COMP 6231
Distributed System Design
Project

By Vasu Ratanpara (Student ID: 40135264)

❖ Distributed System Architecture

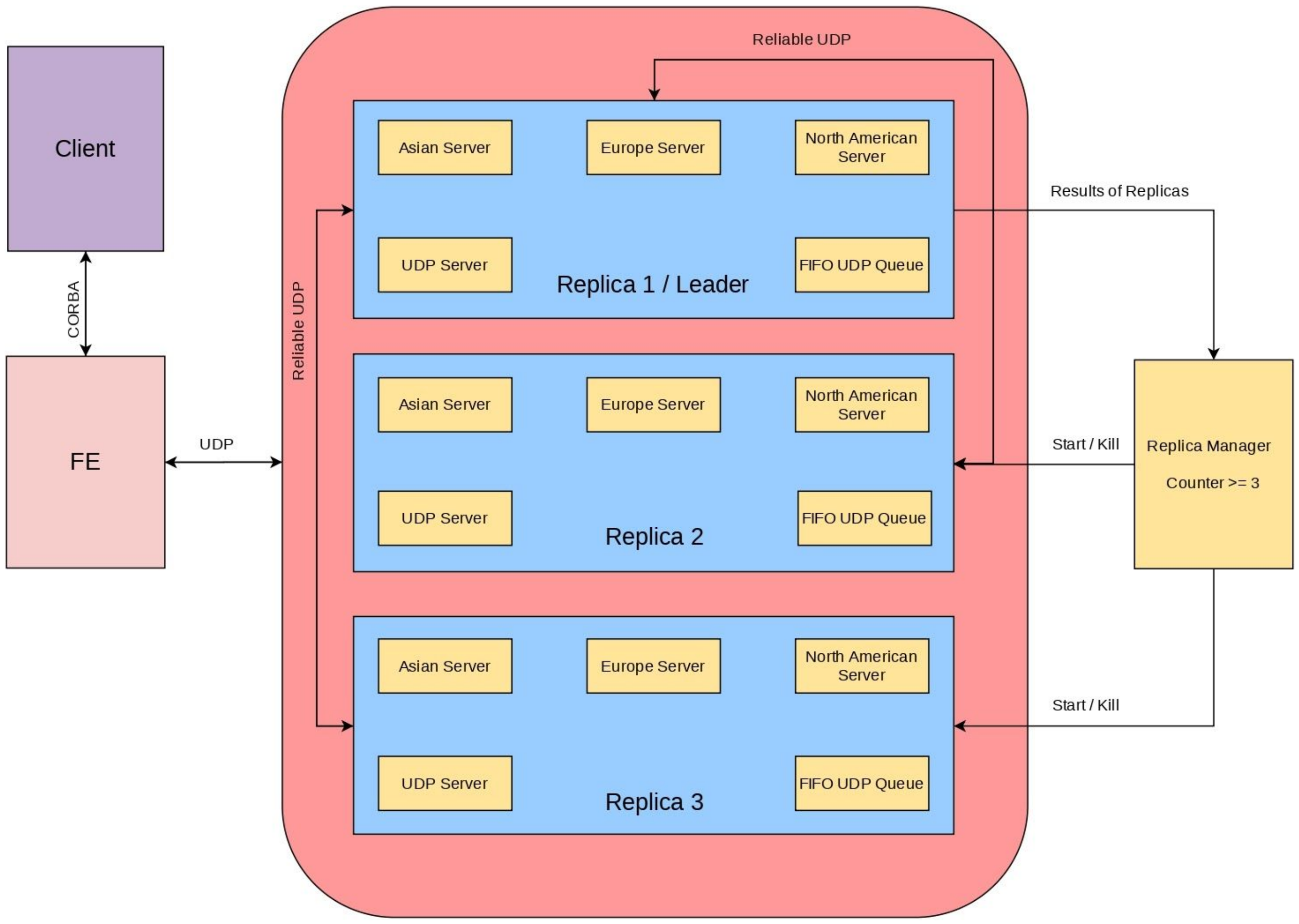


Figure : Architecture of a Distributed System

As you can see in the above figure, I tried to present a pictorial representation of the Project architecture. Here, I have 3 Replicas namely, Replica 1 (Leader), Replica 2, and Replica 3. Also, I have one Front-end and Replica manager. The Front-end is connected with clients by CORBA connection which is described further. The Front-end is connected with the leader replica with UDP tunnel which will be used by Front-end to send client's requests to the leader replica.

The leader replica sends the client's request to two other follower replicas via unicast UDP FIFO queue. Here, in real life situations we should use multicast but since we are running our project on the same system (localhost) so we cannot use multicast hence I have used the unicast which was suggested by the teaching assistant during the lab.

All Replicas will filter the requests by the IP address and send them to their internal servers (i.e. asian server, european server and north american server). This communication is handled by UDP as well. The internal servers will send back the result to parent replicas and the parent replicas will send the same output to the leader replica.

The leader replica will compare all the outputs and send the output of the comparison to RM (Replica manager). The RM will parse the output and if any of the three replicas is producing the wrong output it will increment the count for that replica by one. If any replica will continuously produce the 3 times wrong output the replica manager will restart that replica to fix the Byzantine software failure.

The replica will take a snapshot of its own database before killing itself so that when it will restart it can load the previous data from the database. I have used java's serialization for taking the snapshot of the database.

After the restarting replica the Byzantine software will be fixed.

For the Client program, the player can send a request to create an account, sign in, sign out, and transfer an account to another server. It will initially go to the Front-end. Moreover, when a player will request a transfer of an account to different servers during that time the player's server will communicate with the receiving server via UDP and check if the account with the given name already exists or not. The transfer account operation is an atomic operation. I have described it in-depth below in the report.

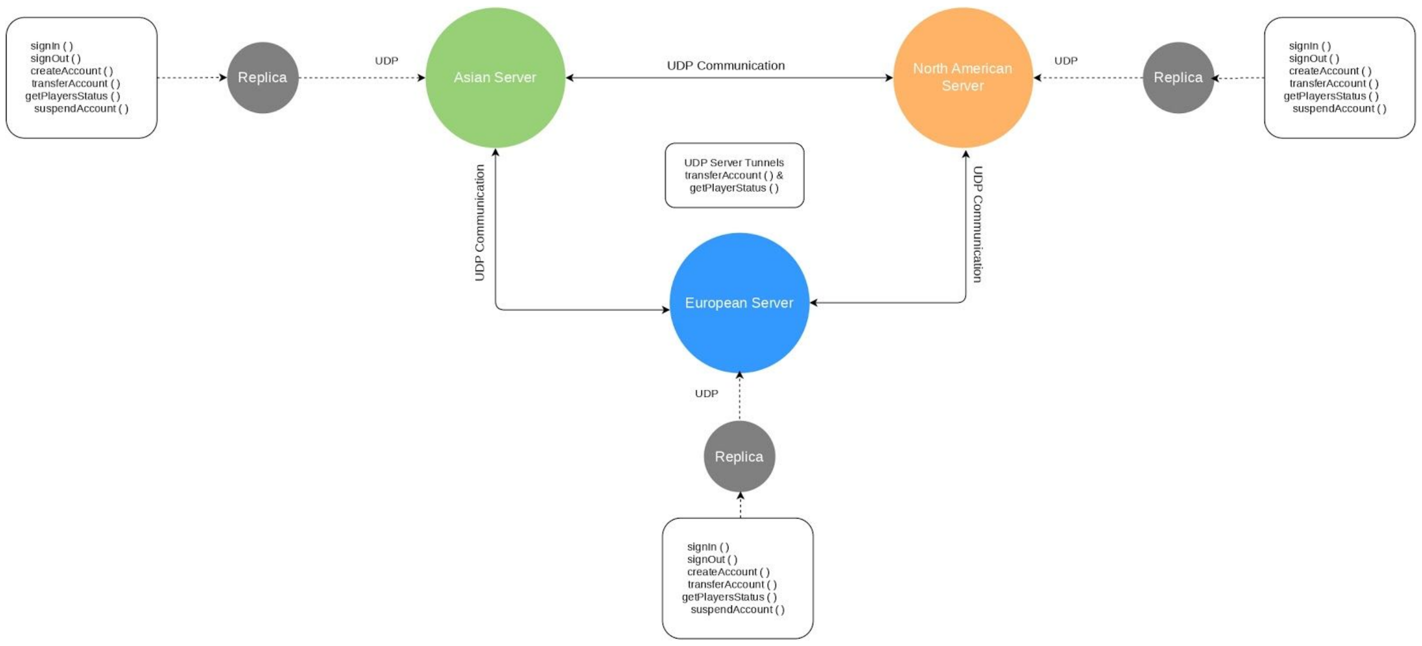


Figure : Internal Architecture of a Replica

In addition, when the player will send a request for any operation it will redirect to the appropriate server as per their IP address. In the case of Admin, When Admin wants to know the status of players in the whole system. Admin will send a request to anyone server and that server will talk with the other 2 servers and request for their own status and send back to the Admin by combining all status of 3 servers.

❖ Design Of Distributed System

❖ Object Request Broker Architecture

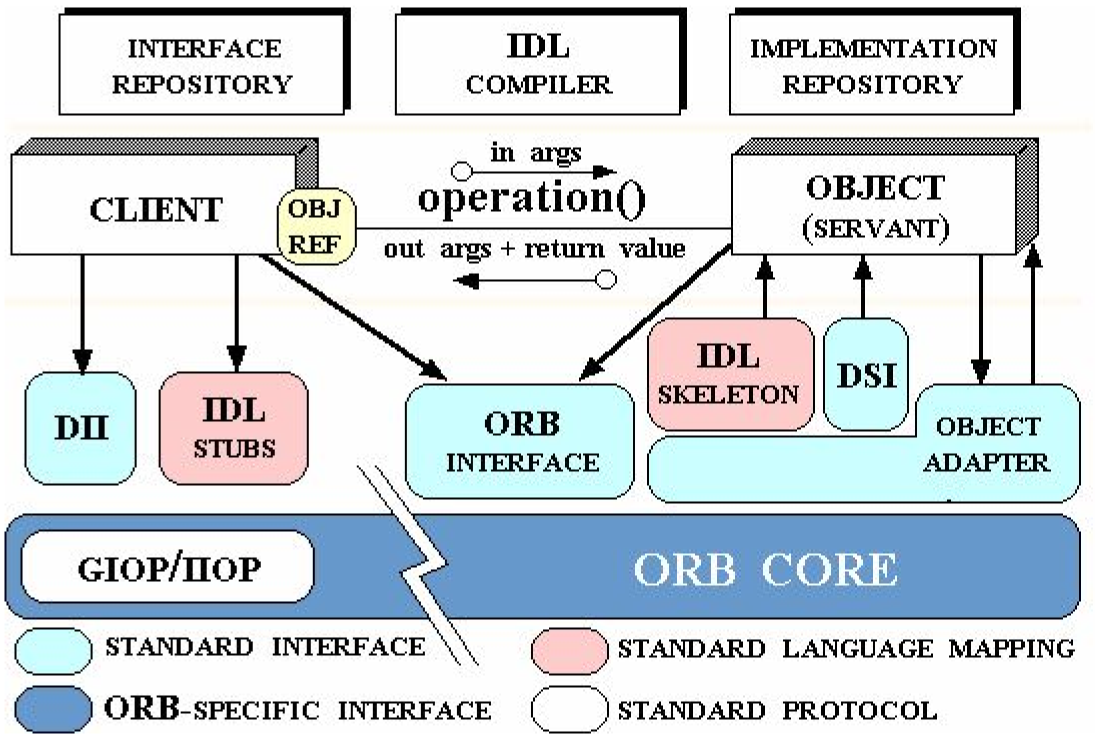


Figure : ORB architecture (Ref. professor's PPT)

CORBA enables communication between software written in different languages and running on different computers. Implementation details from specific operating systems, programming languages, and hardware platforms are all removed in CORBA. CORBA normalizes the method-call semantics between application objects residing either in the same address-space (application) or in remote address-spaces (same host, or remote host on a network).

CORBA uses an interface definition language (IDL) to specify the interfaces that objects present to the outer world. CORBA then specifies a mapping from IDL to a specific implementation language like C++ or Java.

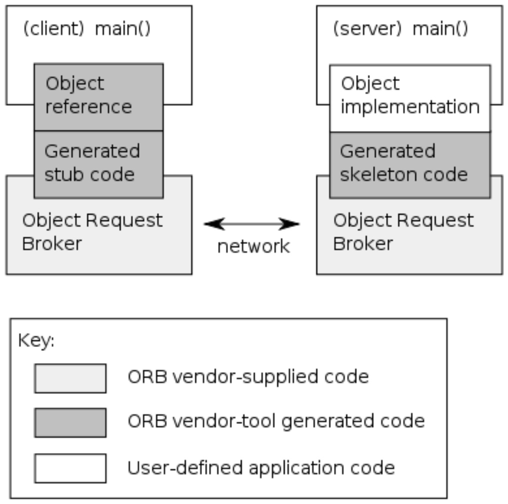


Figure: how the generated code is used within the CORBA infrastructure (Ref. Wikipedia)

❖ Implementation of FIFO ordering algorithm over basic multicast

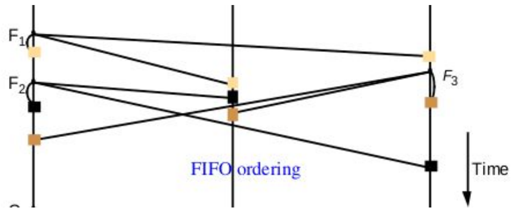


Figure: FIFO algorithm (Ref.Prof.'s PPTs)

FIFO ordered multicast with operations FO-multicast and FO-deliver for non-overlapping groups.It can be implemented on top of any basic multicast. Each process p holds, S a count of messages sent by p to g and R the sequence number of the latest message to g that p delivered from q. For p to FO-multicast a message to g , it piggybacks S pg on the message, B-multicasts it and increments S pg by 1. On receipt of a message from q with sequence number S , p checks whether S = R qg + 1. If so, it FO-delivers it. if S > R qg + 1 then p places a message in the hold-back queue until intervening messages have been delivered.

❖ FIFO ordering and hold-back queue for arriving multicast messages

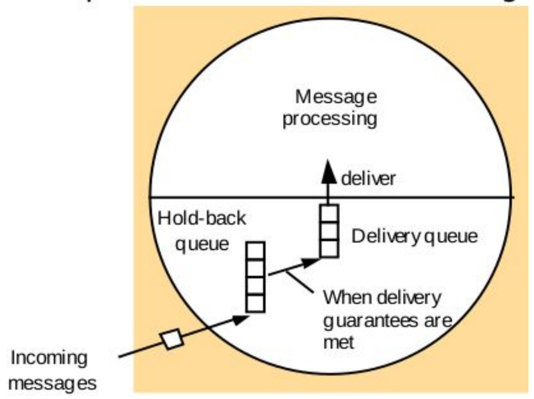


Figure: hold-back queue (Ref.Prof.'s PPTs)

The hold-back queue is not necessary for reliability as in the implementation using IP multicast, but it simplifies the protocol, allowing sequence numbers to represent sets of messages. Hold-back queues are also used for ordering protocols.

❖ including theories (protocol, algorithm) you apply, how to implement.

The Algorithm and theories which are explained above was applied in the development of the project. The UDP FIFO hold-back queue is used inside the replica leader. It holds the request inside the queue and processes it according to FIFO ordering. So, the request which will arrive first will process first.

❖ Distributed System Techniques & Data Structures

```
ConcurrentHashMap<String, ArrayList<HashMap<String, String>>> players
     = new ConcurrentHashMap<String, ArrayList<HashMap<String, String>>>();
```

To store the players' information on the servers, I have used Java's HashMap. Which will store the data in memory somewhat similar as described below?

```
{
    "A": [<Player 1's HashMap>, <Player 2's HashMap>, ......, <Player N's HashMap>],
    "B": [<Player 1's HashMap>, <Player 2's HashMap>, ......, <Player N's HashMap>],
    "C": [<Player 1's HashMap>, <Player 2's HashMap>, ......, <Player N's HashMap>],
    .
    .
    .
    "Z": [<Player 1's HashMap>, <Player 2's HashMap>, ......, <Player N's HashMap>],
}
```

Here, the Keys are the First upper letter of the player's username and it would store an Array list of players' HashMap as a value whose first letter of the username is the same as Key. The structure of players' HashMap is as shown below.

