Concordia University

Engineering and Computer Science

COMP 6231

Distributed Systems Design

Assignment-1 Report on

Distributed Player Status System (DPSS) using Java RMI

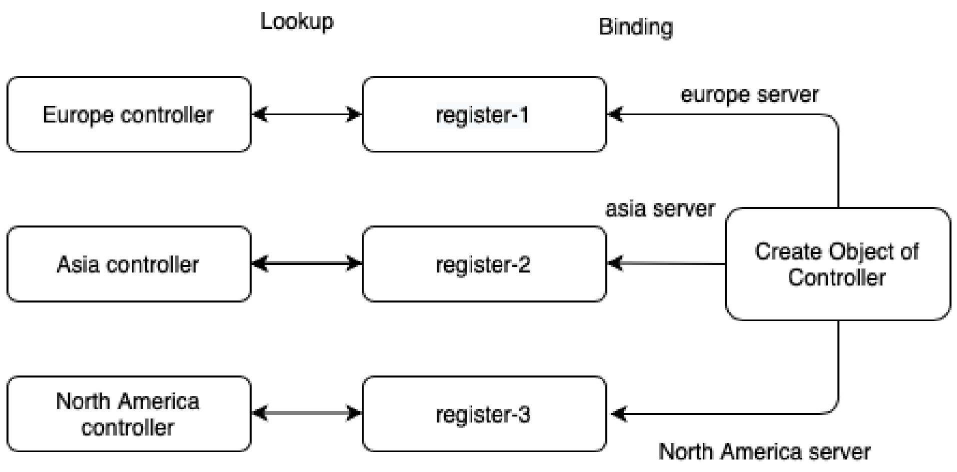
Submitted To: Prof. M. Taleb

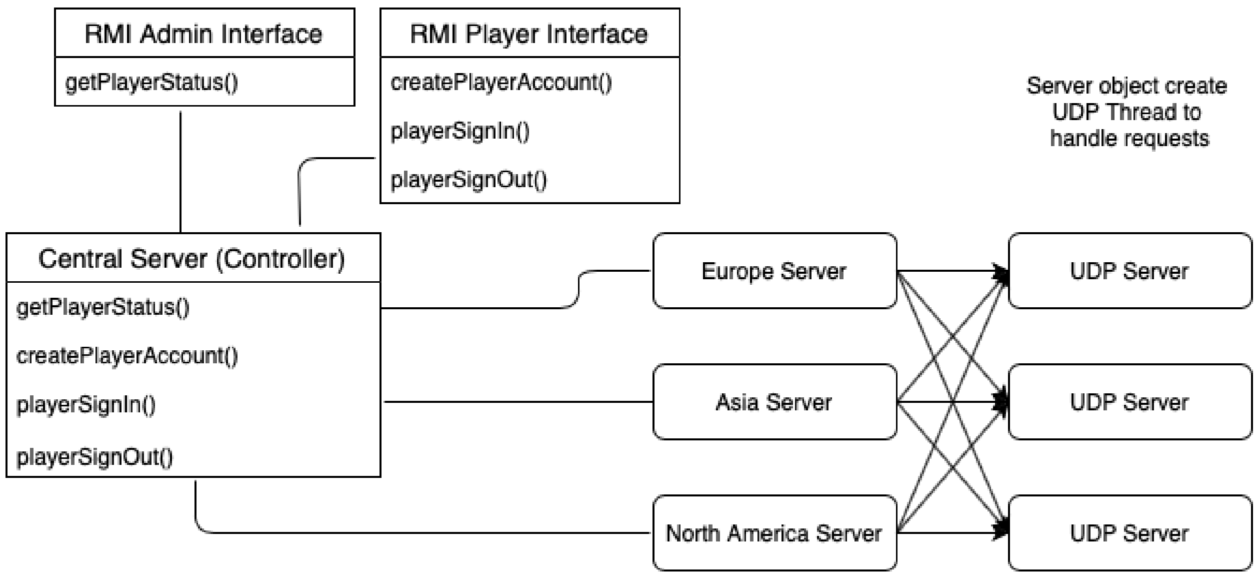
Yash Jayeshkumar Pandya (40119272)

Description:

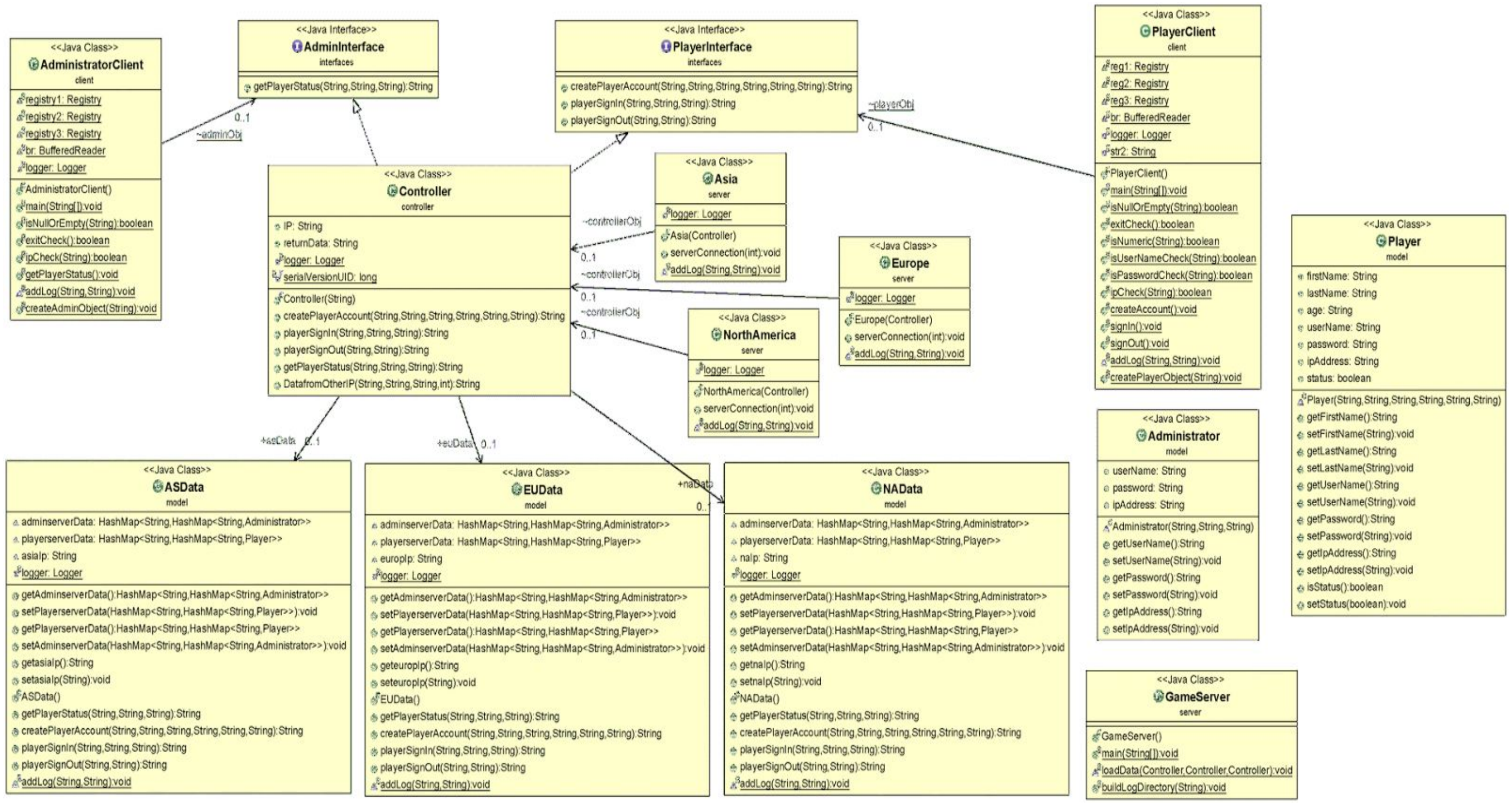
Distributed player status system (DPSS) is implemented as a distributed system to register players across multiple servers and signin and signout from those servers by the players. Additionally, each server has an admin user that can check how many players are online and offline in all the servers. The system is built using Java RMI and the players can see a single system handling requests providing location and access transparency. It also manages simultaneous requests with adequate synchronization.

