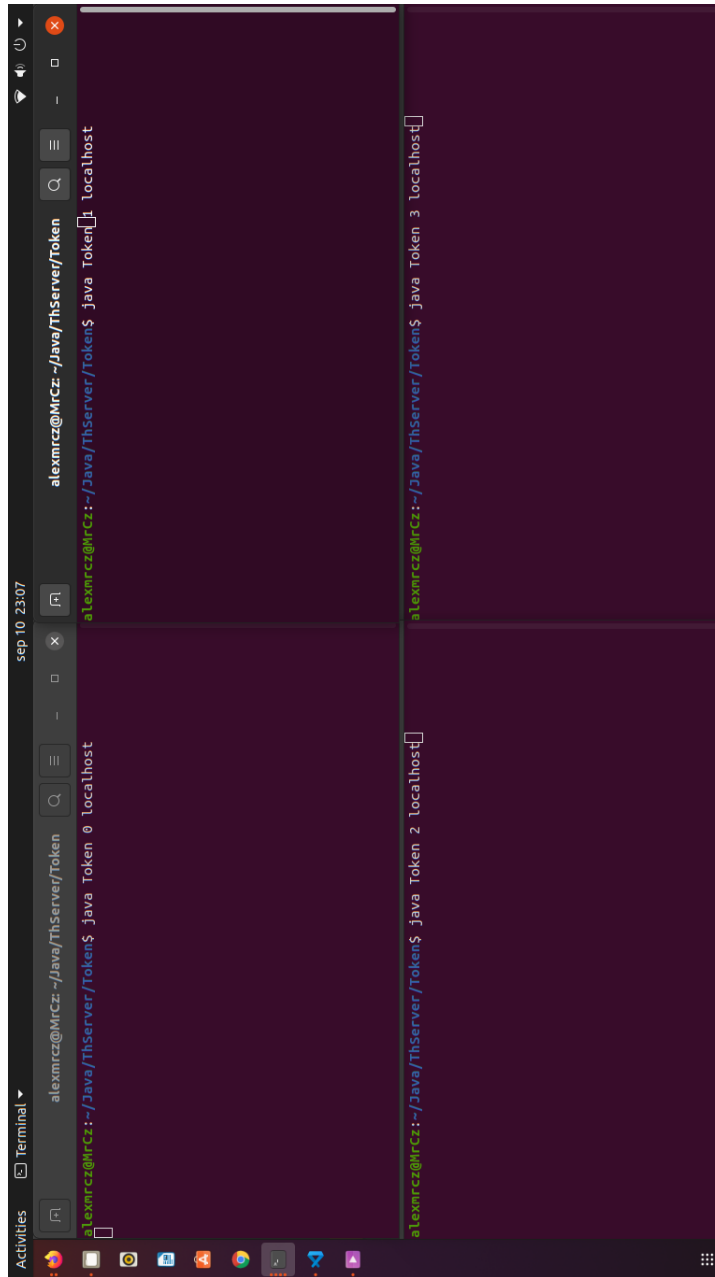


Figura 5: nodo 0

## Ejecución nodo 1

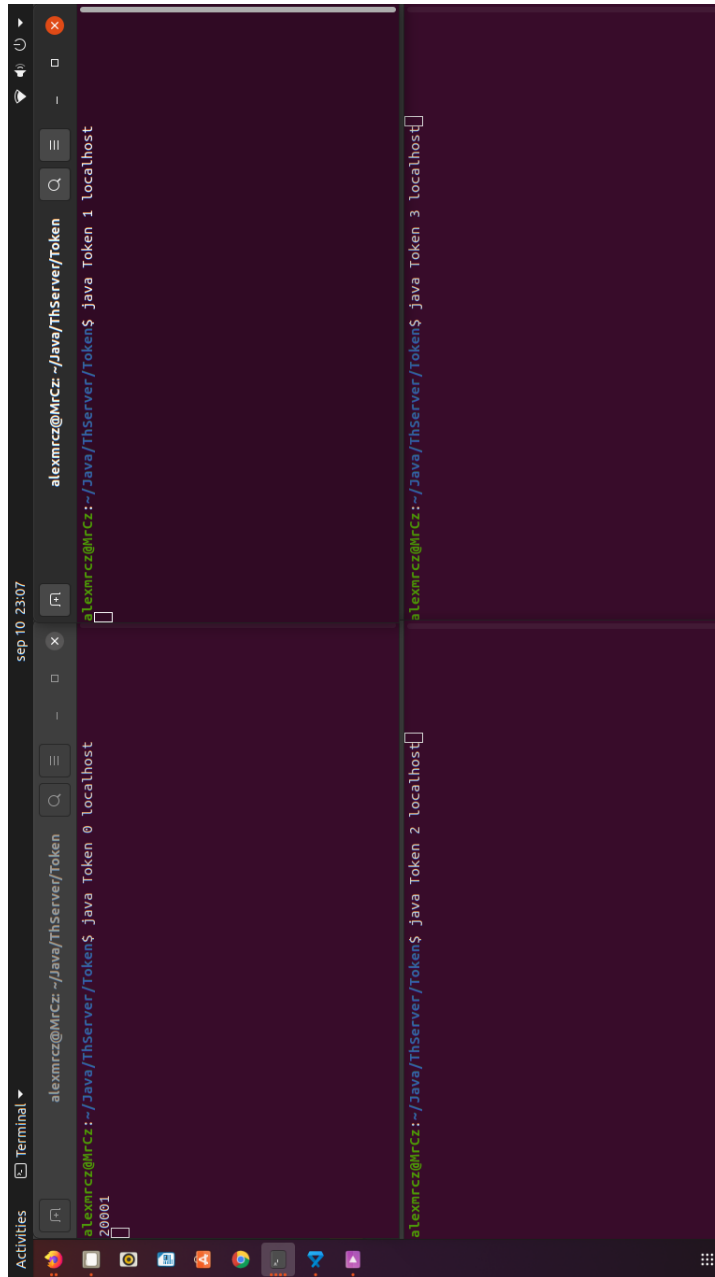