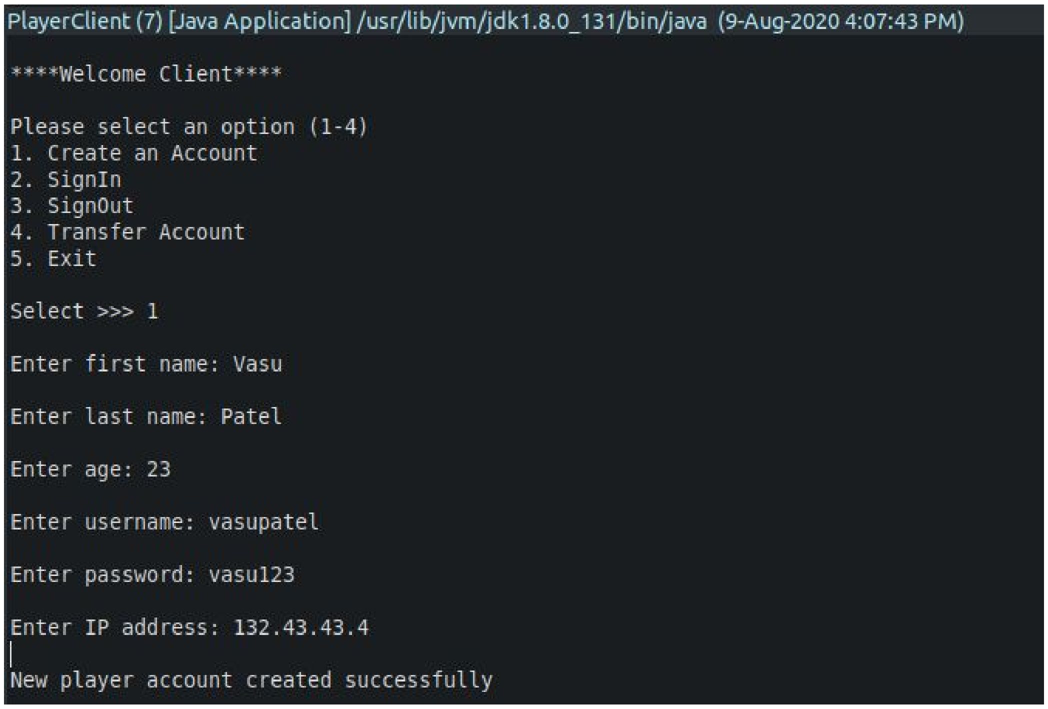
```
{
      “username”: ”Vasu123”,
      “password”: ”Vasu1997”,
      “firstname”: ”Vasu”,
      “lastname”: ”Ratanpara”,
      “age”: 23,
      “ipaddress”: ”182.45.12.3”,
      “status”: ”offline”,
}
```

This player will be stored in the Array list of Key “V” because his username is starting with “V”.

Moreover, I used ConcurrentHashMap<K,V> which is part of Java's in-built package java.util.concurrent. It has full concurrency support and it is a thread-safe data structure so that only one thread can access it at a time.

❖ Test Case Scenarios & Explanation

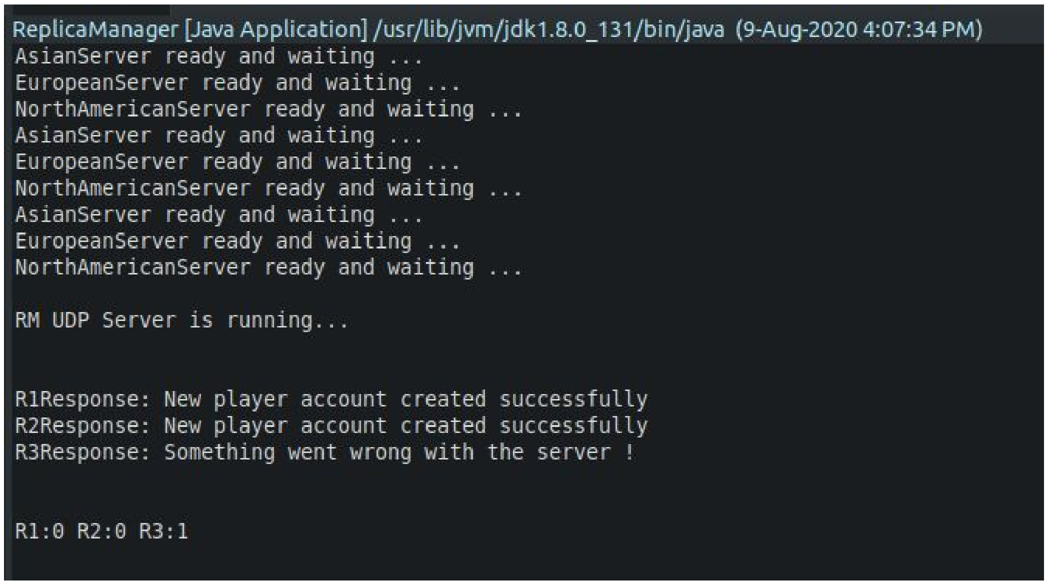
1. Create an account



```
PlayerClient (7) [Java Application] /usr/lib/jvm/jdk1.8.0_131/bin/java (9-Aug-2020 4:07:43 PM)

****Welcome Client****

Please select an option (1-4)
1. Create an Account
2. SignIn
3. SignOut
4. Transfer Account
5. Exit

Select >>> 1

Enter first name: Vasu

Enter last name: Patel

Enter age: 23

Enter username: vasupatel

Enter password: vasu123

Enter IP address: 132.43.43.4

New player account created successfully
```

Figure : Create an account

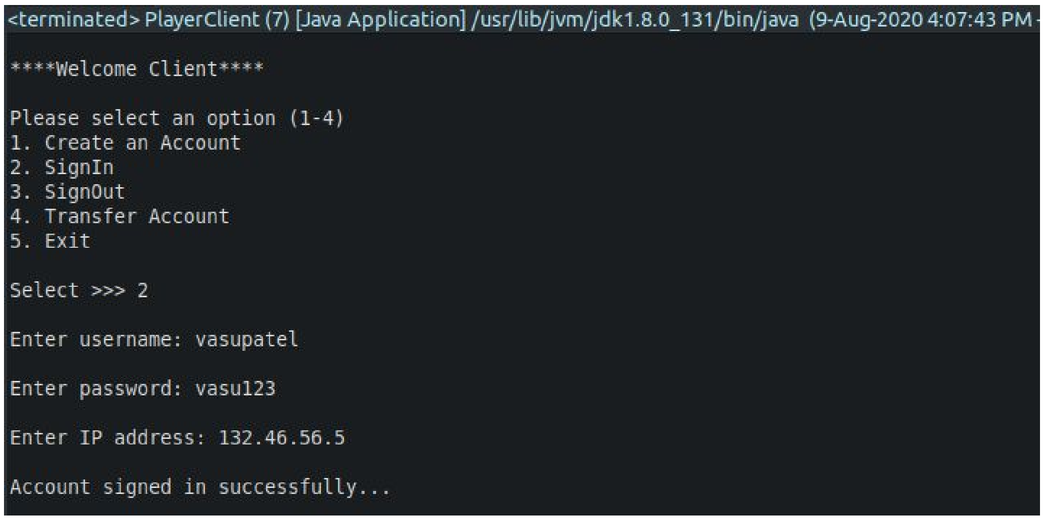
Here as you can see a player can register a new account with the PlayerClient program. All fields are mandatory and the username and password field has length validations which will allow a minimum length of the field as 6 characters. The user account will be stored on the server which is relevant to the given IP address. If the user is already present with the same username then it will give an error that will not create an account. I have also attached the screenshot of the processing that request at the back-end.



```
ReplicaManager [Java Application] /usr/lib/jvm/jdk1.8.0_131/bin/java (9-Aug-2020 4:07:34 PM)
AsianServer ready and waiting ...
EuropeanServer ready and waiting ...
NorthAmericanServer ready and waiting ...
AsianServer ready and waiting ...
EuropeanServer ready and waiting ...
NorthAmericanServer ready and waiting ...
AsianServer ready and waiting ...
EuropeanServer ready and waiting ...
NorthAmericanServer ready and waiting ...

RM UDP Server is running...


R1Response: New player account created successfully
R2Response: New player account created successfully
R3Response: Something went wrong with the server !


R1:0 R2:0 R3:1
```

Figure : Create an account (Back-end)

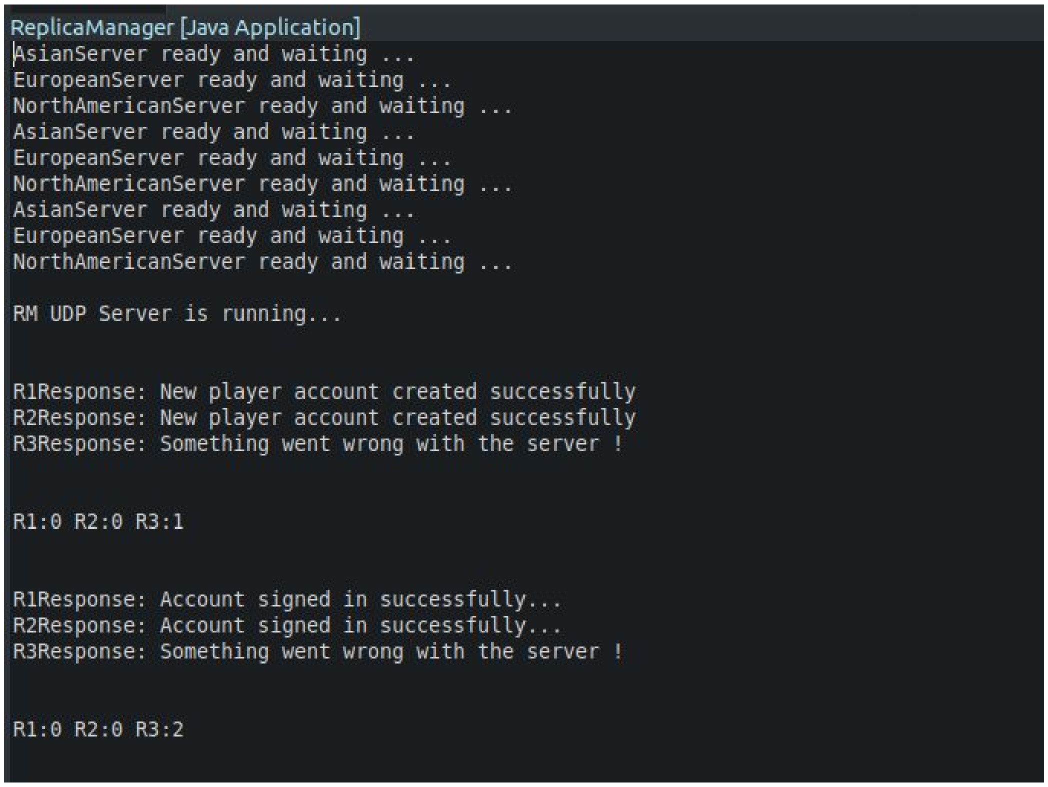
2. Sign in an account



```
<terminated> PlayerClient (7) [Java Application] /usr/lib/jvm/jdk1.8.0_131/bin/java (9-Aug-2020 4:07:43 PM

****Welcome Client****

Please select an option (1-4)
1. Create an Account
2. SignIn
3. SignOut
4. Transfer Account
5. Exit

Select >>> 2

Enter username: vasupatel

Enter password: vasu123

Enter IP address: 132.46.56.5

Account signed in successfully...
```

Figure : Sign in an account

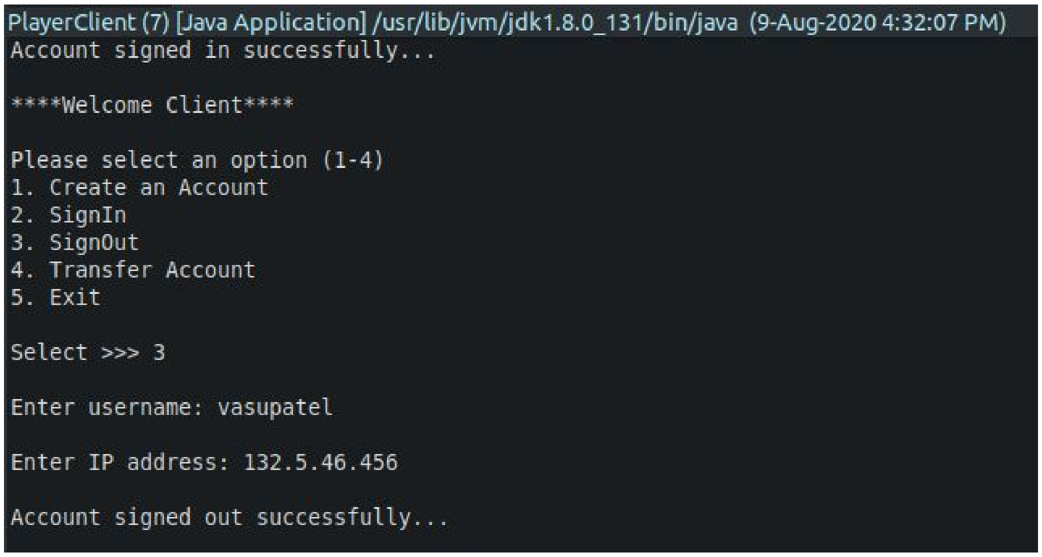
To sign in an account the player needs to provide a username, a password, and the IP address of his/her associated server. For example, the request will be redirected to the European server because the IP address is of Europe I.e. starting with 132.xxx.xxx.xxx. If the password is wrong or the user with the given username doesn’t exist, then the server will discard the request and return an appropriate error. If the username and password are valid then the status of the user will be online in the server.



```
ReplicaManager [Java Application]
AsianServer ready and waiting ...
EuropeanServer ready and waiting ...
NorthAmericanServer ready and waiting ...
AsianServer ready and waiting ...
EuropeanServer ready and waiting ...
NorthAmericanServer ready and waiting ...
AsianServer ready and waiting ...
EuropeanServer ready and waiting ...
NorthAmericanServer ready and waiting ...

RM UDP Server is running...


R1Response: New player account created successfully
R2Response: New player account created successfully
R3Response: Something went wrong with the server !


R1:0 R2:0 R3:1


R1Response: Account signed in successfully...
R2Response: Account signed in successfully...
R3Response: Something went wrong with the server !


R1:0 R2:0 R3:2
```

Figure : Sign in an account (back-end)

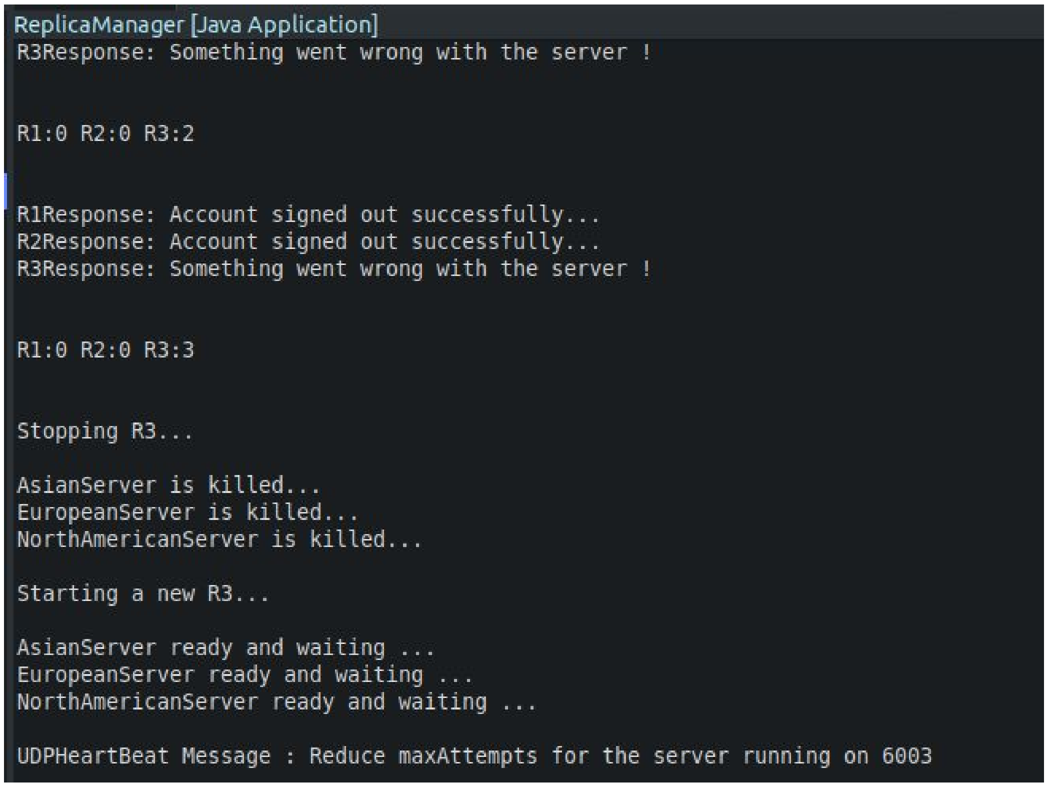
3. Sign out an account



```
PlayerClient (7) [Java Application] /usr/lib/jvm/jdk1.8.0_131/bin/java (9-Aug-2020 4:32:07 PM)
Account signed in successfully...

****Welcome Client****

Please select an option (1-4)
1. Create an Account
2. SignIn
3. SignOut
4. Transfer Account
5. Exit

Select >>> 3

Enter username: vasupatel

Enter IP address: 132.5.46.456

Account signed out successfully...
```