Design Architecture:





Class Diagram:


<<Java Class>>
AdministratorClient
client
registry1: Registry
registry2: Registry
registry3: Registry
br: BufferedReader
logger: Logger
AdministratorClient()
main(String[]):void
isNullOrEmpty(String):boolean
exitCheck():boolean
ipCheck(String):boolean
getPlayerStatus():void
addLog(String,String):void
createAdminObject(String):void
<<Java Interface>>
AdminInterface
interfaces
getPlayerStatus(String,String,String):String
0..1
~adminObj
<<Java Interface>>
PlayerInterface
interfaces
createPlayerAccount(String,String,String,String,String,String):String
playerSignIn(String,String,String):String
playerSignOut(String,String):String
~playerObj
0..1
<<Java Class>>
PlayerClient
client
reg1: Registry
reg2: Registry
reg3: Registry
br: BufferedReader
logger: Logger
str2: String
PlayerClient()
main(String[]):void
isNullOrEmpty(String):boolean
exitCheck():boolean
isNumeric(String):boolean
isUserNameCheck(String):boolean
isPasswordCheck(String):boolean
ipCheck(String):boolean
createAccount():void
signIn():void
signOut():void
addLog(String,String):void
createPlayerObject(String):void
<<Java Class>>
Controller
controller
IP: String
returnData: String
logger: Logger
serialVersionUID: long
Controller(String)
createPlayerAccount(String,String,String,String,String,String):String
playerSignIn(String,String,String):String
playerSignOut(String,String):String
getPlayerStatus(String,String,String):String
DatafromOtherIP(String,String,String,int):String
~controllerObj
0..1
~controllerObj
0..1
~controllerObj
0..1
<<Java Class>>
Asia
server
logger: Logger
Asia(Controller)
serverConnection(int):void
addLog(String,String):void
<<Java Class>>
Europe
server
logger: Logger
Europe(Controller)
serverConnection(int):void
addLog(String,String):void
<<Java Class>>
NorthAmerica
server
logger: Logger
NorthAmerica(Controller)
serverConnection(int):void
addLog(String,String):void
<<Java Class>>
Player
model
firstName: String
lastName: String
age: String
userName: String
password: String
ipAddress: String
status: boolean
Player(String,String,String,String,String,String)
getFirstName():String
setFirstName(String):void
getLastName():String
setLastName(String):void
getUserName():String
setUserName(String):void
getPassword():String
setPassword(String):void
getIpAddress():String
setIpAddress(String):void
isStatus():boolean
setStatus(boolean):void
+asData 0..1
+euData 0..1
+naData 0..1
<<Java Class>>
ASData
model
adminserverData: HashMap<String,HashMap<String,Administrator>>
playerserverData: HashMap<String,HashMap<String,Player>>
asiaIp: String
logger: Logger
getAdminserverData():HashMap<String,HashMap<String,Administrator>>
setPlayerserverData(HashMap<String,HashMap<String,Player>>):void
getPlayerserverData():HashMap<String,HashMap<String,Player>>
setAdminserverData(HashMap<String,HashMap<String,Administrator>>):void
getasiaIp():String
setasiaIp(String):void
ASData()
getPlayerStatus(String,String,String):String
createPlayerAccount(String,String,String,String,String,String):String
playerSignIn(String,String,String):String
playerSignOut(String,String):String
addLog(String,String):void
<<Java Class>>
EUData
model
adminserverData: HashMap<String,HashMap<String,Administrator>>
playerserverData: HashMap<String,HashMap<String,Player>>
europIp: String
logger: Logger
getAdminserverData():HashMap<String,HashMap<String,Administrator>>
setPlayerserverData(HashMap<String,HashMap<String,Player>>):void
getPlayerserverData():HashMap<String,HashMap<String,Player>>
setAdminserverData(HashMap<String,HashMap<String,Administrator>>):void
geteuropIp():String
seteuropIp(String):void
EUData()
getPlayerStatus(String,String,String):String
createPlayerAccount(String,String,String,String,String,String):String
playerSignIn(String,String,String):String
playerSignOut(String,String):String
addLog(String,String):void
<<Java Class>>
NAData
model
adminserverData: HashMap<String,HashMap<String,Administrator>>
playerserverData: HashMap<String,HashMap<String,Player>>
naIp: String
logger: Logger
getAdminserverData():HashMap<String,HashMap<String,Administrator>>
setPlayerserverData(HashMap<String,HashMap<String,Player>>):void
getPlayerserverData():HashMap<String,HashMap<String,Player>>
setAdminserverData(HashMap<String,HashMap<String,Administrator>>):void
getnaIp():String
setnaIp(String):void
NAData()
getPlayerStatus(String,String,String):String
createPlayerAccount(String,String,String,String,String,String):String
playerSignIn(String,String,String):String
playerSignOut(String,String):String
addLog(String,String):void
<<Java Class>>
Administrator
model
userName: String
password: String
ipAddress: String
Administrator(String,String,String)
getUserName():String
setUserName(String):void
getPassword():String
setPassword(String):void
getIpAddress():String
setIpAddress(String):void
<<Java Class>>
GameServer
server
GameServer()
main(String[]):void
loadData(Controller,Controller,Controller):void
buildLogDirectory(String):void

RMI Interfaces:

AdminInterface

- getPlayerStatus(String userName, String password, String ipAddress)

PlayerInterface

- createPlayerAccount(String firstName, String lastName, String age, String userName, String password, String ipAddress)
- playerSignIn(String userName, String password, String ipAddress)
- playerSignOut(String userName, String ipAddress)

RMI Server (Controller):

- The Class implements AdminInterface and PlayerInterface as RMI interfaces.
- Each ip creates objects of RMI server implementation EU (Europe), NA (NorthAmerica), AS (Asia)

RMI Registry:

Instances of Controller is bound with the 3 different registries with three different strings to expose the instances to the admin and players.

- registry1.bind("North America", northamerica);
- registry2.bind("Europe", europe);
- registry3.bind("Asia", asia);

Data Models:

Hashmap is used as a data model. Hashmap contains an alphabet as a key to store all data that begins with the same username character. Value of the hashmap is another hashmap which contains the username as a key and player object as a value.

Ie. HashMap<String, HashMap<String, Player>>

Logs:

To perform logging for investigating, we have used the java.util.logging logger.

Log Format:

Each log data contains below details:

- Timestamp of the request
- Type of the feature. Ie. createplayer/signin/signout/getplayerstatus
- Parameter. Ie. username
- Message. Ie. success/failure

IP Server:

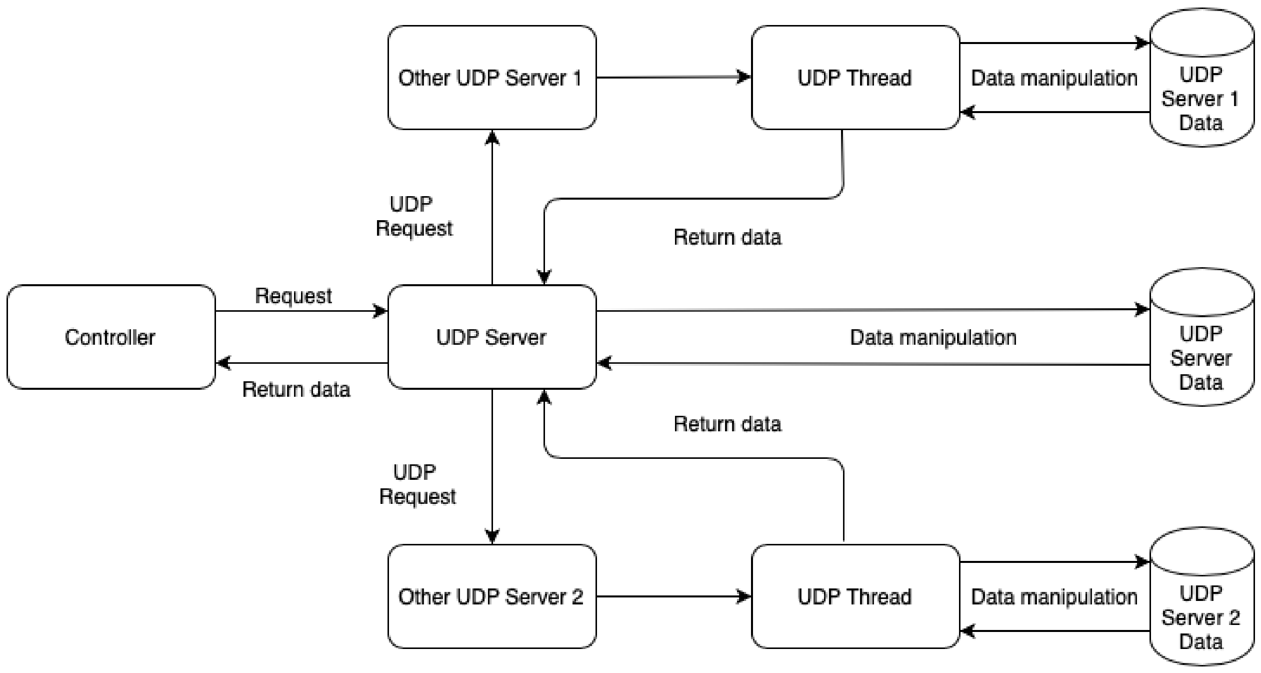
Each server log will be saved in their respective file

- logs/AS.log
- logs/NA.log
- logs/EU.log

Player/Admin Client:

For every action performed by the player or admin, a log file with username is getting created:

UDP Server Design:



Admin Flow:

- The Administratorclient communicates with the controller and sends a request to the respective server based on ip value.
- The server collects the request.
- It forks new requests to send the playerstatus request to the other servers over the UDP.
- The UDP servers receive the request and create new threads to process the request.
- The newly created threads fetches the respective data and responds to the request.
- The server which received the request responds to the controller with appropriate data.