Figura 6: nodo 1

## Ejecución nodo 2

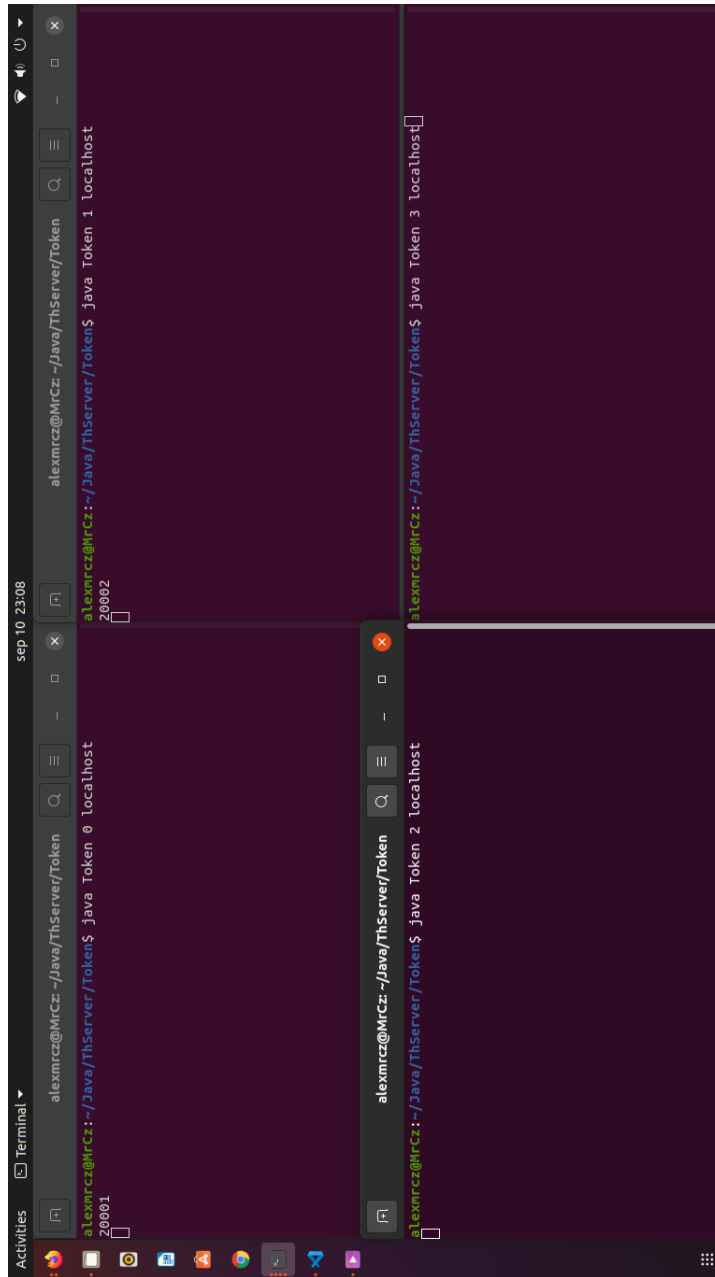
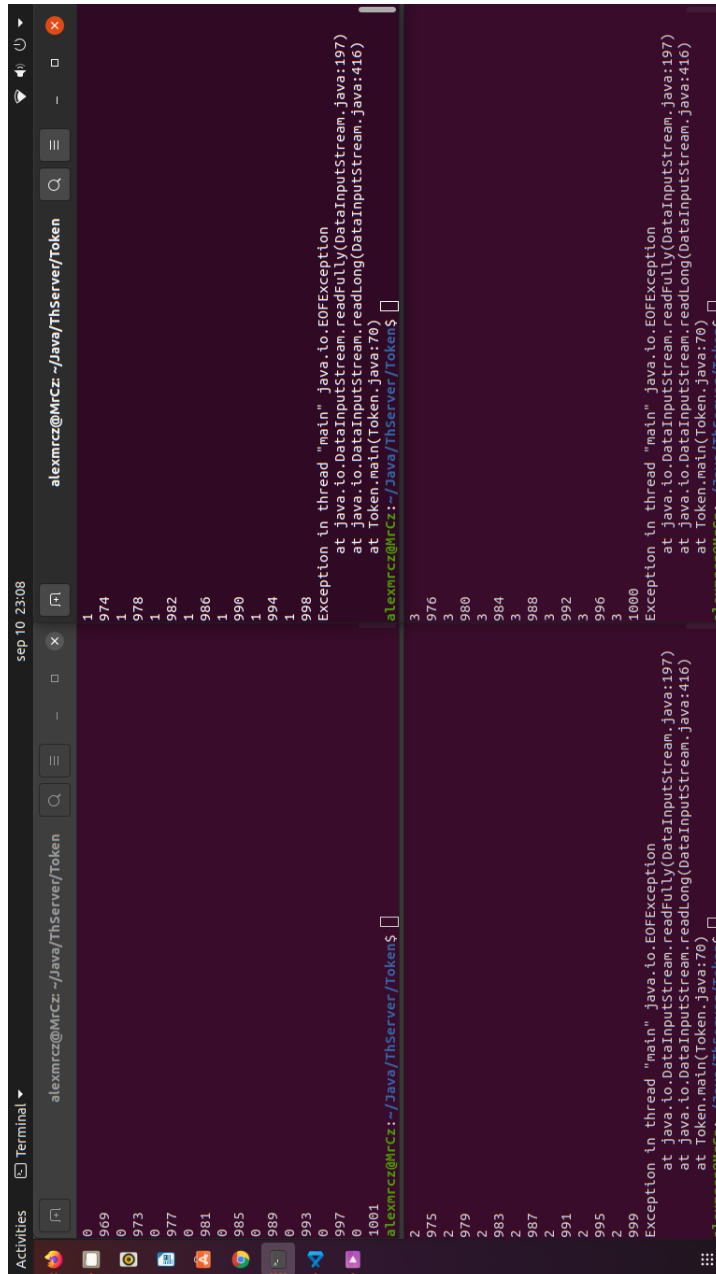