Figure : Sign out an account

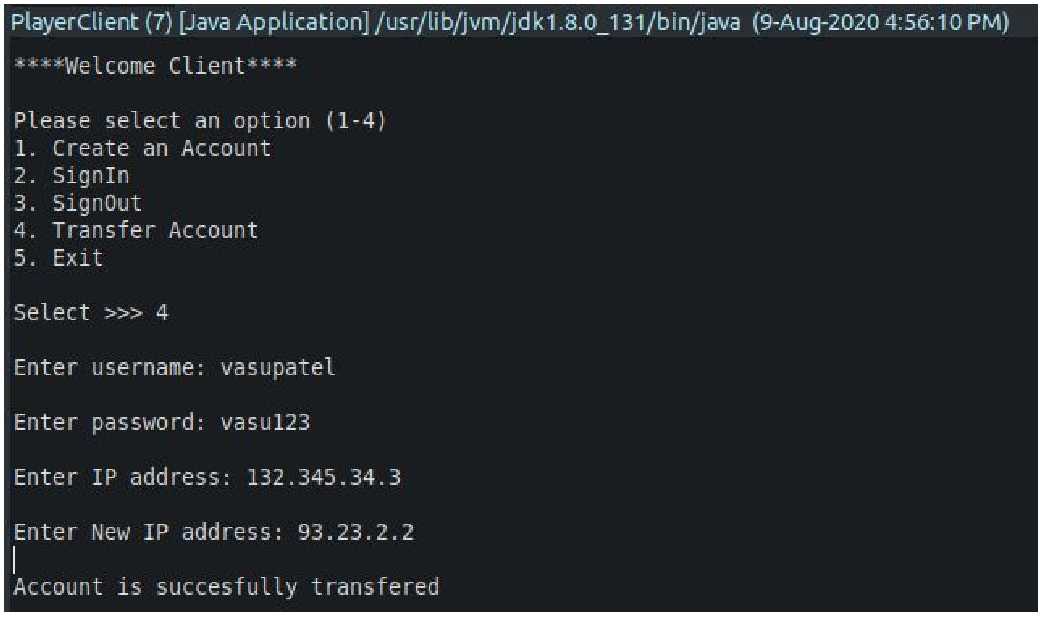
Sign out operation is straightforward. The player just needs to provide a username and the IP address of his/her location. If the user with given username exists and online then it will be set to an offline and sign out message will be sent to the player.



```
ReplicaManager [Java Application]
R3Response: Something went wrong with the server !


R1:0 R2:0 R3:2


R1Response: Account signed out successfully...
R2Response: Account signed out successfully...
R3Response: Something went wrong with the server !


R1:0 R2:0 R3:3


Stopping R3...

AsianServer is killed...
EuropeanServer is killed...
NorthAmericanServer is killed...

Starting a new R3...

AsianServer ready and waiting ...
EuropeanServer ready and waiting ...
NorthAmericanServer ready and waiting ...

UDPHeartBeat Message : Reduce maxAttempts for the server running on 6003
```

Figure : Sign out an account (back-end)

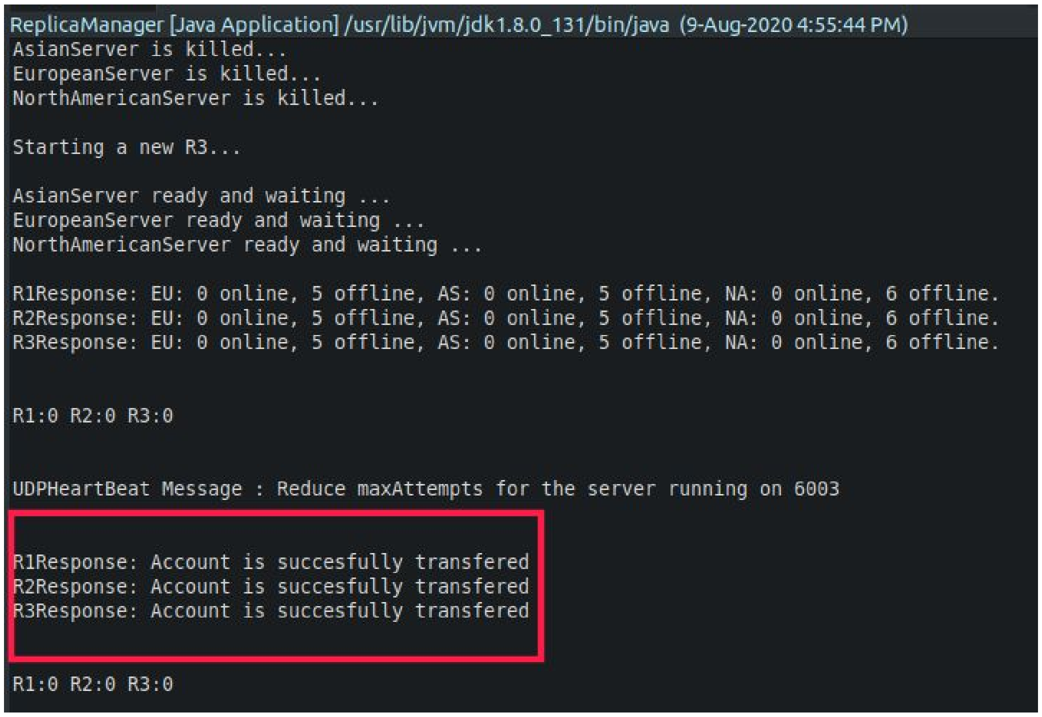
4. Transfer an account



```
PlayerClient (7) [Java Application] /usr/lib/jvm/jdk1.8.0_131/bin/java (9-Aug-2020 4:56:10 PM)
****Welcome Client****

Please select an option (1-4)
1. Create an Account
2. SignIn
3. SignOut
4. Transfer Account
5. Exit

Select >>> 4

Enter username: vasupatel

Enter password: vasu123

Enter IP address: 132.345.34.3

Enter New IP address: 93.23.2.2

Account is succesfully transfered
```

Figure: Transfer an account

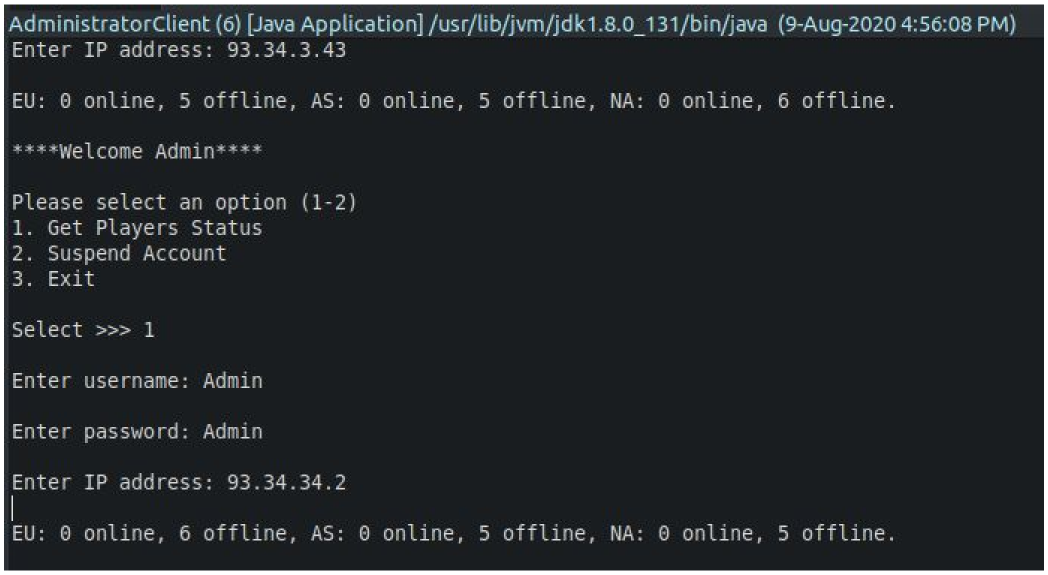
The transfer account will take 4 parameters namely, username, password, IP address (Old IP address), and New IP address. The account will be transferred from an old server to a new server (both will be detected by each IP separately). The whole process is explained in-depth under an atomicity section.



```
ReplicaManager [Java Application] /usr/lib/jvm/jdk1.8.0_131/bin/java (9-Aug-2020 4:55:44 PM)
AsianServer is killed...
EuropeanServer is killed...
NorthAmericanServer is killed...

Starting a new R3...

AsianServer ready and waiting ...
EuropeanServer ready and waiting ...
NorthAmericanServer ready and waiting ...

R1Response: EU: 0 online, 5 offline, AS: 0 online, 5 offline, NA: 0 online, 6 offline.
R2Response: EU: 0 online, 5 offline, AS: 0 online, 5 offline, NA: 0 online, 6 offline.
R3Response: EU: 0 online, 5 offline, AS: 0 online, 5 offline, NA: 0 online, 6 offline.


R1:0 R2:0 R3:0


UDPHeartBeat Message : Reduce maxAttempts for the server running on 6003


R1Response: Account is succesfully transfered
R2Response: Account is succesfully transfered
R3Response: Account is succesfully transfered


R1:0 R2:0 R3:0
```

Figure: Transfer an account (back-end)

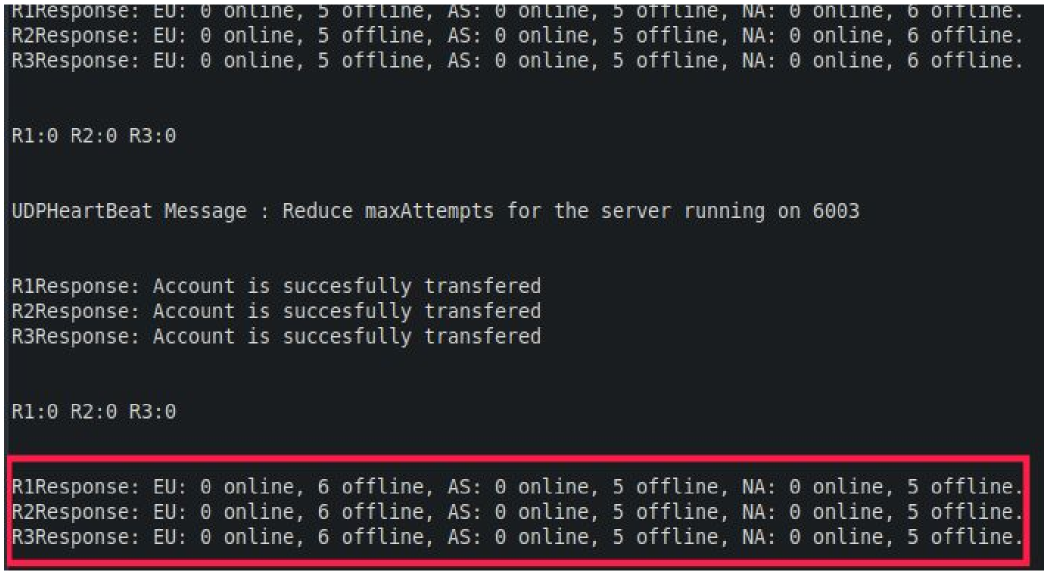
5. Get the players' status



```
AdministratorClient (6) [Java Application] /usr/lib/jvm/jdk1.8.0_131/bin/java (9-Aug-2020 4:56:08 PM)
Enter IP address: 93.34.3.43

EU: 0 online, 5 offline, AS: 0 online, 5 offline, NA: 0 online, 6 offline.

****Welcome Admin****

Please select an option (1-2)
1. Get Players Status
2. Suspend Account
3. Exit

Select >>> 1

Enter username: Admin

Enter password: Admin

Enter IP address: 93.34.34.2

EU: 0 online, 6 offline, AS: 0 online, 5 offline, NA: 0 online, 5 offline.
```

Figure: Get players' status

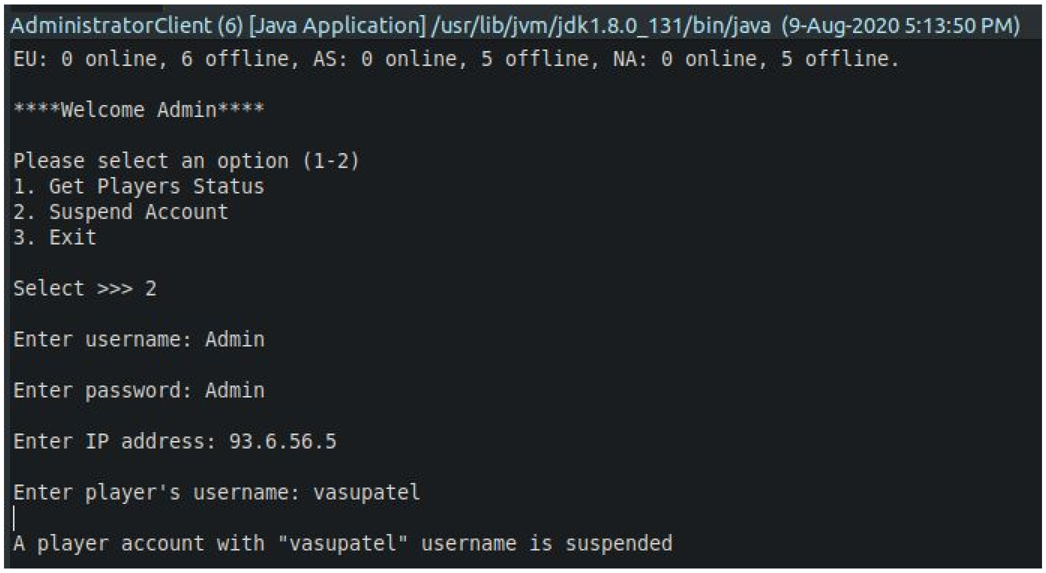
This operation will display the status of all players in the distributed system. It will ask for an Admin's username and password and the IP address on any of one server. The request will go to that server and that server will send UDP requests to the other two servers to know their players' status. Once it will get responses it will merge the response and send it to an AdminClient.



```
R1Response: EU: 0 online, 5 offline, AS: 0 online, 5 offline, NA: 0 online, 6 offline.
R2Response: EU: 0 online, 5 offline, AS: 0 online, 5 offline, NA: 0 online, 6 offline.
R3Response: EU: 0 online, 5 offline, AS: 0 online, 5 offline, NA: 0 online, 6 offline.


R1:0 R2:0 R3:0


UDPHeartBeat Message : Reduce maxAttempts for the server running on 6003


R1Response: Account is succesfully transfered
R2Response: Account is succesfully transfered
R3Response: Account is succesfully transfered


R1:0 R2:0 R3:0


R1Response: EU: 0 online, 6 offline, AS: 0 online, 5 offline, NA: 0 online, 5 offline.
R2Response: EU: 0 online, 6 offline, AS: 0 online, 5 offline, NA: 0 online, 5 offline.
R3Response: EU: 0 online, 6 offline, AS: 0 online, 5 offline, NA: 0 online, 5 offline.
```