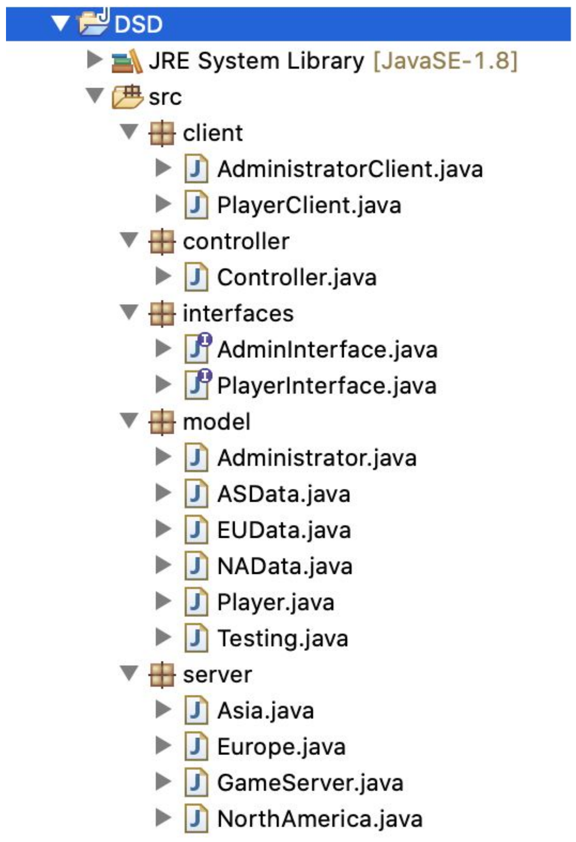
Player Flow:

- The Playerclient communicates with the controller and sends a request to the respective server based on ip value.
- The server collects the request.
- The server which received the request responds to the controller with appropriate data.

Concurrency Control:

The controller creates new thread to communicate to each of servers to handle requests for the same or different function at the same time.

Code Structure:



```
DSD
    JRE System Library [JavaSE-1.8]
    src
        client
            AdministratorClient.java
            PlayerClient.java
        controller
            Controller.java
        interfaces
            AdminInterface.java
            PlayerInterface.java
        model
            Administrator.java
            ASData.java
            EUData.java
            NAData.java
            Player.java
            Testing.java
        server
            Asia.java
            Europe.java
            GameServer.java
            NorthAmerica.java
```

Challenges:

Implementation of synchronization while managing multiple requests at the same time has been challenging.

Run Program:

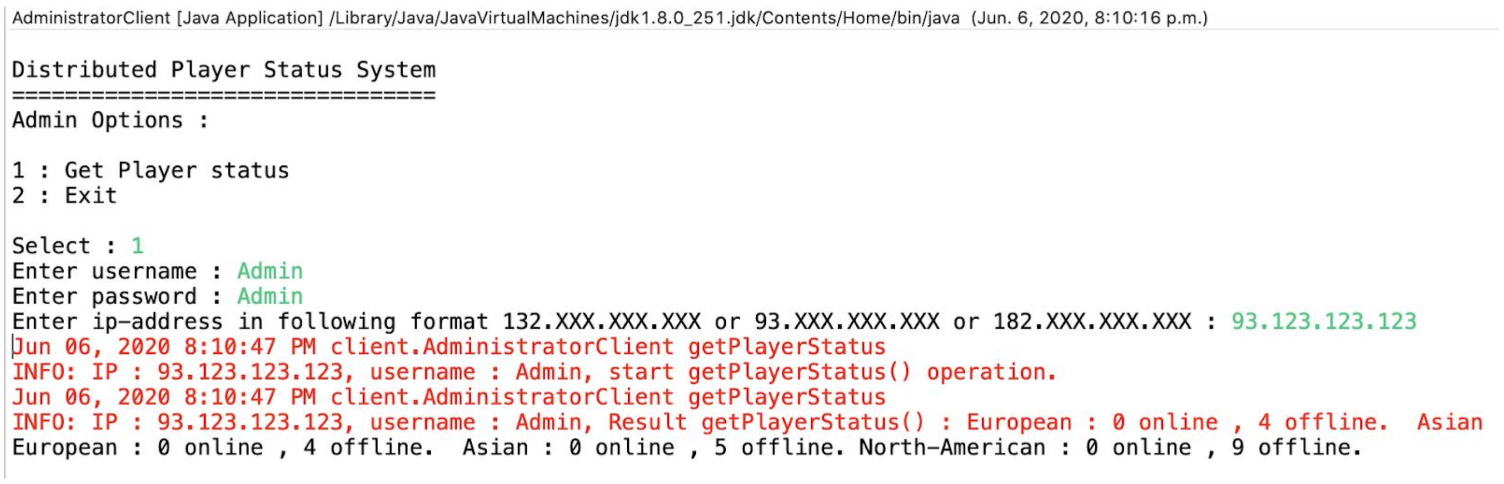
- Run GameServer.java to start the servers
- Run PlayerClient.java to start player client
- Run AdministratorClient.java to start admin client

Test Scenarios:

Before all the tests, a few players are added using data.txt file in each server.

Ie. Northamerica has 9 players loaded, Europe has 4 players loaded and Asia has 5 players loaded.

- **Admin Get Player status to validate online/offline counts**



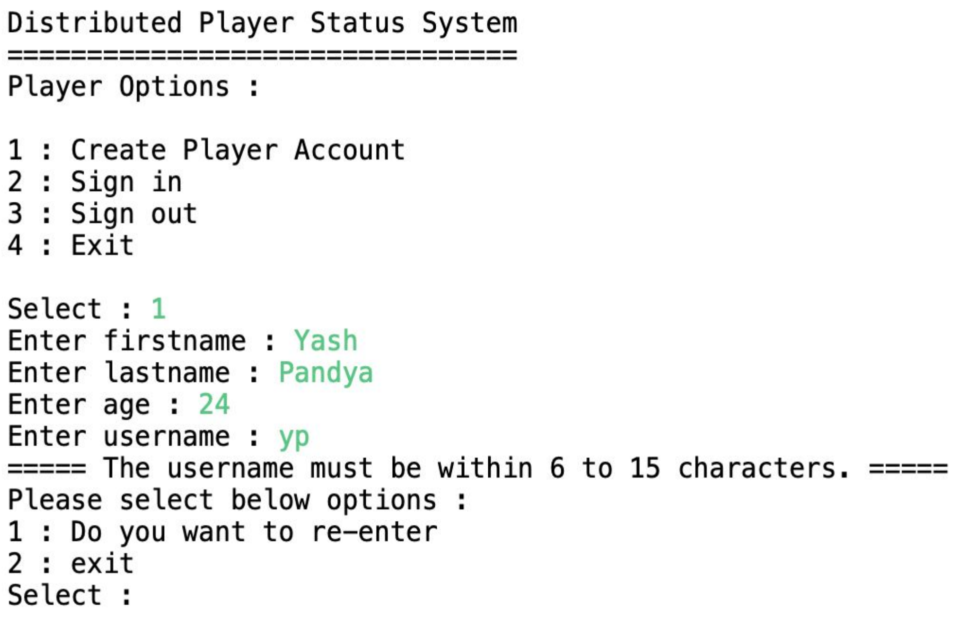
```
AdministratorClient [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_251.jdk/Contents/Home/bin/java  (Jun. 6, 2020, 8:10:16 p.m.)

Distributed Player Status System
================================
Admin Options :

1 : Get Player status
2 : Exit

Select : 1
Enter username : Admin
Enter password : Admin
Enter ip-address in following format 132.XXX.XXX.XXX or 93.XXX.XXX.XXX or 182.XXX.XXX.XXX : 93.123.123.123
Jun 06, 2020 8:10:47 PM client.AdministratorClient getPlayerStatus
INFO: IP : 93.123.123.123, username : Admin, start getPlayerStatus() operation.
Jun 06, 2020 8:10:47 PM client.AdministratorClient getPlayerStatus
INFO: IP : 93.123.123.123, username : Admin, Result getPlayerStatus() : European : 0 online , 4 offline.  Asian
European : 0 online , 4 offline.  Asian : 0 online , 5 offline. North-American : 0 online , 9 offline.
```