Figura 7: nodo 2



## Ejecución nodo 3 y resultados



```
alexmrz@MrCz: ~/Java/ThServer/Token
1
974
1
978
1
982
1
986
1
990
1
994
1
998
Exception in thread "main" java.io.EOFException
at java.io.DataInputStream.readFully(DataInputStream.java:197)
at java.io.DataInputStream.readLong(DataInputStream.java:416)
at Token.main(Token.java:70)
alexmrz@MrCz: ~/Java/ThServer/Token$
alexmrz@MrCz: ~/Java/ThServer/Token$
0
969
0
973
0
977
0
981
0
985
0
989
0
993
0
997
1001
alexmrz@MrCz: ~/Java/ThServer/Token$
2
975
2
979
2
983
2
987
2
991
2
995
2
999
Exception in thread "main" java.io.EOFException
at java.io.DataInputStream.readFully(DataInputStream.java:197)
at java.io.DataInputStream.readLong(DataInputStream.java:416)
at Token.main(Token.java:70)
alexmrz@MrCz: ~/Java/ThServer/Token$
3
976
3
980
3
984
3
988
3
992
3
996
3
1000
Exception in thread "main" java.io.EOFException
at java.io.DataInputStream.readFully(DataInputStream.java:197)
at java.io.DataInputStream.readLong(DataInputStream.java:416)
at Token.main(Token.java:70)
alexmrz@MrCz: ~/Java/ThServer/Token$
```

Figura 8: nodo 3 y resultados

### **3. Conclusiones**

Se puede observar que al momento de ejecutar de manera independiente sin los otros nodos, este se queda esperando la conexión y ya que esta se establezca se genera el puerto destino. Se tiene que tener cuidado donde se va a poner cada una de las sentencias ya que puede entrar en error si una de ellas queda dentro de un ciclo y esta puede ocasionar un mal funcionamiento de nuestro programa. Como última observación se puede deducir que el programa no se ejecuta hasta que todos los nodos estén ejecutandose.