Figure: Get players' status (back-end

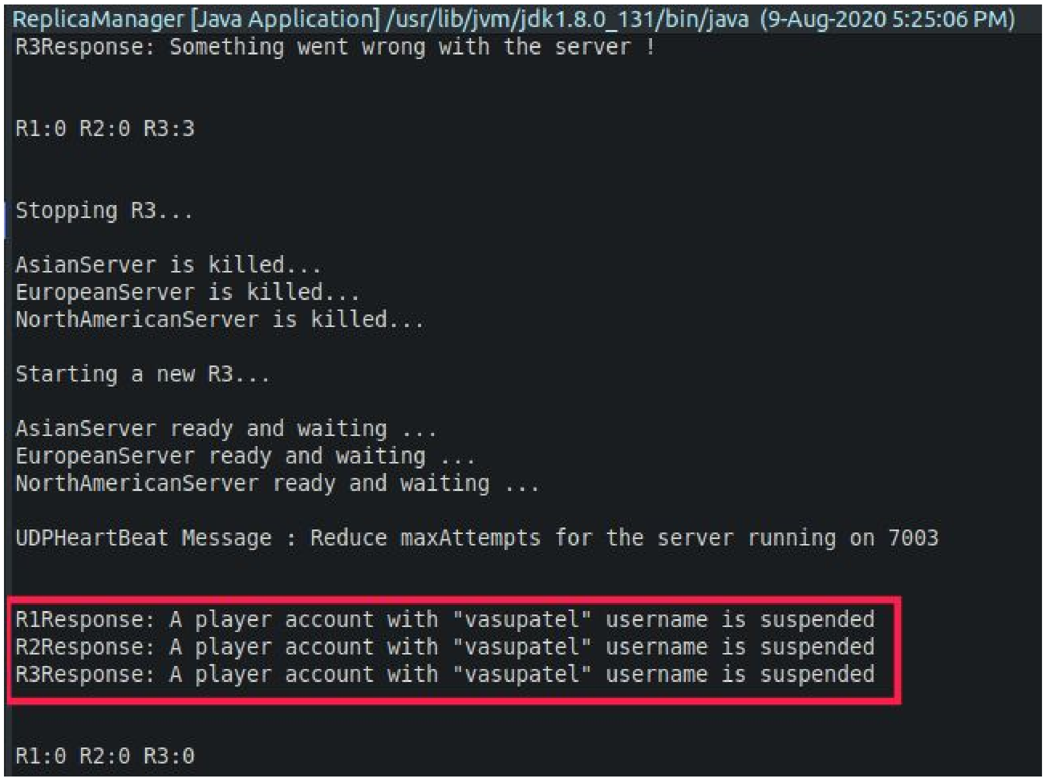
6. Suspend an account



```
AdministratorClient (6) [Java Application] /usr/lib/jvm/jdk1.8.0_131/bin/java (9-Aug-2020 5:13:50 PM)
EU: 0 online, 6 offline, AS: 0 online, 5 offline, NA: 0 online, 5 offline.

****Welcome Admin****

Please select an option (1-2)
1. Get Players Status
2. Suspend Account
3. Exit

Select >>> 2

Enter username: Admin

Enter password: Admin

Enter IP address: 93.6.56.5

Enter player's username: vasupatel

A player account with "vasupatel" username is suspended
```

Figure .: Suspend an account

The suspended account will also take an Admin's username and password. Moreover, it will also take an IP address of a server and the username which needs to be suspended. The suspended account will delete the user with the given username and send a response back to AdminClient. If the user doesn't present the server with a given username it will send an error message.

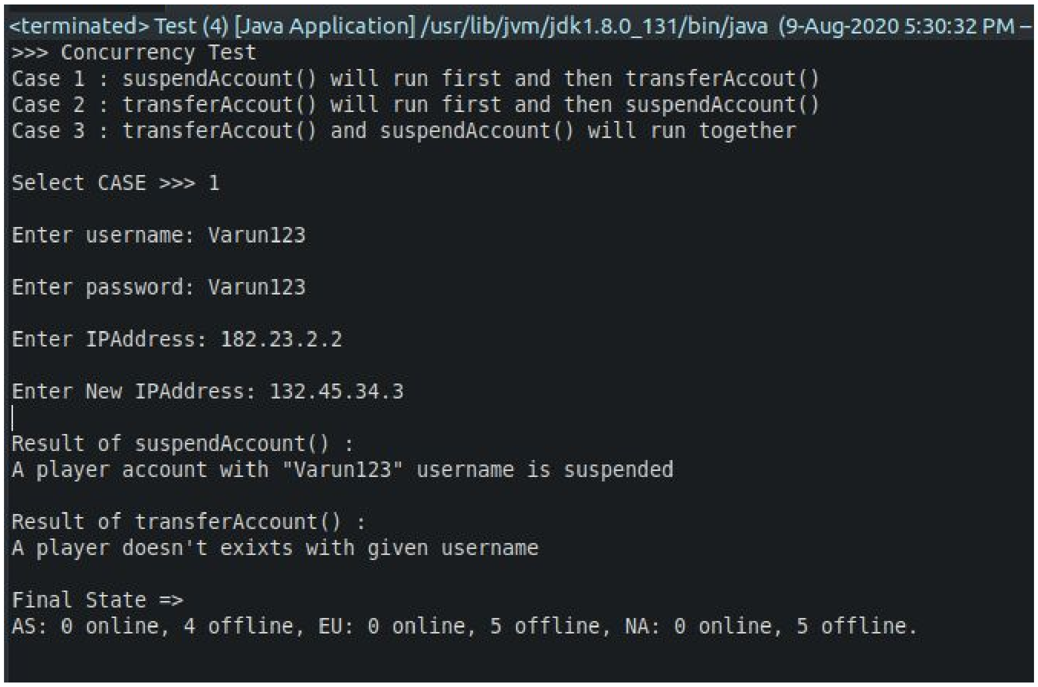


```
ReplicaManager [Java Application] /usr/lib/jvm/jdk1.8.0_131/bin/java (9-Aug-2020 5:25:06 PM)
R3Response: Something went wrong with the server !


R1:0 R2:0 R3:3


Stopping R3...

AsianServer is killed...
EuropeanServer is killed...
NorthAmericanServer is killed...

Starting a new R3...

AsianServer ready and waiting ...
EuropeanServer ready and waiting ...
NorthAmericanServer ready and waiting ...

UDPHeartBeat Message : Reduce maxAttempts for the server running on 7003


R1Response: A player account with "vasupatel" username is suspended
R2Response: A player account with "vasupatel" username is suspended
R3Response: A player account with "vasupatel" username is suspended


R1:0 R2:0 R3:0
```

Figure .: Suspend an account (back-end)

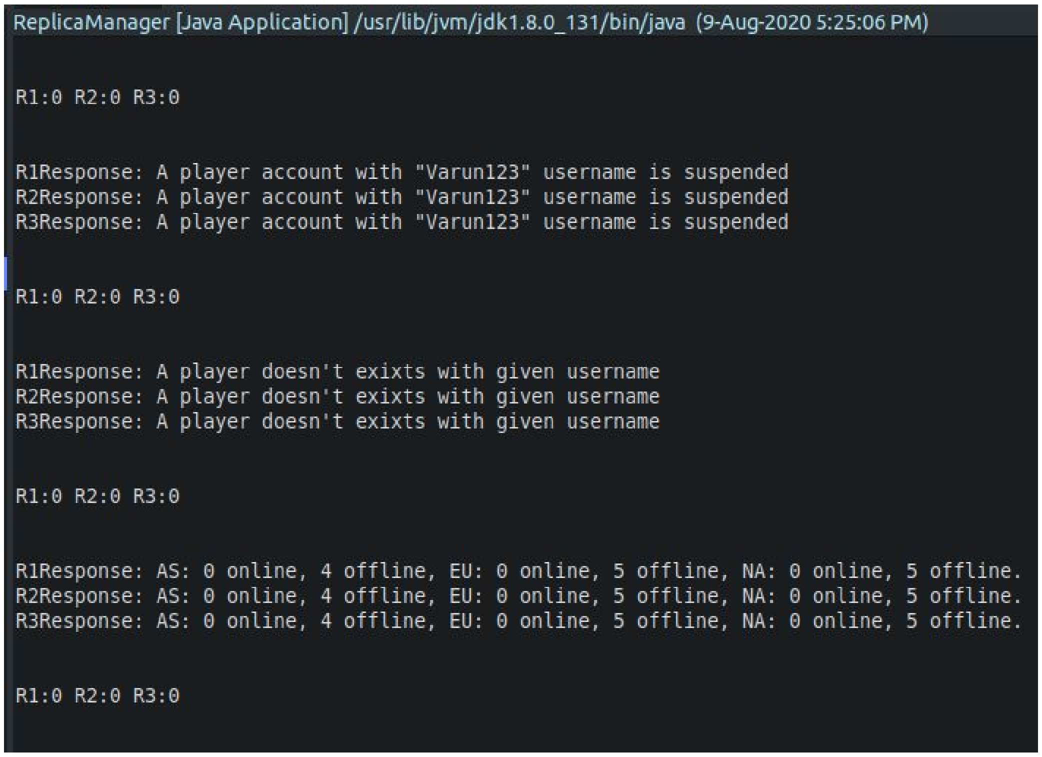
7. Concurrency test (Case 1)

   Here, I have created a new Java file called Test.java in the Client package. Which will test concurrency in scenario 7,8 and 9th. In case 1, the suspendAccount() will execute first and after 1 second the transferAccount() will be called. To simulate the real-life scenario, I have used the Thread.sleep() function. The output would be as shown above the account will be suspended and when another thread will try to transfer an account it will give an error.



```
<terminated> Test (4) [Java Application] /usr/lib/jvm/jdk1.8.0_131/bin/java (9-Aug-2020 5:30:32 PM –
>>> Concurrency Test
Case 1 : suspendAccount() will run first and then transferAccout()
Case 2 : transferAccout() will run first and then suspendAccount()
Case 3 : transferAccout() and suspendAccount() will run together

Select CASE >>> 1

Enter username: Varun123

Enter password: Varun123

Enter IPAddress: 182.23.2.2

Enter New IPAddress: 132.45.34.3

Result of suspendAccount() :
A player account with "Varun123" username is suspended

Result of transferAccount() :
A player doesn't exixts with given username

Final State =>
AS: 0 online, 4 offline, EU: 0 online, 5 offline, NA: 0 online, 5 offline.
```

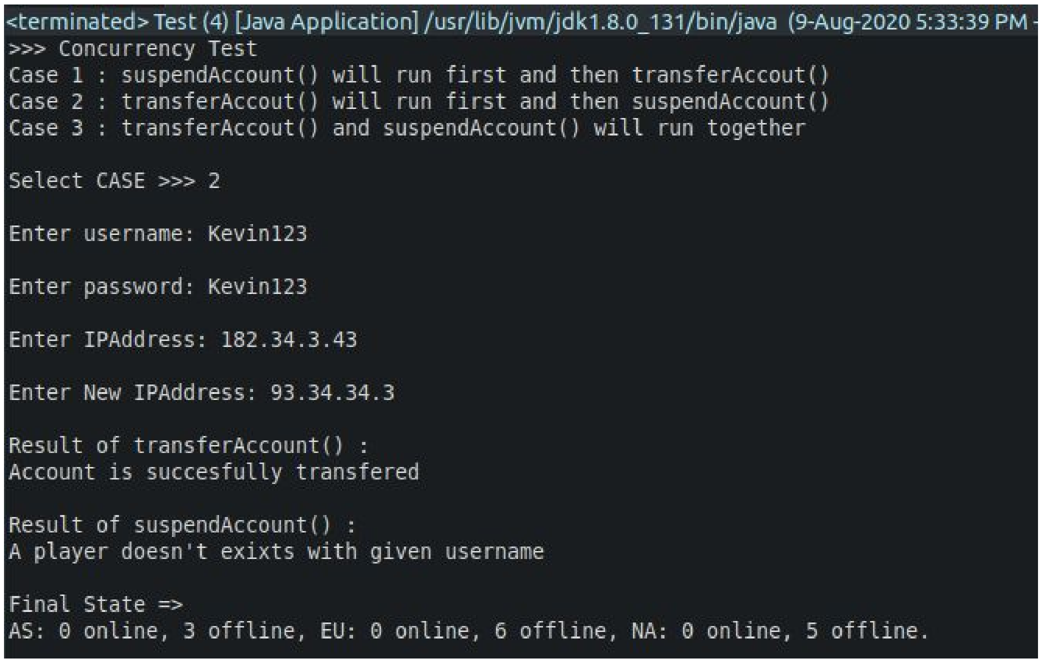
Figure : Concurrency Test (Case 1)



```
ReplicaManager [Java Application] /usr/lib/jvm/jdk1.8.0_131/bin/java (9-Aug-2020 5:25:06 PM)


R1:0 R2:0 R3:0


R1Response: A player account with "Varun123" username is suspended
R2Response: A player account with "Varun123" username is suspended
R3Response: A player account with "Varun123" username is suspended


R1:0 R2:0 R3:0


R1Response: A player doesn't exixts with given username
R2Response: A player doesn't exixts with given username
R3Response: A player doesn't exixts with given username


R1:0 R2:0 R3:0


R1Response: AS: 0 online, 4 offline, EU: 0 online, 5 offline, NA: 0 online, 5 offline.
R2Response: AS: 0 online, 4 offline, EU: 0 online, 5 offline, NA: 0 online, 5 offline.
R3Response: AS: 0 online, 4 offline, EU: 0 online, 5 offline, NA: 0 online, 5 offline.


R1:0 R2:0 R3:0
```

Figure : Concurrency Test (Case 1 - backend)

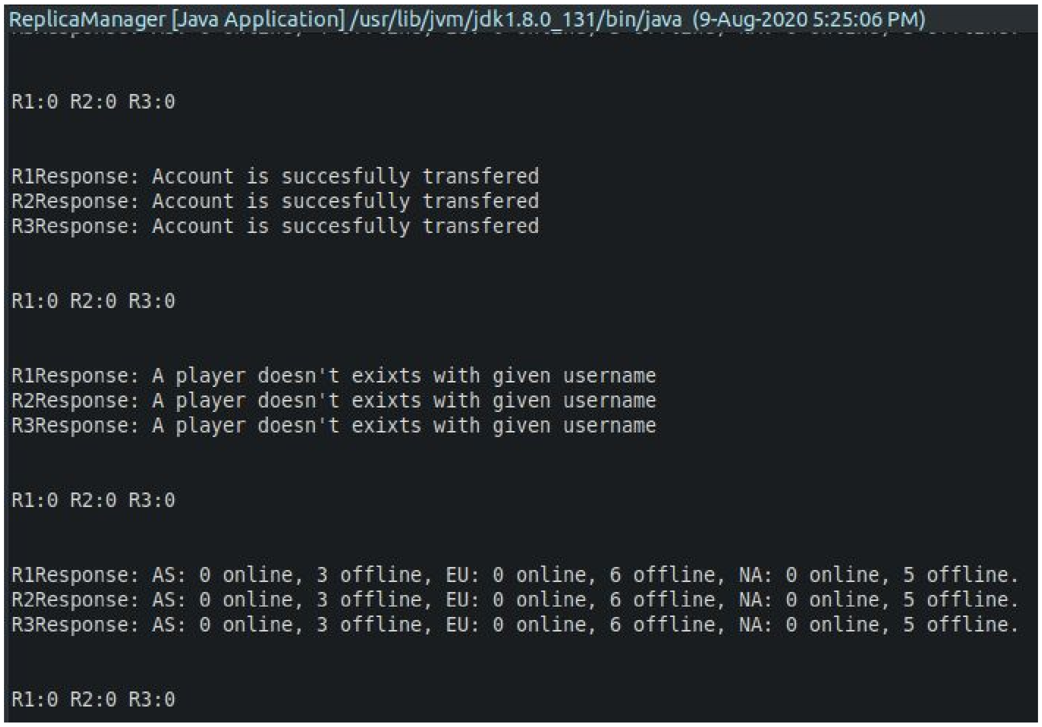
8. Concurrency test (Case 2)



```
<terminated> Test (4) [Java Application] /usr/lib/jvm/jdk1.8.0_131/bin/java (9-Aug-2020 5:33:39 PM -
>>> Concurrency Test
Case 1 : suspendAccount() will run first and then transferAccout()
Case 2 : transferAccout() will run first and then suspendAccount()
Case 3 : transferAccout() and suspendAccount() will run together

Select CASE >>> 2

Enter username: Kevin123

Enter password: Kevin123

Enter IPAddress: 182.34.3.43

Enter New IPAddress: 93.34.34.3

Result of transferAccount() :
Account is succesfully transfered

Result of suspendAccount() :
A player doesn't exixts with given username

Final State =>
AS: 0 online, 3 offline, EU: 0 online, 6 offline, NA: 0 online, 5 offline.
```

Figure: Concurrency Test (Case 2)