- **Create Player to check username validation error**

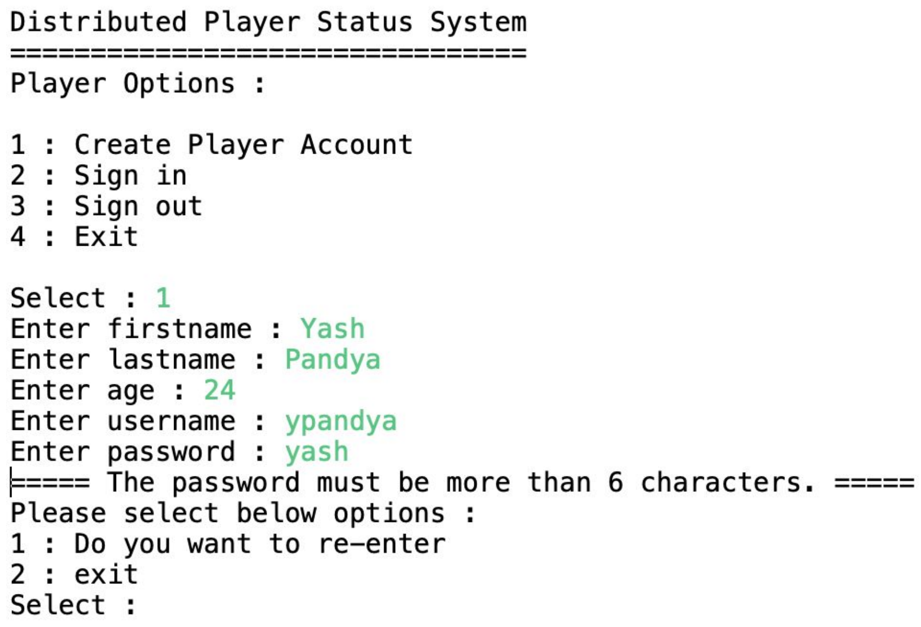


```
Distributed Player Status System
================================
Player Options :

1 : Create Player Account
2 : Sign in
3 : Sign out
4 : Exit

Select : 1
Enter firstname : Yash
Enter lastname : Pandya
Enter age : 24
Enter username : yp
===== The username must be within 6 to 15 characters. =====
Please select below options :
1 : Do you want to re-enter
2 : exit
Select :
```

- **Create Player to check password validation error**

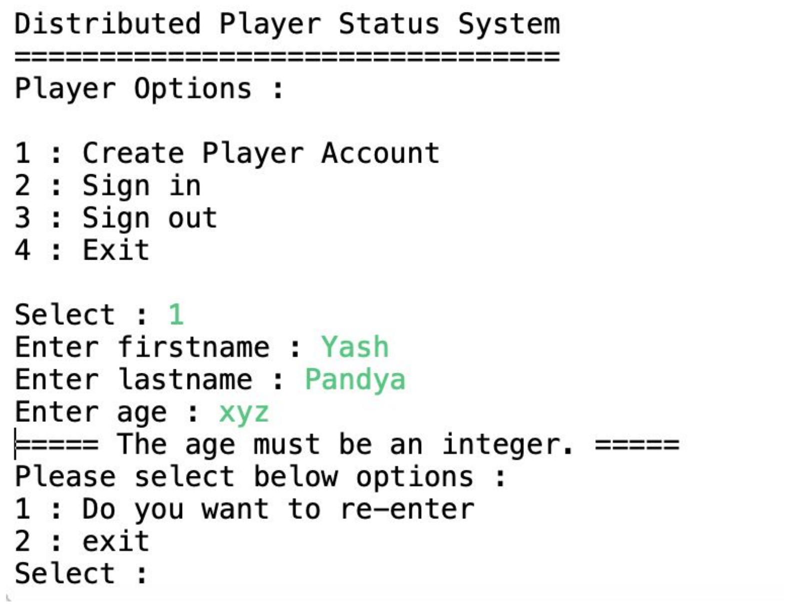


```
Distributed Player Status System
================================
Player Options :

1 : Create Player Account
2 : Sign in
3 : Sign out
4 : Exit

Select : 1
Enter firstname : Yash
Enter lastname : Pandya
Enter age : 24
Enter username : ypandya
Enter password : yash
===== The password must be more than 6 characters. =====
Please select below options :
1 : Do you want to re-enter
2 : exit
Select :
```

- **Create Player to check age validation error**

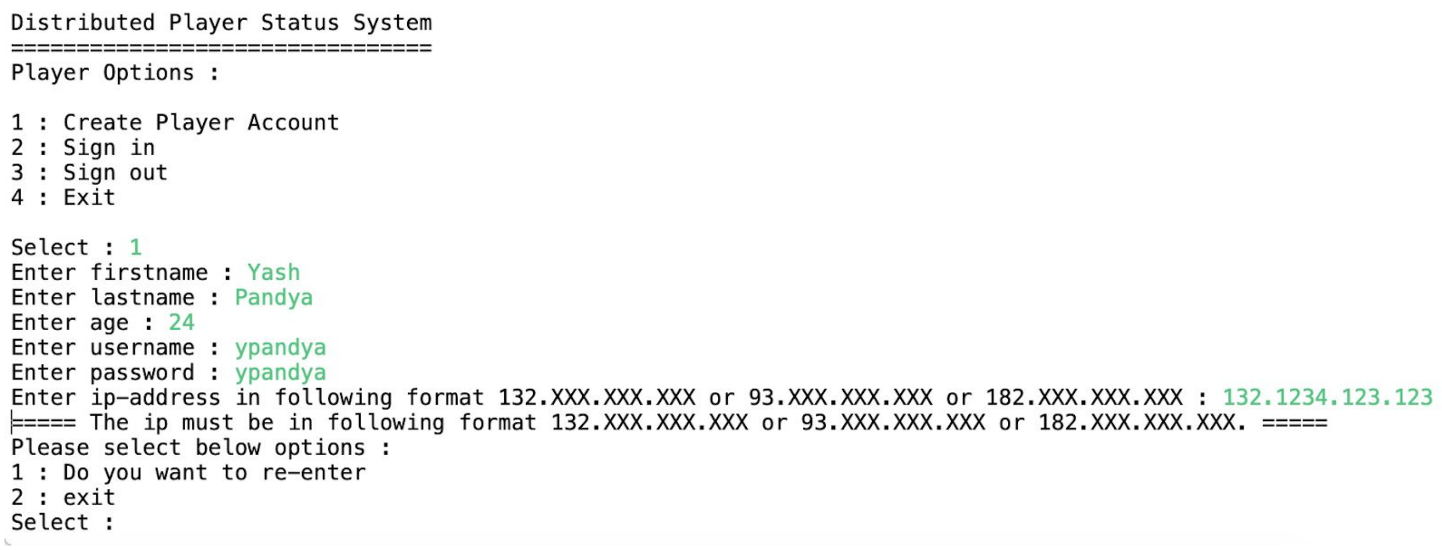


```
Distributed Player Status System
================================
Player Options :

1 : Create Player Account
2 : Sign in
3 : Sign out
4 : Exit

Select : 1
Enter firstname : Yash
Enter lastname : Pandya
Enter age : xyz
===== The age must be an integer. =====
Please select below options :
1 : Do you want to re-enter
2 : exit
Select :
```

- **Create Player to check IP validation error**

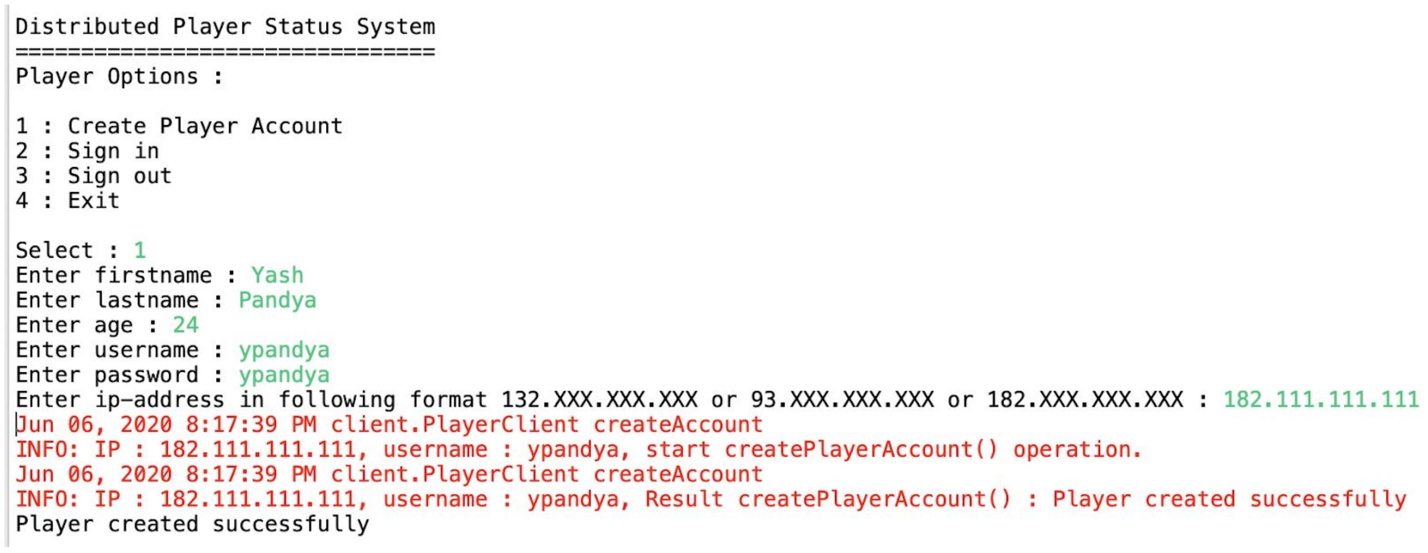


```
Distributed Player Status System
================================
Player Options :

1 : Create Player Account
2 : Sign in
3 : Sign out
4 : Exit

Select : 1
Enter firstname : Yash
Enter lastname : Pandya
Enter age : 24
Enter username : ypandya
Enter password : ypandya
Enter ip-address in following format 132.XXX.XXX.XXX or 93.XXX.XXX.XXX or 182.XXX.XXX.XXX : 132.1234.123.123
===== The ip must be in following format 132.XXX.XXX.XXX or 93.XXX.XXX.XXX or 182.XXX.XXX.XXX. =====
Please select below options :
1 : Do you want to re-enter
2 : exit
Select :
```

- **Create New Player**

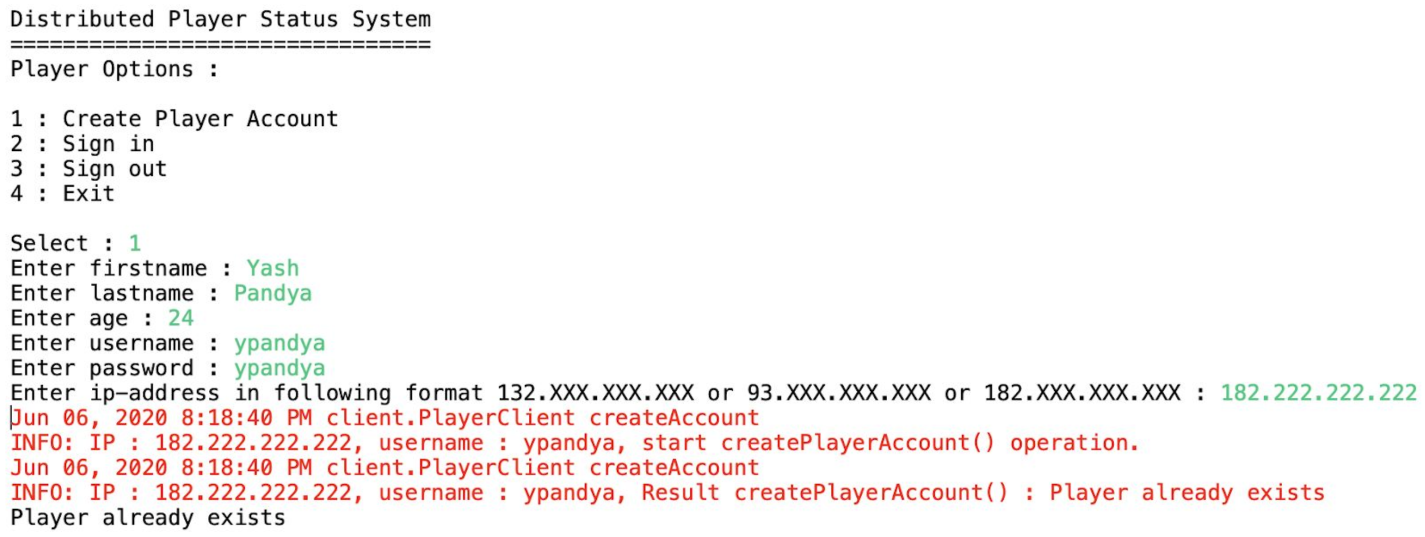


```
Distributed Player Status System
================================
Player Options :

1 : Create Player Account
2 : Sign in
3 : Sign out
4 : Exit

Select : 1
Enter firstname : Yash
Enter lastname : Pandya
Enter age : 24
Enter username : ypandya
Enter password : ypandya
Enter ip-address in following format 132.XXX.XXX.XXX or 93.XXX.XXX.XXX or 182.XXX.XXX.XXX : 182.111.111.111
Jun 06, 2020 8:17:39 PM client.PlayerClient createAccount
INFO: IP : 182.111.111.111, username : ypandya, start createPlayerAccount() operation.
Jun 06, 2020 8:17:39 PM client.PlayerClient createAccount
INFO: IP : 182.111.111.111, username : ypandya, Result createPlayerAccount() : Player created successfully
Player created successfully
```

- **Create Player to validate existing user error**

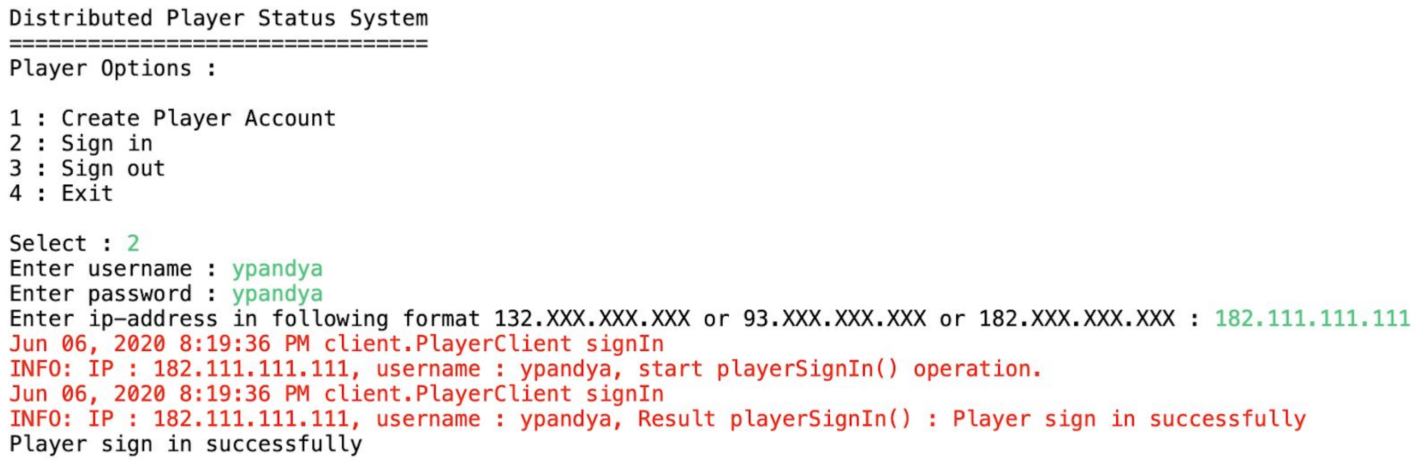


```
Distributed Player Status System
================================
Player Options :

1 : Create Player Account
2 : Sign in
3 : Sign out
4 : Exit

Select : 1
Enter firstname : Yash
Enter lastname : Pandya
Enter age : 24
Enter username : ypandya
Enter password : ypandya
Enter ip-address in following format 132.XXX.XXX.XXX or 93.XXX.XXX.XXX or 182.XXX.XXX.XXX : 182.222.222.222
Jun 06, 2020 8:18:40 PM client.PlayerClient createAccount
INFO: IP : 182.222.222.222, username : ypandya, start createPlayerAccount() operation.
Jun 06, 2020 8:18:40 PM client.PlayerClient createAccount
INFO: IP : 182.222.222.222, username : ypandya, Result createPlayerAccount() : Player already exists
Player already exists
```