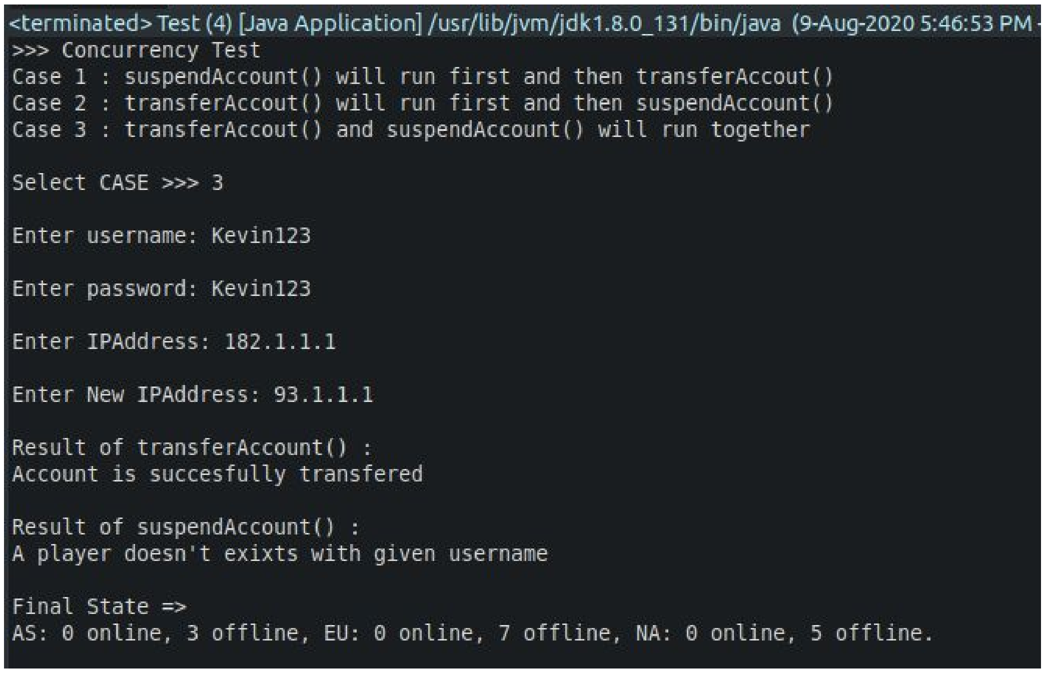
This case is the vice versa of case 1. Here the transferAccount() will run first and after 1 second delay suspendAccount() will run. So, the user account would be transferred to another server before it gets suspended.



```
ReplicaManager [Java Application] /usr/lib/jvm/jdk1.8.0_131/bin/java (9-Aug-2020 5:25:06 PM)

R1:0 R2:0 R3:0


R1Response: Account is succesfully transfered
R2Response: Account is succesfully transfered
R3Response: Account is succesfully transfered


R1:0 R2:0 R3:0


R1Response: A player doesn't exixts with given username
R2Response: A player doesn't exixts with given username
R3Response: A player doesn't exixts with given username


R1:0 R2:0 R3:0


R1Response: AS: 0 online, 3 offline, EU: 0 online, 6 offline, NA: 0 online, 5 offline.
R2Response: AS: 0 online, 3 offline, EU: 0 online, 6 offline, NA: 0 online, 5 offline.
R3Response: AS: 0 online, 3 offline, EU: 0 online, 6 offline, NA: 0 online, 5 offline.


R1:0 R2:0 R3:0
```

Figure: Concurrency Test (Case 2 back-end)

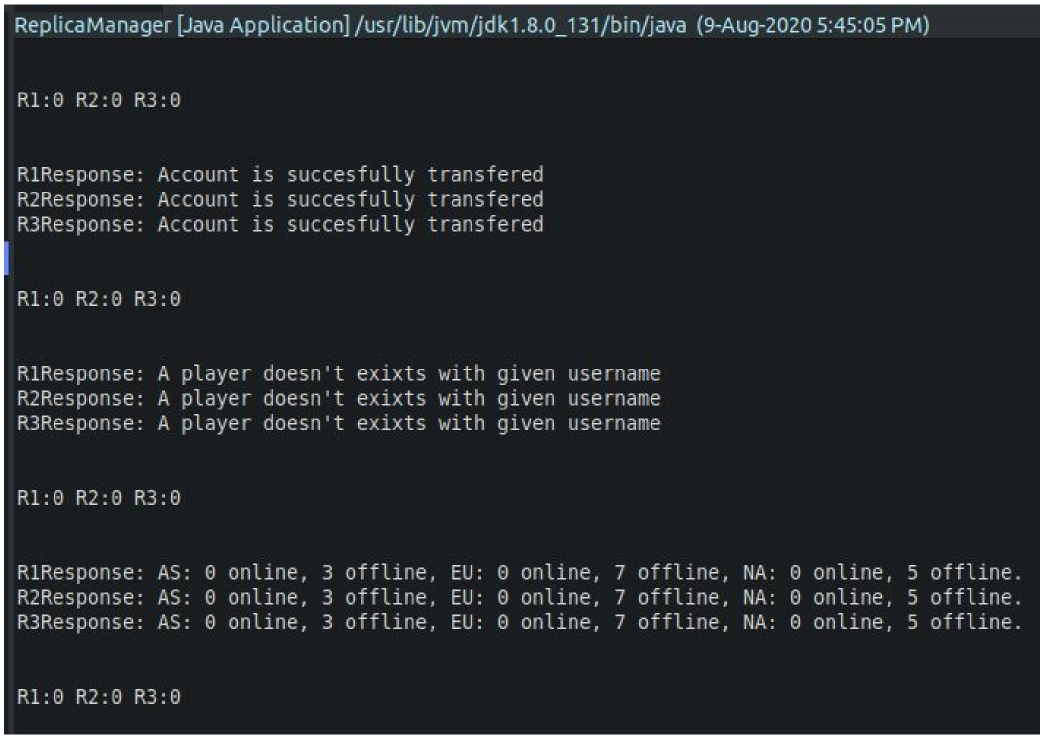
9. Concurrency test (Case 3)
   This is a very special case for concurrency testing. I removed synchronized keywords from transferAccount() and suspendAccount() methods from all 3 servers to see what will happen if the server gets two requests at the same time and how it will be handled by the server.



```
<terminated> Test (4) [Java Application] /usr/lib/jvm/jdk1.8.0_131/bin/java (9-Aug-2020 5:46:53 PM
>>> Concurrency Test
Case 1 : suspendAccount() will run first and then transferAccout()
Case 2 : transferAccout() will run first and then suspendAccount()
Case 3 : transferAccout() and suspendAccount() will run together

Select CASE >>> 3

Enter username: Kevin123

Enter password: Kevin123

Enter IPAddress: 182.1.1.1

Enter New IPAddress: 93.1.1.1

Result of transferAccount() :
Account is succesfully transfered

Result of suspendAccount() :
A player doesn't exixts with given username

Final State =>
AS: 0 online, 3 offline, EU: 0 online, 7 offline, NA: 0 online, 5 offline.
```

Figure : Concurrency Test (Case 3)

As you can see in the screenshot, both threads executed at the same time so when the account was transferring from one server to another server meanwhile that account was suspended by an Admin. So, when the response from server 2 will come and server 1 will try to remove that user from the database. But that user was suspended by the Admin so deletion operation will fail, and it will start the rollback process. In this process, it will send a request to server 2 to remove that user's account because it was suspended by an Admin when it was transferring, and the rollback will be successful after that.



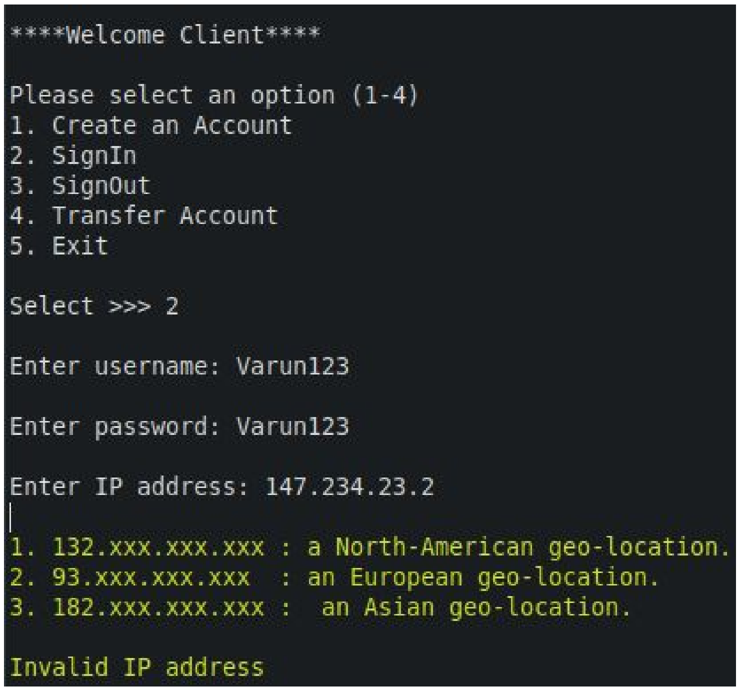
```
ReplicaManager [Java Application] /usr/lib/jvm/jdk1.8.0_131/bin/java (9-Aug-2020 5:45:05 PM)


R1:0 R2:0 R3:0


R1Response: Account is succesfully transfered
R2Response: Account is succesfully transfered
R3Response: Account is succesfully transfered


R1:0 R2:0 R3:0


R1Response: A player doesn't exixts with given username
R2Response: A player doesn't exixts with given username
R3Response: A player doesn't exixts with given username


R1:0 R2:0 R3:0


R1Response: AS: 0 online, 3 offline, EU: 0 online, 7 offline, NA: 0 online, 5 offline.
R2Response: AS: 0 online, 3 offline, EU: 0 online, 7 offline, NA: 0 online, 5 offline.
R3Response: AS: 0 online, 3 offline, EU: 0 online, 7 offline, NA: 0 online, 5 offline.


R1:0 R2:0 R3:0
```

Figure : Concurrency Test (Case 3 back-end)

Note: To test this case you must need to remove a synchronized keyword from transferAccount() and suspendAccount() methods from all 3 servers

10. Other Validations (IP, Age, Username and Password)

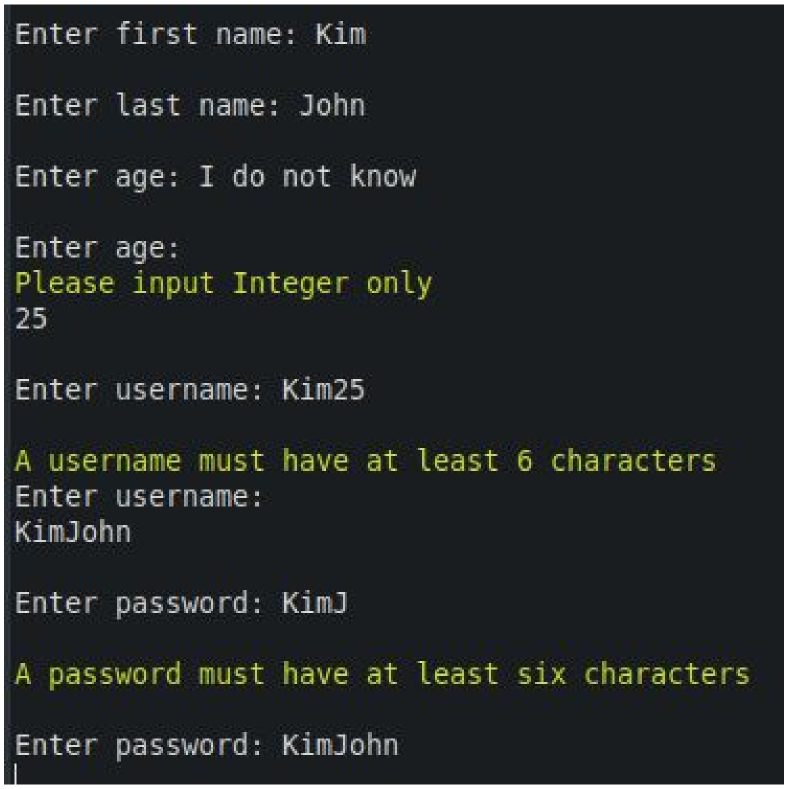
- IP validation



```
****Welcome Client****

Please select an option (1-4)
1. Create an Account
2. SignIn
3. SignOut
4. Transfer Account
5. Exit

Select >>> 2

Enter username: Varun123

Enter password: Varun123

Enter IP address: 147.234.23.2

1. 132.xxx.xxx.xxx : a North-American geo-location.
2. 93.xxx.xxx.xxx  : an European geo-location.
3. 182.xxx.xxx.xxx :  an Asian geo-location.

Invalid IP address
```

Figure : IP validations

- Age, Username and Password validations

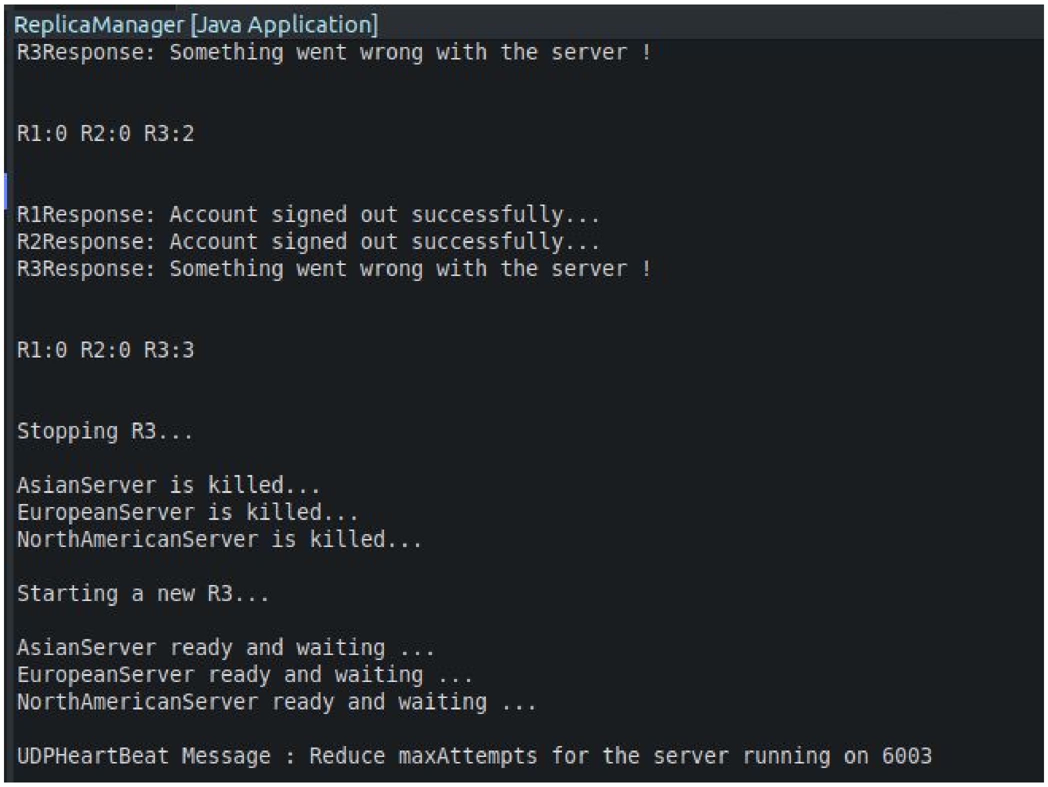


```
Enter first name: Kim

Enter last name: John

Enter age: I do not know

Enter age:
Please input Integer only
25

Enter username: Kim25

A username must have at least 6 characters
Enter username:
KimJohn

Enter password: KimJ

A password must have at least six characters

Enter password: KimJohn
```

Figure : Age, Username and Password validations

11. Byzantine Software failure

When the project starts, one replica will produce the wrong output continuously 3 times. Which can present the Byzantine software failure. After the 3rd wrong results the replica manager will restart that replica which was producing the wrong output. So it can start producing correct output.



```
ReplicaManager [Java Application]
R3Response: Something went wrong with the server !


R1:0 R2:0 R3:2


R1Response: Account signed out successfully...
R2Response: Account signed out successfully...
R3Response: Something went wrong with the server !


R1:0 R2:0 R3:3


Stopping R3...

AsianServer is killed...
EuropeanServer is killed...
NorthAmericanServer is killed...

Starting a new R3...

AsianServer ready and waiting ...
EuropeanServer ready and waiting ...
NorthAmericanServer ready and waiting ...

UDPHeartBeat Message : Reduce maxAttempts for the server running on 6003
```

Figure : Byzantine Software failure & Reliability of UDP

12. Reliability of UDP

To make UDP reliable I used the HeartBeat mechanism which is already mentioned in the report. As you can see in the above image the last line shows that the UDP HeartBeat message wasn’t reached to the server. If it will fail 3 times continuously then it will notify to the RM and the RM will restart the complete replica