- **Player Sign in**

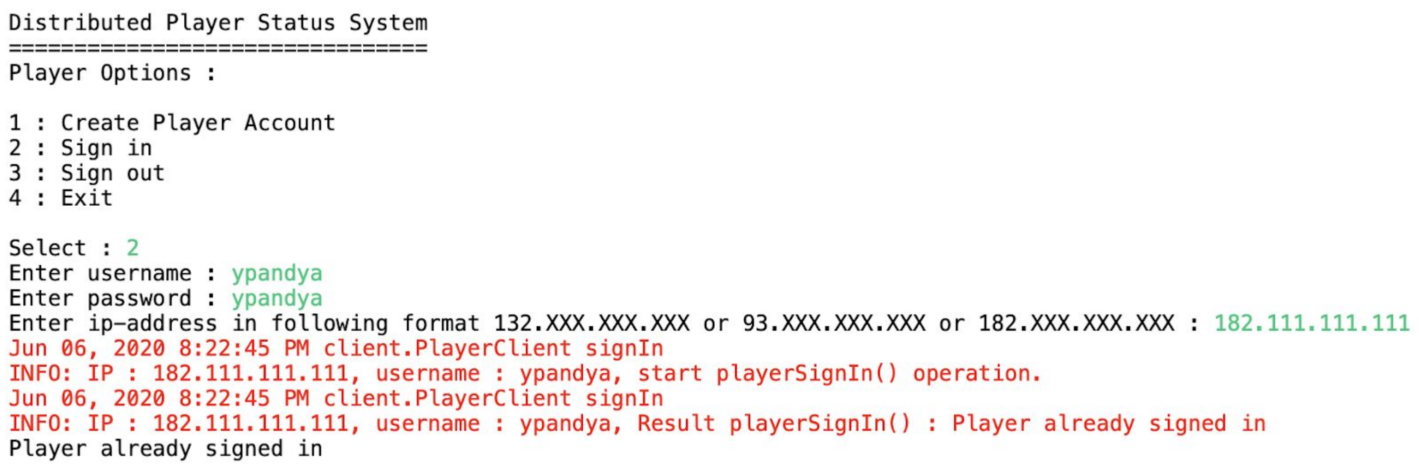


```
Distributed Player Status System
================================
Player Options :

1 : Create Player Account
2 : Sign in
3 : Sign out
4 : Exit

Select : 2
Enter username : ypandya
Enter password : ypandya
Enter ip-address in following format 132.XXX.XXX.XXX or 93.XXX.XXX.XXX or 182.XXX.XXX.XXX : 182.111.111.111
Jun 06, 2020 8:19:36 PM client.PlayerClient signIn
INFO: IP : 182.111.111.111, username : ypandya, start playerSignIn() operation.
Jun 06, 2020 8:19:36 PM client.PlayerClient signIn
INFO: IP : 182.111.111.111, username : ypandya, Result playerSignIn() : Player sign in successfully
Player sign in successfully
```

- **Player Sign in to validate already signin error**

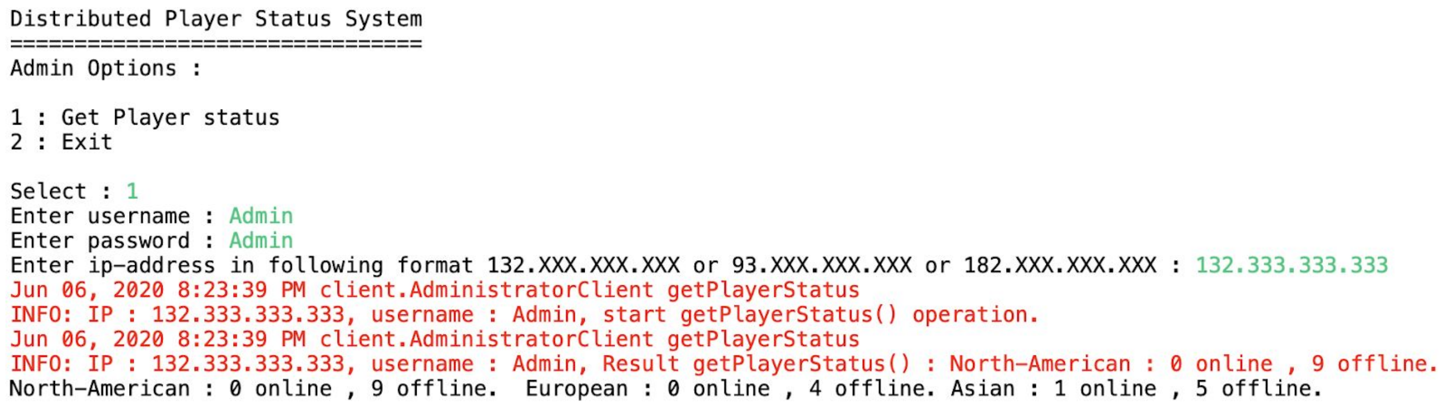


```
Distributed Player Status System
================================
Player Options :

1 : Create Player Account
2 : Sign in
3 : Sign out
4 : Exit

Select : 2
Enter username : ypandya
Enter password : ypandya
Enter ip-address in following format 132.XXX.XXX.XXX or 93.XXX.XXX.XXX or 182.XXX.XXX.XXX : 182.111.111.111
Jun 06, 2020 8:22:45 PM client.PlayerClient signIn
INFO: IP : 182.111.111.111, username : ypandya, start playerSignIn() operation.
Jun 06, 2020 8:22:45 PM client.PlayerClient signIn
INFO: IP : 182.111.111.111, username : ypandya, Result playerSignIn() : Player already signed in
Player already signed in
```

- **Admin Get Player status to validate online/offline counts**

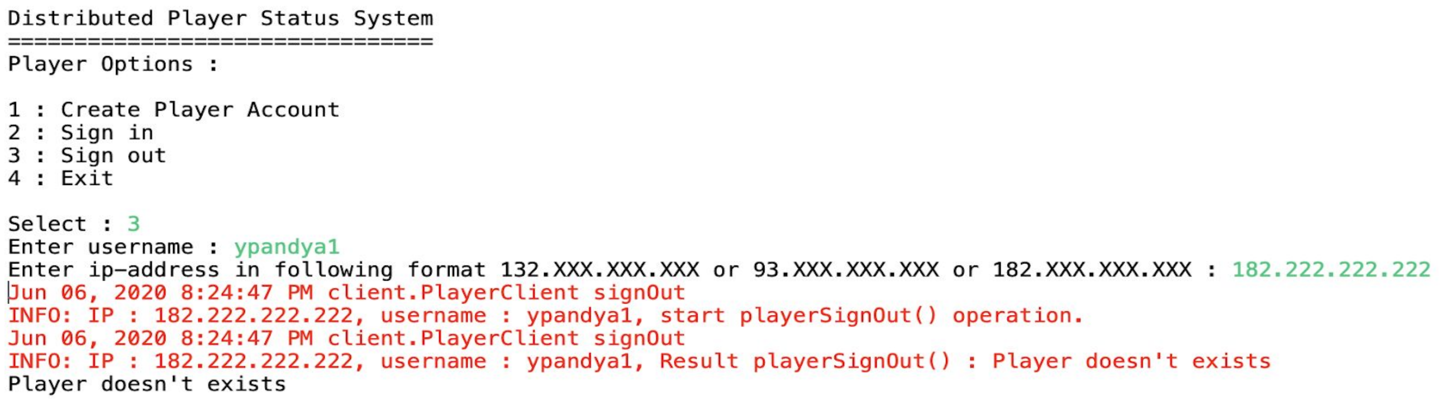


```
Distributed Player Status System
================================
Admin Options :

1 : Get Player status
2 : Exit

Select : 1
Enter username : Admin
Enter password : Admin
Enter ip-address in following format 132.XXX.XXX.XXX or 93.XXX.XXX.XXX or 182.XXX.XXX.XXX : 132.333.333.333
Jun 06, 2020 8:23:39 PM client.AdministratorClient getPlayerStatus
INFO: IP : 132.333.333.333, username : Admin, start getPlayerStatus() operation.
Jun 06, 2020 8:23:39 PM client.AdministratorClient getPlayerStatus
INFO: IP : 132.333.333.333, username : Admin, Result getPlayerStatus() : North-American : 0 online , 9 offline.
North-American : 0 online , 9 offline.  European : 0 online , 4 offline. Asian : 1 online , 5 offline.
```