❖ important/Difficult Part in The Project

1. Atomicity

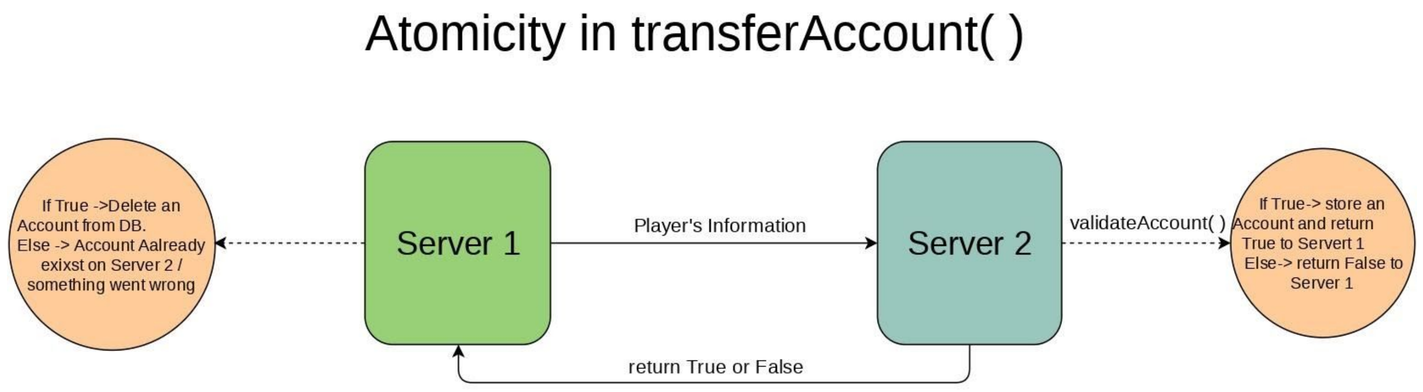

Atomicity in transferAccount( )
If True ->Delete an Account from DB.
Else -> Account Aalready exixst on Server 2 / something went wrong
Server 1
Player's Information
Server 2
validateAccount( )
If True-> store an Account and return True to Servert 1
Else-> return False to Server 1
return True or False

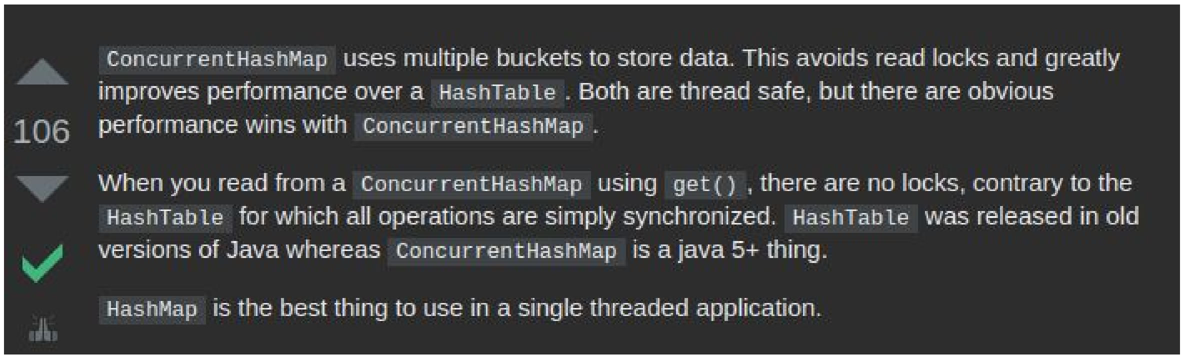
Figure : Atomicity in transferAccount()

The important part of the project was to achieve atomicity in transferAccount() operation. It can be done in multiple ways. In the above diagram, I tried to show a simple way to achieve atomicity.

When the server 1 ger account transfer request from a player it will send a new request to server 2 via UDP tunnel with all data of the user. When server 2 will receive a UDP request it will validate an account and check if any user is already present with the receiver user's username if the user with the same name is present then the server 2 will send "False" to server 1 and server 1 will send a message to the client that the user already exists with given username. Else, it will create a new account for that transferred user and return "True" to server 1.

If server 1 will receive a "True" from server 2 then it will delete that user's account from the database. For any reason If the deletion operation won't succeed then, the rollback process will start, and it will bring the system in its original state before the transfer operation was started.

2. ConcurrentHashMap Vs. HashTable in Java


106

`ConcurrentHashMap` uses multiple buckets to store data. This avoids read locks and greatly improves performance over a `HashTable`. Both are thread safe, but there are obvious performance wins with `ConcurrentHashMap`.

When you read from a `ConcurrentHashMap` using `get()`, there are no locks, contrary to the `HashTable` for which all operations are simply synchronized. `HashTable` was released in old versions of Java whereas `ConcurrentHashMap` is a java 5+ thing.

`HashMap` is the best thing to use in a single threaded application.

Figure : ConcurrentHashMap vs HashTable (Ref. Stack Overflow)

To maximize the concurrency and make the data structure more thread-safe I decided to use ConcurrentHashMap which gives better performance. ConcurrentHashMap only locks the data structure while writing.

3. A Special Method: getConfig()

To reduce the run time complications, I have created a new method called getConfig() in each server files. which will pass the arguments for ORB.int() method which is passing statically from the getConfig() method. The ORB port is defined in ORB_PORT which is 1050.

4. UDP HeartBeat Monitor

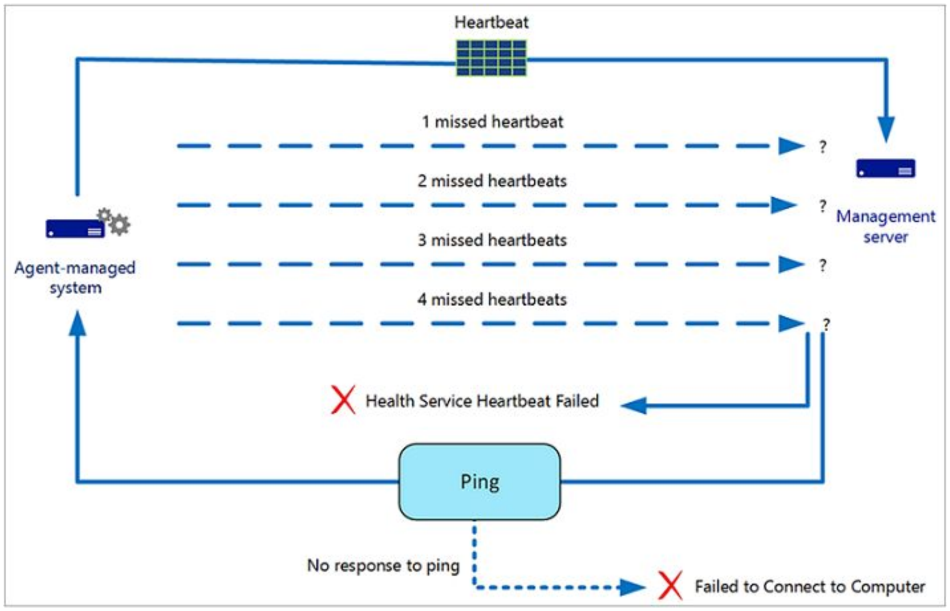


Figure : HeartBeat System (Ref. Google Images)

To make UDP more reliable I have added a HeartBeat checker. It will keep sending the UDP requests to all the running servers in the whole system. If any server will go down the HeartBeat monitor will resend the request 3 times after that it will notify the replica manager so that the replica manager will restart that particular replica.

❖ Distributed System UML Design

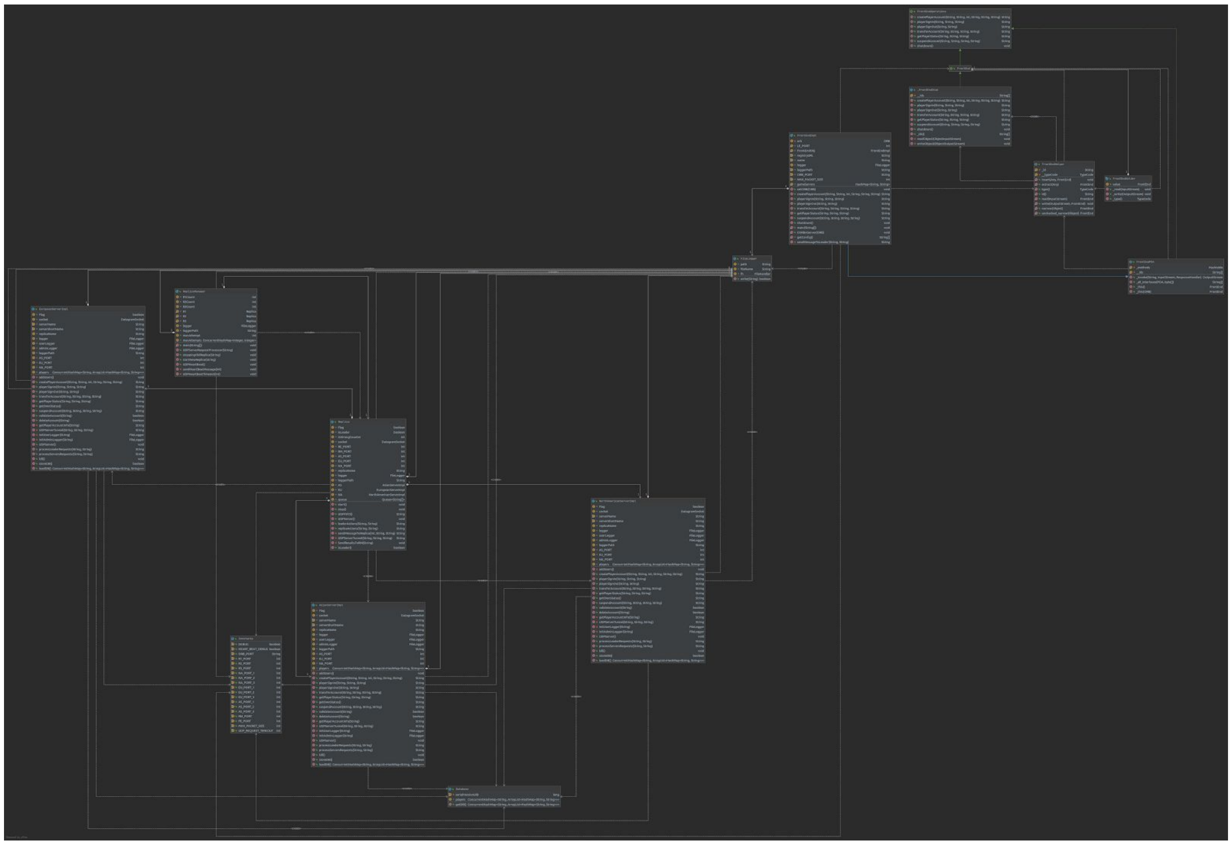


Figure: UML diagram

The figure represents the UML class diagrams. In figure all classes are covered, and it represents the different relation between all classes.

Note: To see the diagrams properly please goto ‘uml’ folder which is inside the Project folder.

❖ Project’s Defaults

- CORBA Port
  ORB_PORT = 1050

- Replicas’ Ports
  R1_PORT = 4001
  R2_PORT = 4002
  R2_PORT = 4003

- North American Servers’ Ports
  NA_PORT_1 = 5001
  NA_PORT_2 = 5002
  NA_PORT_3 = 5003

- European Servers’ Ports
  EU_PORT_1 = 6001
  EU_PORT_2 = 6002
  EU_PORT_3 = 6003

- Asian Servers’ Ports
  AS_PORT_1 = 7001
  AS_PORT_2 = 7002
  AS_PORT_3 = 7003

- Replica Manager Port
  RM_PORT = 8001

- Front-End Port
  FE_PORT = 9001

Note: Please kill all services which are running on the above port before running the project.

❖ Asian Server's Default Users (182.x.x.x)

Sr. No	Username	password
1	Bruce123	Bruce123
2	Charles123	Charles123
3	Adak123	Adak123
4	Varun123	Varun123
5	Kevin123	Kevin123

❖ European Server's Default Users (93.x.x.x)

Sr. No	Username	password
1	Fabienne123	Fabienne123
2	Darcie123	Darcie123
3	Philipa123	Philipa123
4	Davinia123	Davinia123
5	Gyles123	Gyles123

❖ North American Server's Default Users (132.x.x.x)

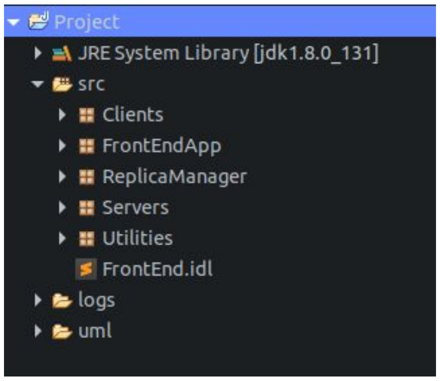
Sr. No	Username	password
1	Siddall123	Siddall123
2	Morce123	Morce123
3	Seabrook123	Seabrook123
4	Upton123	Upton123
5	Garfield123	Garfield123

❖ How to run a project?

The project was created using Oracle's Java 8.

Unzip project submission. There will be 1 file and 1 folder namely "Report.pdf" and "Project" folder. Put the "Project" folder in your Eclipse workspace and open the project in your Eclipse.

There are 4 packages in the folder namely client, server, services and logger.

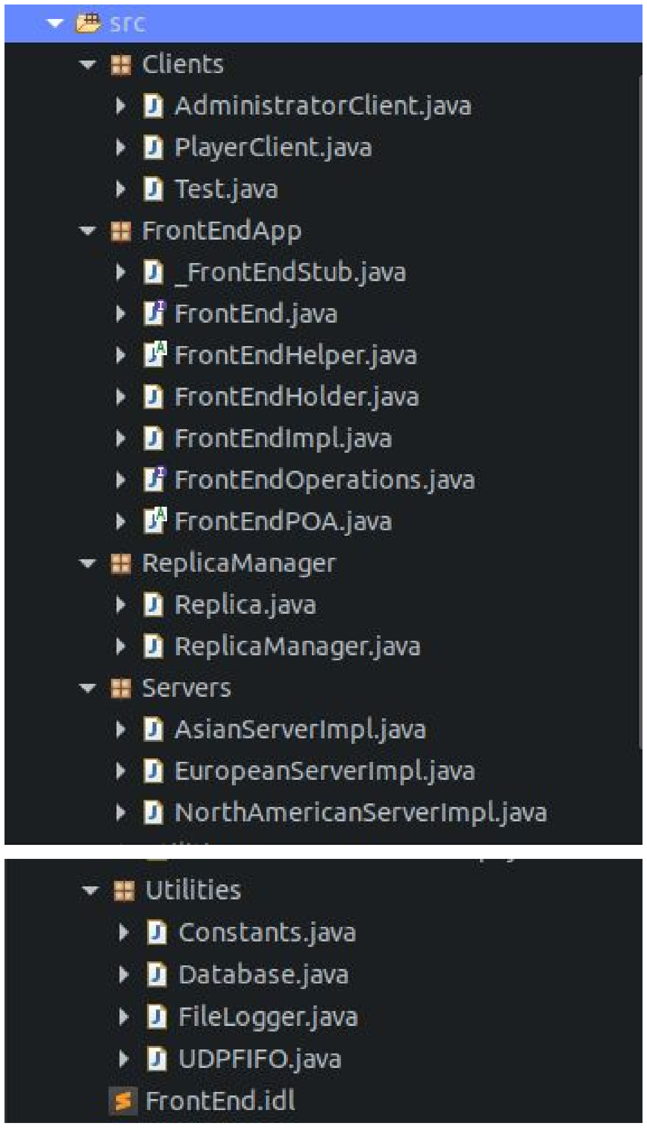


```
Project
  JRE System Library [jdk1.8.0_131]
  src
    Clients
    FrontEndApp
    ReplicaManager
    Servers
    Utilities
    FrontEnd.idl
  logs
  uml
```

Figure : Project's packages

Go to the FrontEndApp package and run FrontEndImpl.java It will start the FrontEnd. Then go to the ReplicaManager package and run ReplicaManager.java it will start the whole back-end servers and replicas etc. Now You can run any files from the Client package. That's it.

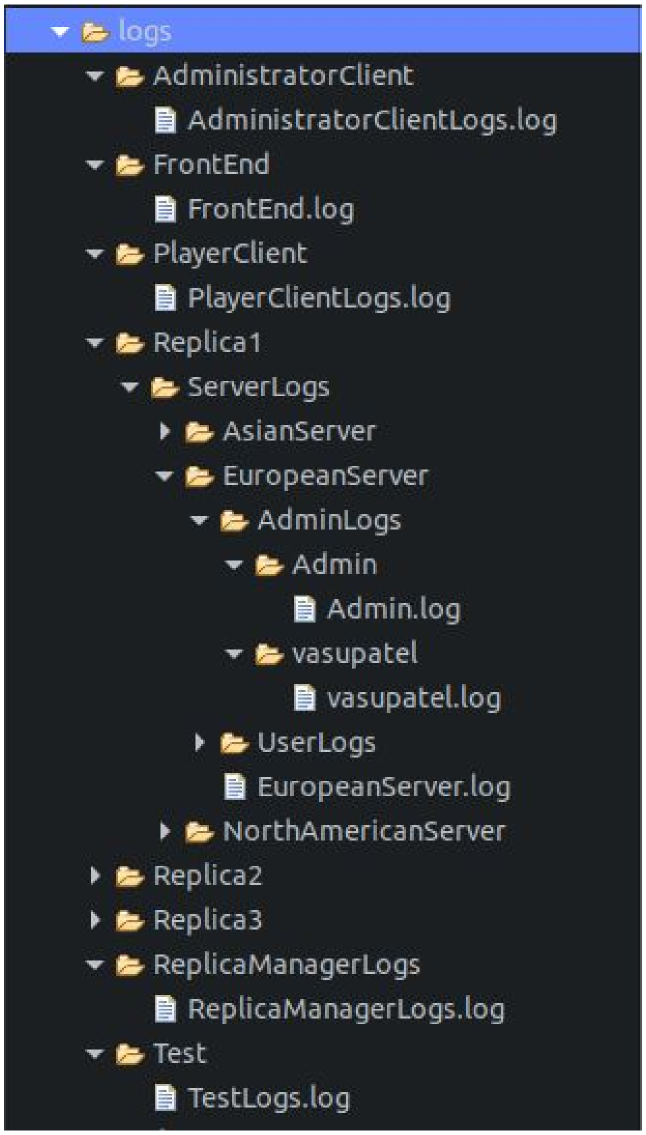
❖ src folder



```
src
  Clients
    AdministratorClient.java
    PlayerClient.java
    Test.java
  FrontEndApp
    _FrontEndStub.java
    FrontEnd.java
    FrontEndHelper.java
    FrontEndHolder.java
    FrontEndImpl.java
    FrontEndOperations.java
    FrontEndPOA.java
  ReplicaManager
    Replica.java
    ReplicaManager.java
  Servers
    AsianServerImpl.java
    EuropeanServerImpl.java
    NorthAmericanServerImpl.java
  Utilities
    Constants.java
    Database.java
    FileLogger.java
    UDPFIFO.java
  FrontEnd.idl
```

Figure : src folder

❖ logs folder



```
logs
    AdministratorClient
        AdministratorClientLogs.log
    FrontEnd
        FrontEnd.log
    PlayerClient
        PlayerClientLogs.log
    Replica1
        ServerLogs
            AsianServer
            EuropeanServer
                AdminLogs
                    Admin
                        Admin.log
                    vasupatel
                        vasupatel.log
                UserLogs
                EuropeanServer.log
            NorthAmericanServer
    Replica2
    Replica3
    ReplicaManagerLogs
        ReplicaManagerLogs.log
    Test
        TestLogs.log
```

Figure : Log Folder Structure

❖ UML folder

UML diagram is stored inside the UML folder which is located in the “Project” folder.

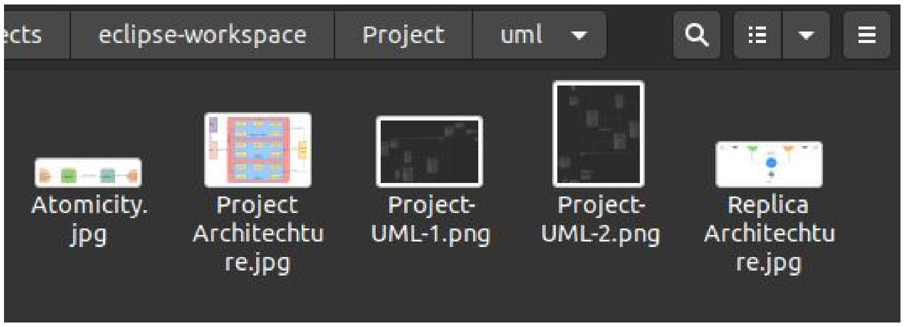
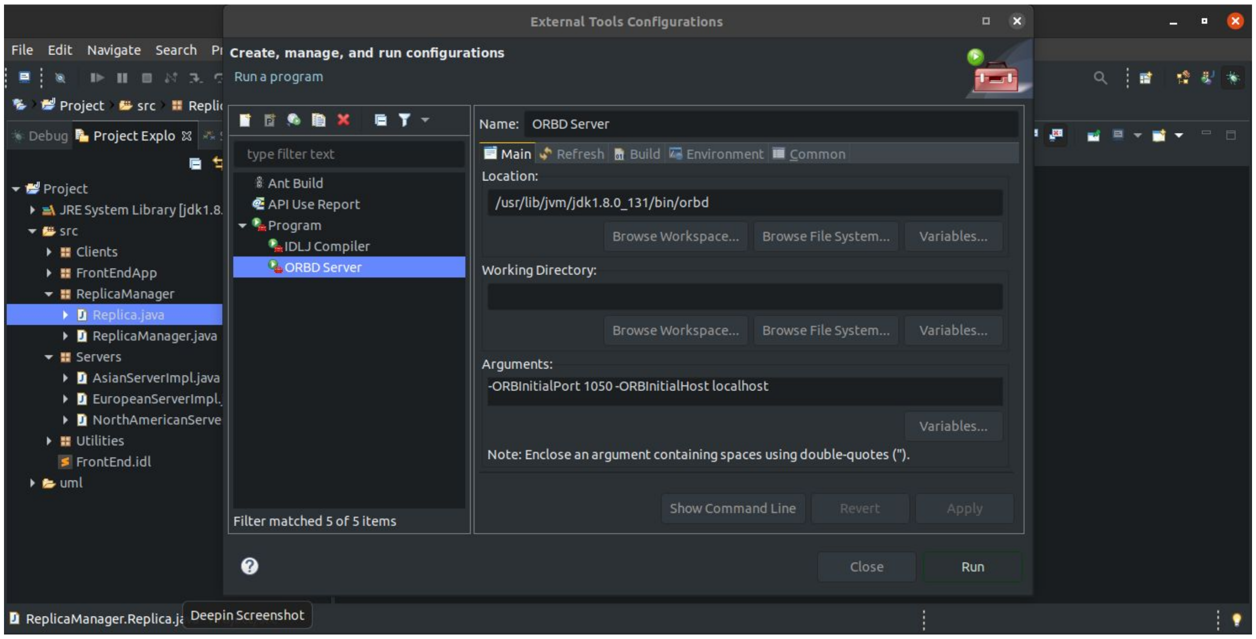


Figure : UML diagrams

❖ Staring ORBD Server



Name: ORBD Server

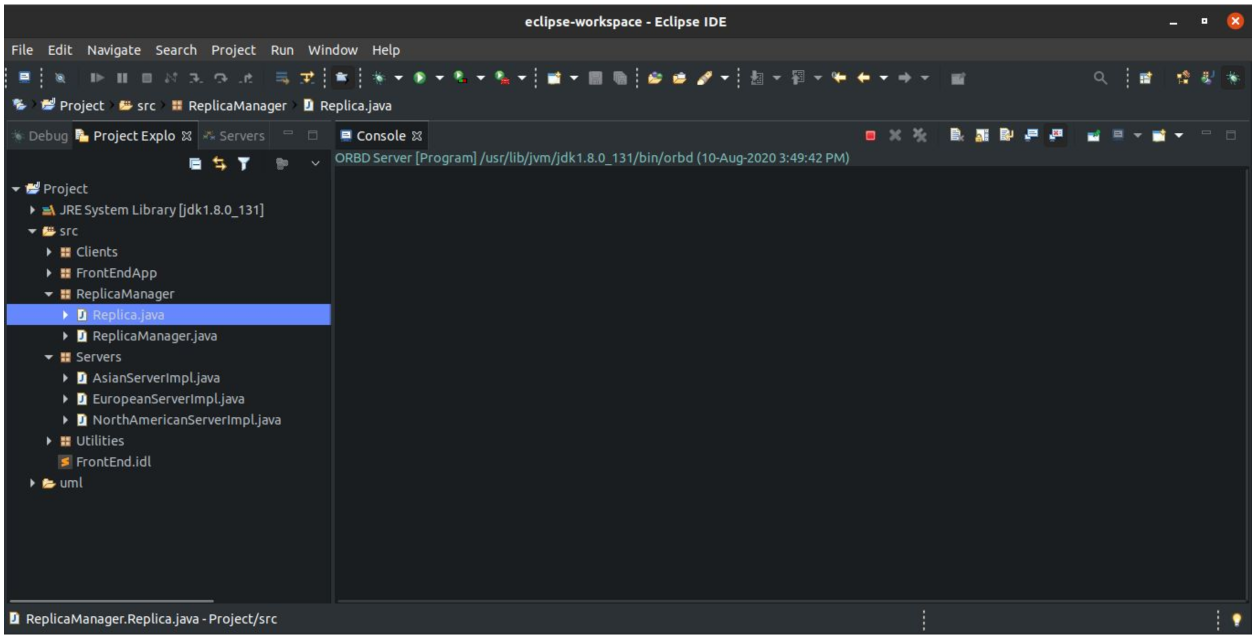
Location:
/usr/lib/jvm/jdk1.8.0_131/bin/orbd

Working Directory:

Arguments:
-ORBInitialPort 1050 -ORBInitialHost localhost

Note: Enclose an argument containing spaces using double-quotes (").

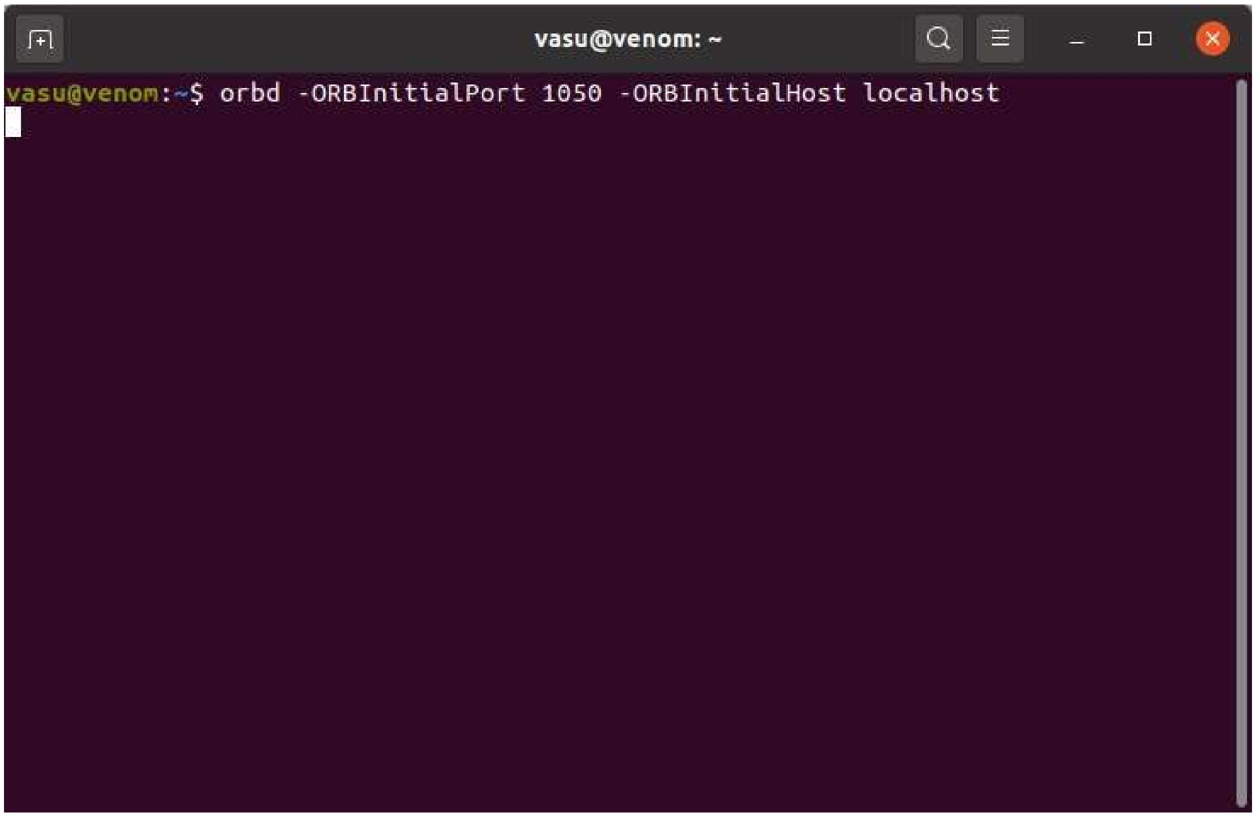
Figure : starting ORBD in Eclipse 1


eclipse-workspace - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Project
JRE System Library [jdk1.8.0_131]
src
Clients
FrontEndApp
ReplicaManager
Replica.java
ReplicaManager.java
Servers
AsianServerImpl.java
EuropeanServerImpl.java
NorthAmericanServerImpl.java
Utilities
FrontEnd.idl
uml
Console
ReplicaManager.Replica.java - Project/src

```
ORBD Server [Program] /usr/lib/jvm/jdk1.8.0_131/bin/orbd (10-Aug-2020 3:49:42 PM)
```

Figure : starting ORBD in Eclipse 2

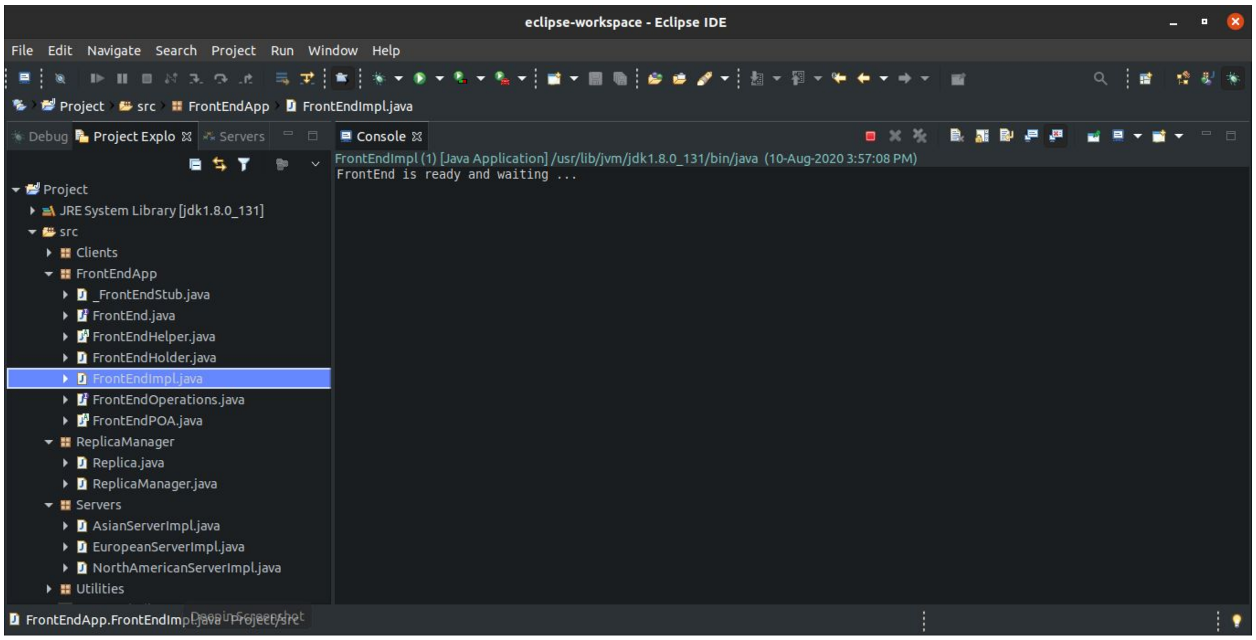
OR



```
vasu@venom:~$ orbd -ORBInitialPort 1050 -ORBInitialHost localhost
```