- **Player Sign out to validate username/ip error**

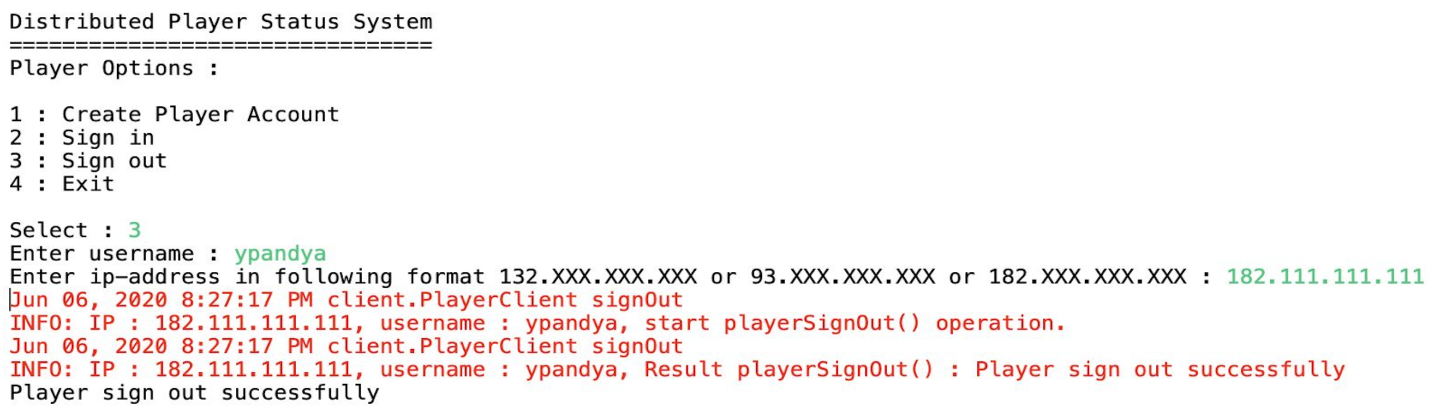


```
Distributed Player Status System
================================
Player Options :

1 : Create Player Account
2 : Sign in
3 : Sign out
4 : Exit

Select : 3
Enter username : ypandya1
Enter ip-address in following format 132.XXX.XXX.XXX or 93.XXX.XXX.XXX or 182.XXX.XXX.XXX : 182.222.222.222
Jun 06, 2020 8:24:47 PM client.PlayerClient signOut
INFO: IP : 182.222.222.222, username : ypandya1, start playerSignOut() operation.
Jun 06, 2020 8:24:47 PM client.PlayerClient signOut
INFO: IP : 182.222.222.222, username : ypandya1, Result playerSignOut() : Player doesn't exists
Player doesn't exists
```

- **Player Sign out**

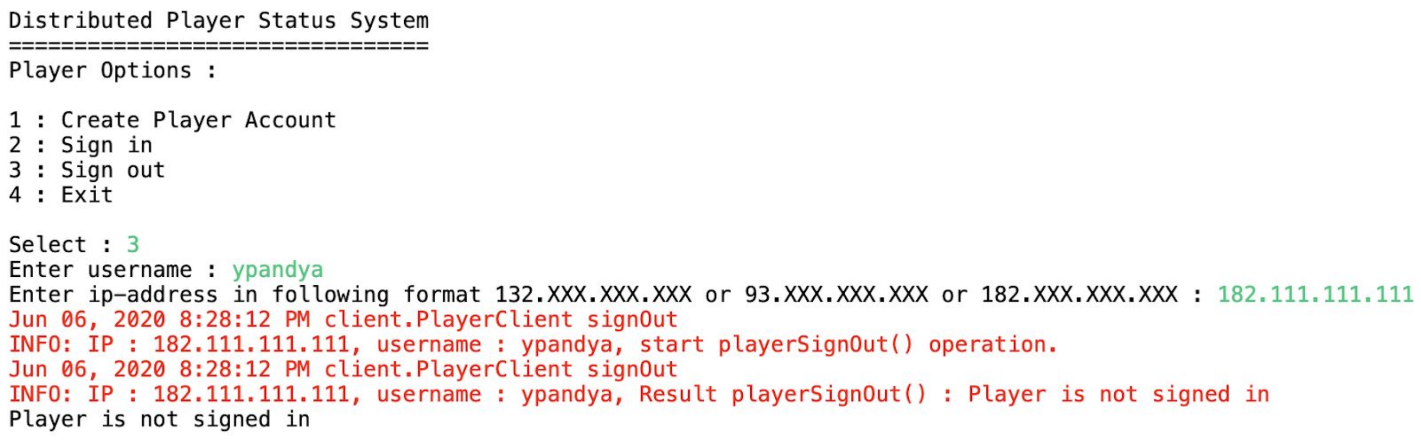


```
Distributed Player Status System
================================
Player Options :

1 : Create Player Account
2 : Sign in
3 : Sign out
4 : Exit

Select : 3
Enter username : ypandya
Enter ip-address in following format 132.XXX.XXX.XXX or 93.XXX.XXX.XXX or 182.XXX.XXX.XXX : 182.111.111.111
Jun 06, 2020 8:27:17 PM client.PlayerClient signOut
INFO: IP : 182.111.111.111, username : ypandya, start playerSignOut() operation.
Jun 06, 2020 8:27:17 PM client.PlayerClient signOut
INFO: IP : 182.111.111.111, username : ypandya, Result playerSignOut() : Player sign out successfully
Player sign out successfully
```

- **Player Sign out to validate not yet signed in error**

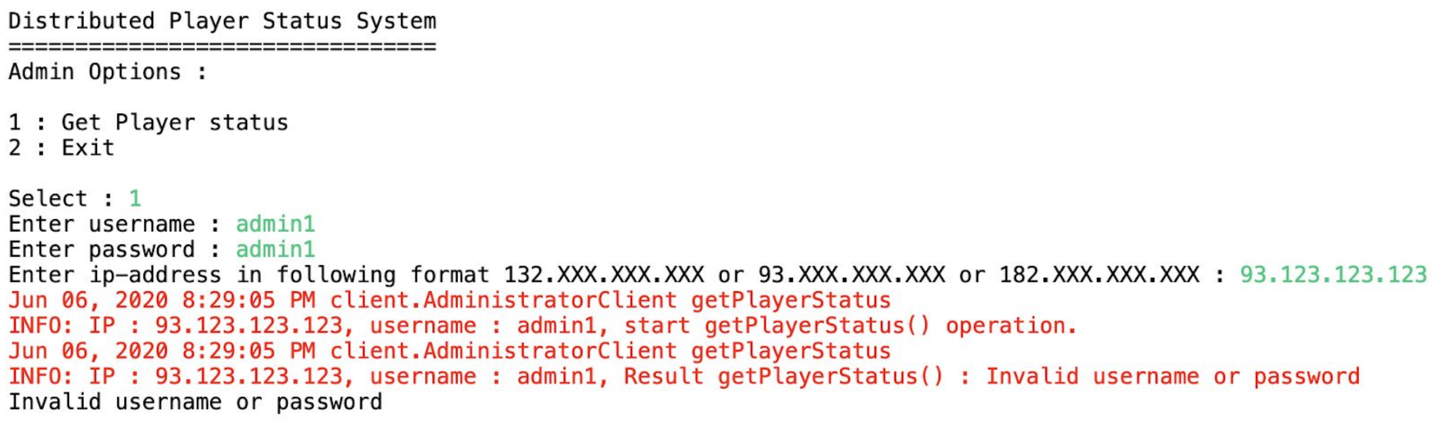


```
Distributed Player Status System
================================
Player Options :

1 : Create Player Account
2 : Sign in
3 : Sign out
4 : Exit

Select : 3
Enter username : ypandya
Enter ip-address in following format 132.XXX.XXX.XXX or 93.XXX.XXX.XXX or 182.XXX.XXX.XXX : 182.111.111.111
Jun 06, 2020 8:28:12 PM client.PlayerClient signOut
INFO: IP : 182.111.111.111, username : ypandya, start playerSignOut() operation.
Jun 06, 2020 8:28:12 PM client.PlayerClient signOut
INFO: IP : 182.111.111.111, username : ypandya, Result playerSignOut() : Player is not signed in
Player is not signed in
```