Figure : starting ORBD in Terminal or cmd

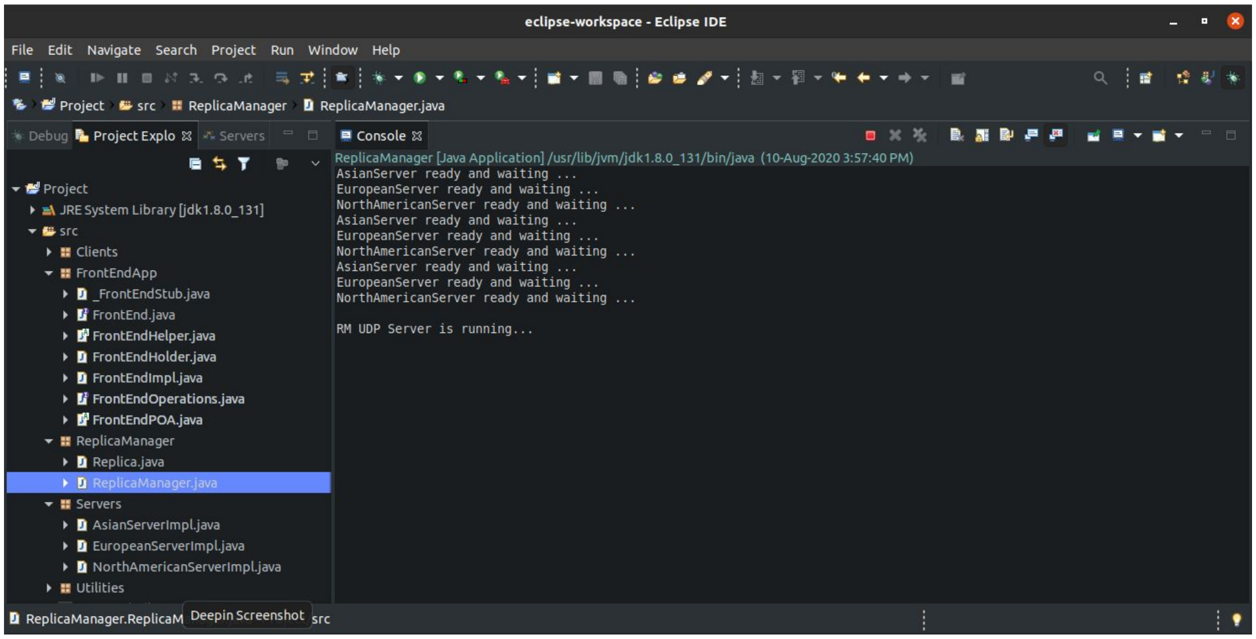
❖ Staring Front-End



```
FrontEndImpl (1) [Java Application] /usr/lib/jvm/jdk1.8.0_131/bin/java (10-Aug-2020 3:57:08 PM)
FrontEnd is ready and waiting ...
```

Figure : Starting the FrontEnd

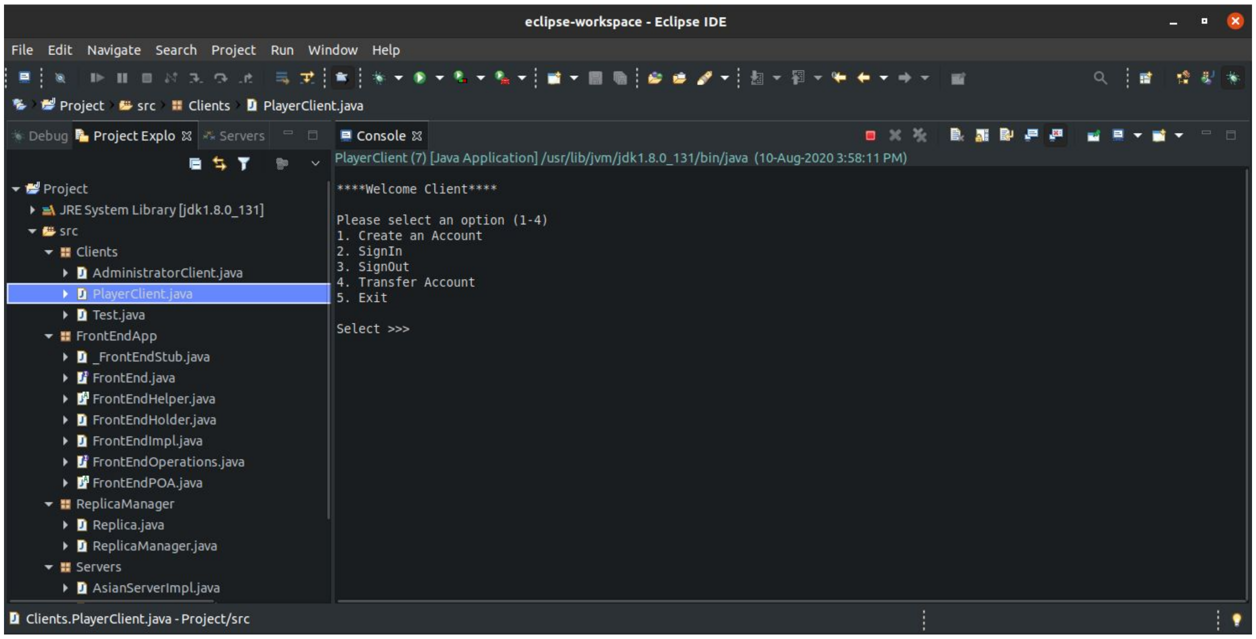
❖ Starting Replica Manager



```
ReplicaManager [Java Application] /usr/lib/jvm/jdk1.8.0_131/bin/java (10-Aug-2020 3:57:40 PM)
AsianServer ready and waiting ...
EuropeanServer ready and waiting ...
NorthAmericanServer ready and waiting ...
AsianServer ready and waiting ...
EuropeanServer ready and waiting ...
NorthAmericanServer ready and waiting ...
AsianServer ready and waiting ...
EuropeanServer ready and waiting ...
NorthAmericanServer ready and waiting ...

RM UDP Server is running...
```

Figure : Starting the Replica Manager

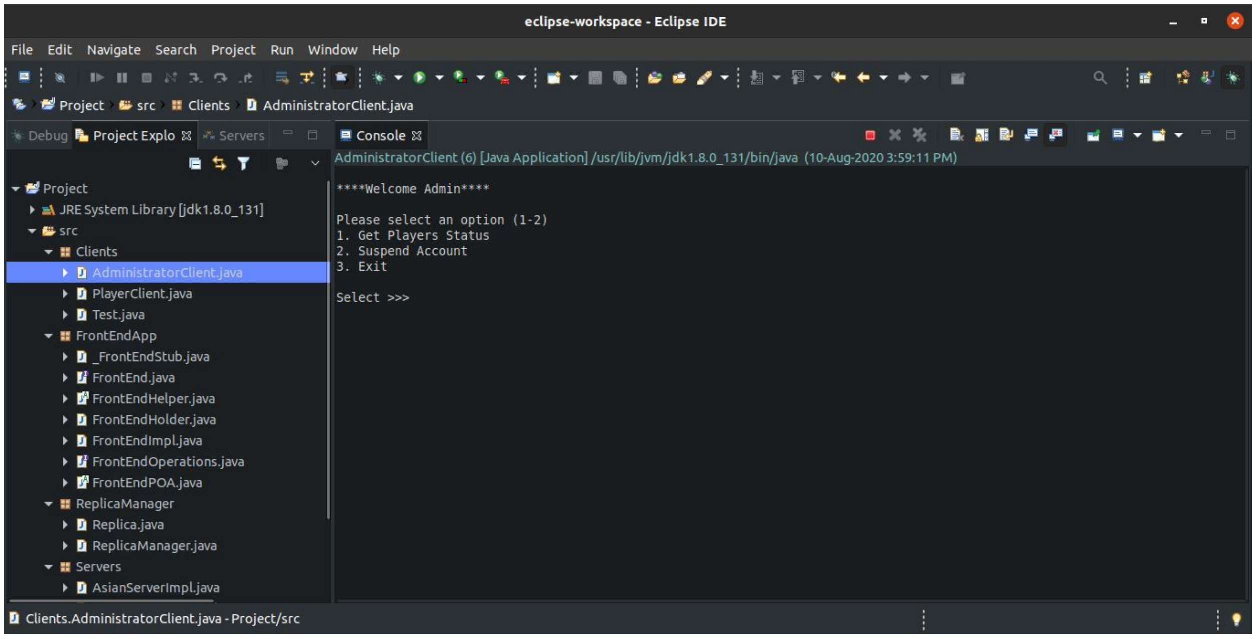
❖ Starting PlayerClient



```
PlayerClient (7) [Java Application] /usr/lib/jvm/jdk1.8.0_131/bin/java (10-Aug-2020 3:58:11 PM)

****Welcome Client****

Please select an option (1-4)
1. Create an Account
2. SignIn
3. SignOut
4. Transfer Account
5. Exit

Select >>>
```

Figure : Starting the PlayerClient

❖ Staring AdministratorClient



```
AdministratorClient (6) [Java Application] /usr/lib/jvm/jdk1.8.0_131/bin/java (10-Aug-2020 3:59:11 PM)
****Welcome Admin****

Please select an option (1-2)
1. Get Players Status
2. Suspend Account
3. Exit

Select >>>
```

Figure : Starting the Admin Client

❖ Logs

Logs folder will be created in the eclipse project folder. Here, you can see “logs” folder with “src” and “bin” folder

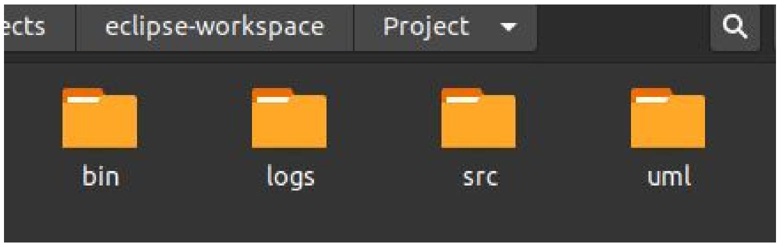


Figure : Project folder structure

There are 3 folders inside “logs” folder

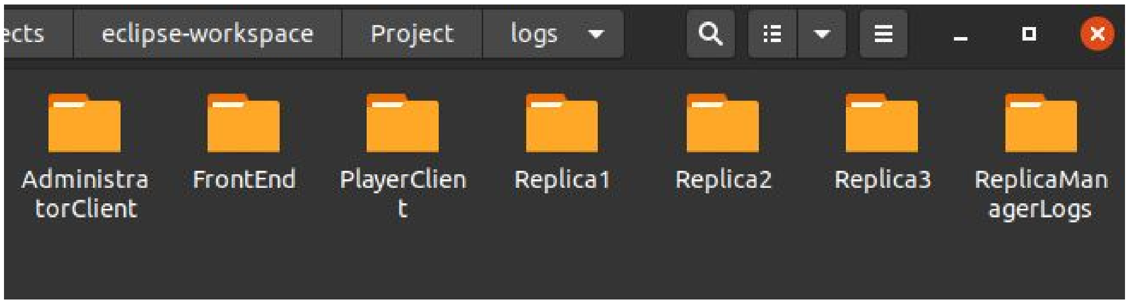


Figure : Logs folder structure

Inside the "ServerLogs" folder there is a separate folder for each server that contains the logs for individual servers and their users.

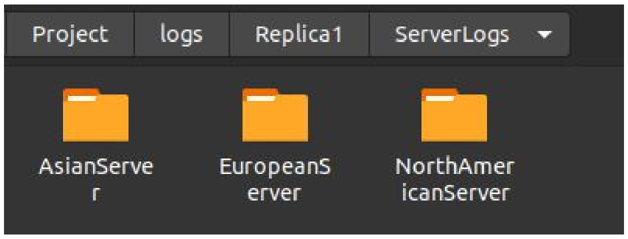


Figure : ServerLogs folder structure

Server's log file will be stored outside and Admin's log are stored in "AdminLogs" folder and users' logs will be stored in "UserLogs" folder

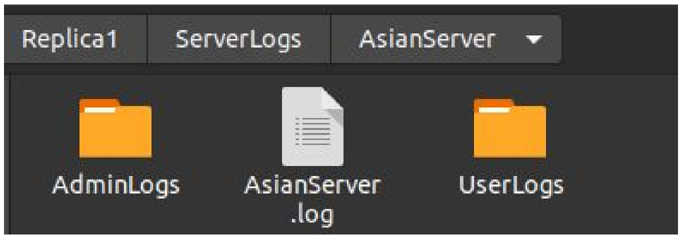


Figure : AsianServer folder structure

Here, as you can see there are 5 folders for 5 different users (by username) each folder contains individual log files for each user

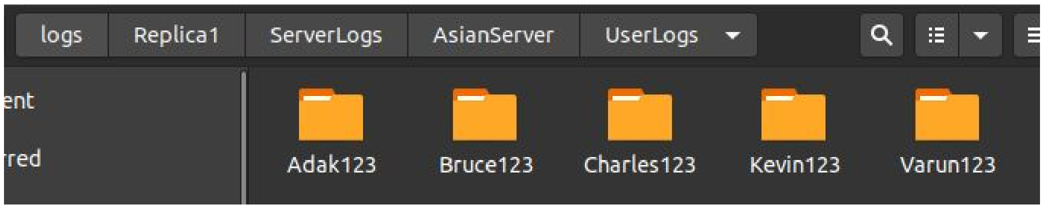


Figure : UserLogs folder structure

The FrontEnd logs will be also stored under the "logs" folder in Project.

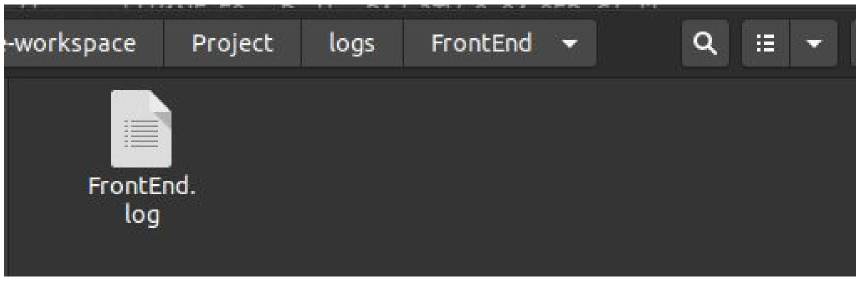


Figure : FrontEnd log folder

The ReplicaManager's log will be stored in ReplicaManagerLogs.

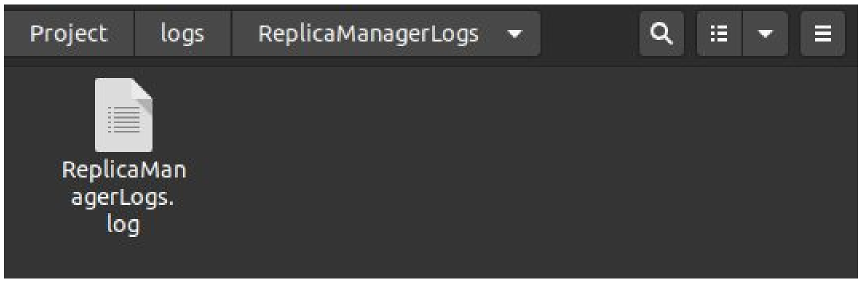


Figure : Replica Manager log folder

Each Replica will create its own folder where it will store all logs which are in part of that replica. Here, as you can see the demo for "Replica 1" which is also known as leader replica.

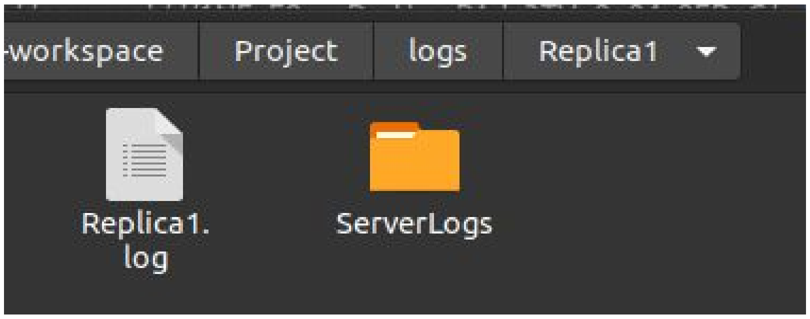


Figure : Replica folder

User log for a player with username "Varun123".

- Feel free to contact me if you do not understand anything or you have any questions. My email address is vasu.ratanpara@mail.concordia.ca

Thank You