- **Admin Get Player Status to validate username and password**

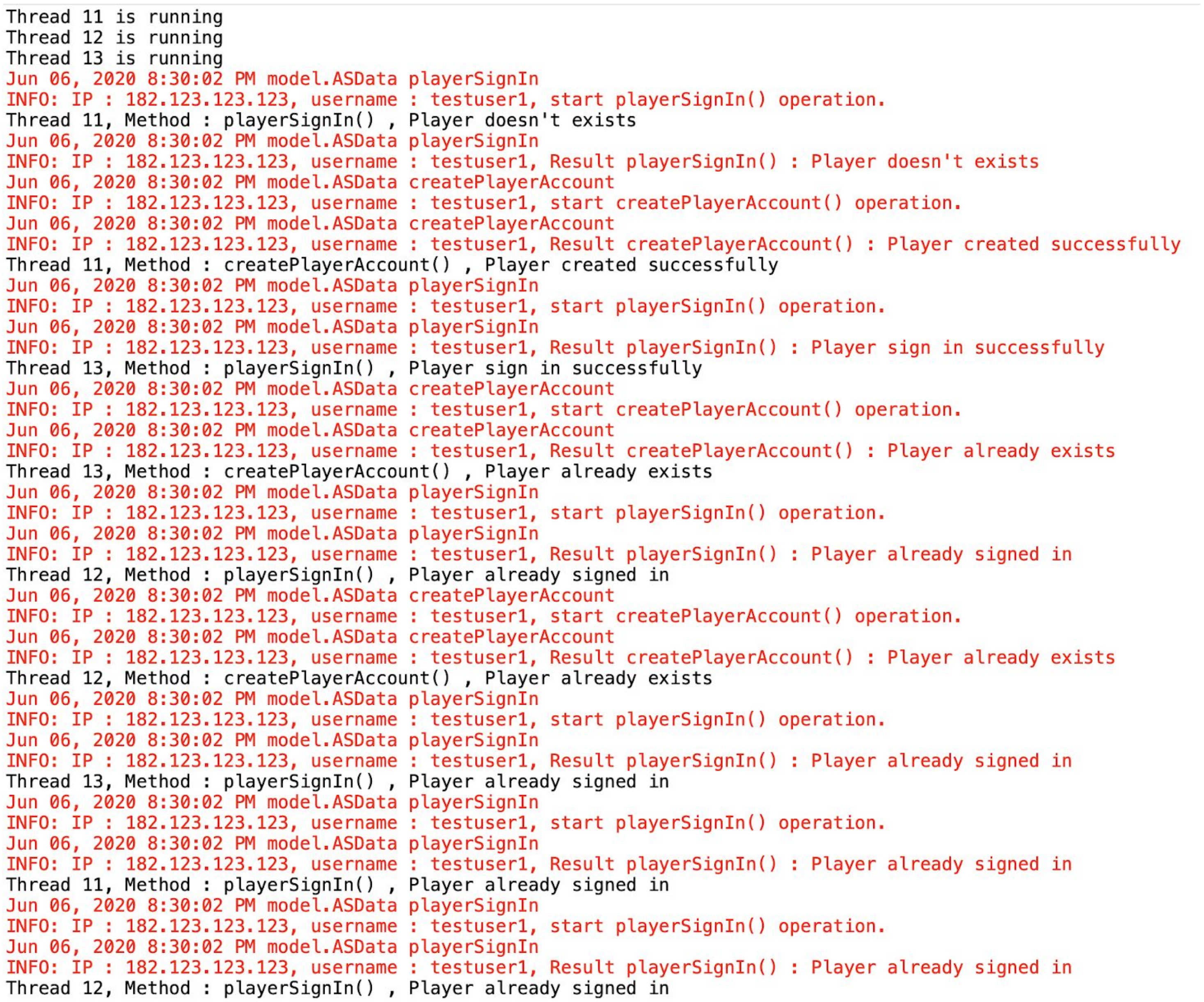


```
Distributed Player Status System
================================
Admin Options :

1 : Get Player status
2 : Exit

Select : 1
Enter username : admin1
Enter password : admin1
Enter ip-address in following format 132.XXX.XXX.XXX or 93.XXX.XXX.XXX or 182.XXX.XXX.XXX : 93.123.123.123
Jun 06, 2020 8:29:05 PM client.AdministratorClient getPlayerStatus
INFO: IP : 93.123.123.123, username : admin1, start getPlayerStatus() operation.
Jun 06, 2020 8:29:05 PM client.AdministratorClient getPlayerStatus
INFO: IP : 93.123.123.123, username : admin1, Result getPlayerStatus() : Invalid username or password
Invalid username or password
```

- **Concurrency validation with one player trying to create and sign in parallely with multiple threads. This test ensures correctness of synchronization. Testing.java file contains a multi threading test case to validate synchronization. 3 threads are trying to create and sign in the same player at same time and synchronization allows concurrent access for create/access.**



```
Thread 11 is running
Thread 12 is running
Thread 13 is running
Jun 06, 2020 8:30:02 PM model.ASData playerSignIn
INFO: IP : 182.123.123.123, username : testuser1, start playerSignIn() operation.
Thread 11, Method : playerSignIn() , Player doesn't exists
Jun 06, 2020 8:30:02 PM model.ASData playerSignIn
INFO: IP : 182.123.123.123, username : testuser1, Result playerSignIn() : Player doesn't exists
Jun 06, 2020 8:30:02 PM model.ASData createPlayerAccount
INFO: IP : 182.123.123.123, username : testuser1, start createPlayerAccount() operation.
Jun 06, 2020 8:30:02 PM model.ASData createPlayerAccount
INFO: IP : 182.123.123.123, username : testuser1, Result createPlayerAccount() : Player created successfully
Thread 11, Method : createPlayerAccount() , Player created successfully
Jun 06, 2020 8:30:02 PM model.ASData playerSignIn
INFO: IP : 182.123.123.123, username : testuser1, start playerSignIn() operation.
Jun 06, 2020 8:30:02 PM model.ASData playerSignIn
INFO: IP : 182.123.123.123, username : testuser1, Result playerSignIn() : Player sign in successfully
Thread 13, Method : playerSignIn() , Player sign in successfully
Jun 06, 2020 8:30:02 PM model.ASData createPlayerAccount
INFO: IP : 182.123.123.123, username : testuser1, start createPlayerAccount() operation.
Jun 06, 2020 8:30:02 PM model.ASData createPlayerAccount
INFO: IP : 182.123.123.123, username : testuser1, Result createPlayerAccount() : Player already exists
Thread 13, Method : createPlayerAccount() , Player already exists
Jun 06, 2020 8:30:02 PM model.ASData playerSignIn
INFO: IP : 182.123.123.123, username : testuser1, start playerSignIn() operation.
Jun 06, 2020 8:30:02 PM model.ASData playerSignIn
INFO: IP : 182.123.123.123, username : testuser1, Result playerSignIn() : Player already signed in
Thread 12, Method : playerSignIn() , Player already signed in
Jun 06, 2020 8:30:02 PM model.ASData createPlayerAccount
INFO: IP : 182.123.123.123, username : testuser1, start createPlayerAccount() operation.
Jun 06, 2020 8:30:02 PM model.ASData createPlayerAccount
INFO: IP : 182.123.123.123, username : testuser1, Result createPlayerAccount() : Player already exists
Thread 12, Method : createPlayerAccount() , Player already exists
Jun 06, 2020 8:30:02 PM model.ASData playerSignIn
INFO: IP : 182.123.123.123, username : testuser1, start playerSignIn() operation.
Jun 06, 2020 8:30:02 PM model.ASData playerSignIn
INFO: IP : 182.123.123.123, username : testuser1, Result playerSignIn() : Player already signed in
Thread 13, Method : playerSignIn() , Player already signed in
Jun 06, 2020 8:30:02 PM model.ASData playerSignIn
INFO: IP : 182.123.123.123, username : testuser1, start playerSignIn() operation.
Jun 06, 2020 8:30:02 PM model.ASData playerSignIn
INFO: IP : 182.123.123.123, username : testuser1, Result playerSignIn() : Player already signed in
Thread 11, Method : playerSignIn() , Player already signed in
Jun 06, 2020 8:30:02 PM model.ASData playerSignIn
INFO: IP : 182.123.123.123, username : testuser1, start playerSignIn() operation.
Jun 06, 2020 8:30:02 PM model.ASData playerSignIn
INFO: IP : 182.123.123.123, username : testuser1, Result playerSignIn() : Player already signed in
Thread 12, Method : playerSignIn() , Player already signed in
```

References:

- https://systembash.com/a-simple-java-udp-server-and-udp-client/
- https://www.tutorialspoint.com/java_rmi/java_rmi_introduction.htm
- https://www.javatpoint.com/RMI
- https://www.geeksforgeeks.org/multithreading-in-java/
- https://www.geeksforgeeks.org/synchronized-in-java/