COMP 6231

Distributed Systems Design

Assignment-3 Report

Web Service Implementation of the Distributed Player Status System (DPSS)

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. The transfer player account feature from the already registered server to another server is added for the user type player. Additionally, each server has an admin user that can check how many players are online and offline in all the servers. Suspend player account from the server feature is added for the user type admin. The system is built using Java Web services and the players/admin can see a single system handling requests providing location and access transparency. It also manages simultaneous requests with adequate synchronization with multiple threading architecture approaches.

Design Architecture:

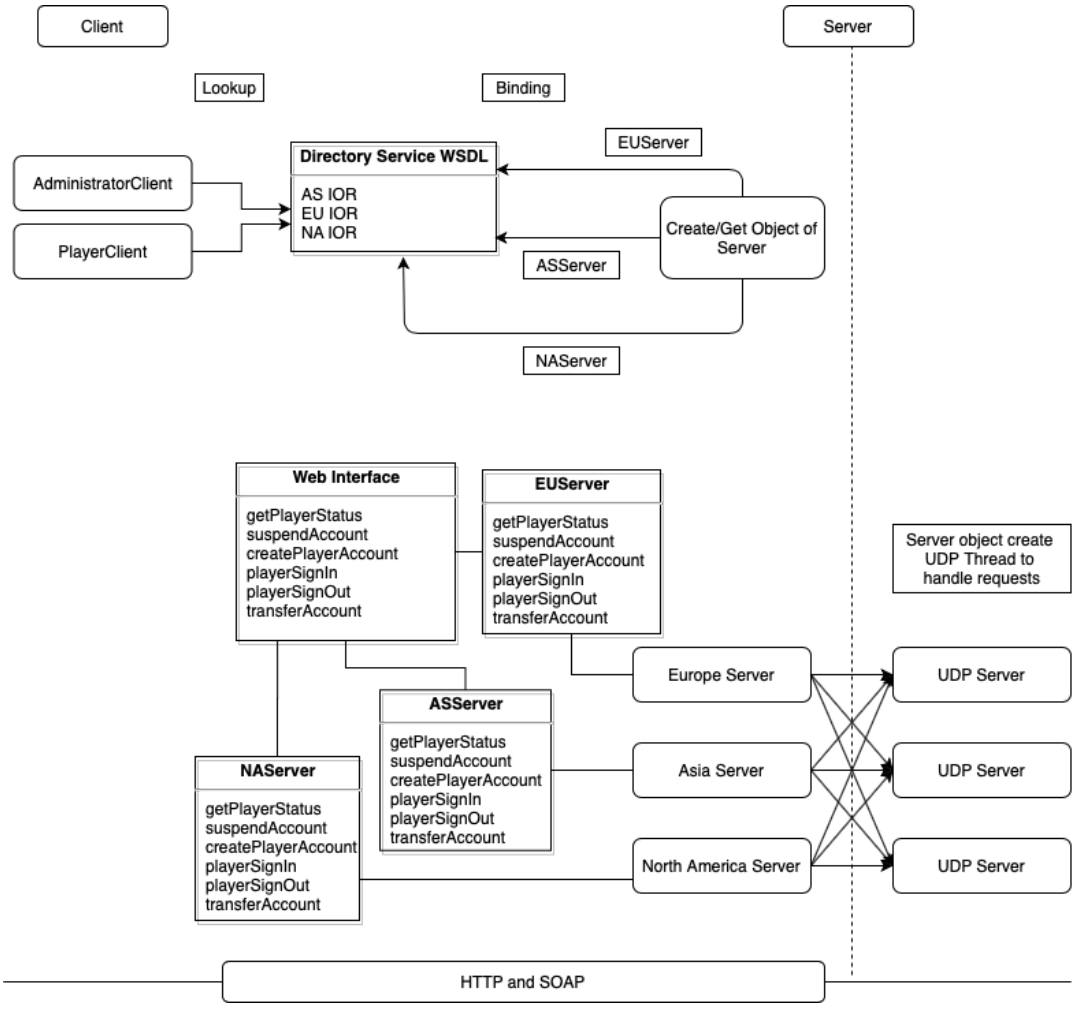


Figure 1: Design Architecture

The above diagram shows the design architecture of the distributed player status system (DPSS). There are mainly 3 different servers for each ip named as Europe (EU), Asian (AS) and North-America (NA). Each server is interconnected with each other using a UDP connection on ports 8880, 8881 and 8882. There is one class called GameServer which creates and initializes the web services for the servers. There are 2 different classes each for different types of user that are going to access the system. AdministratorClient for admin type user and PlayerClient for user type player. The Server classes inherit the methods from the DPSS interface, implement those, create web service methods and it also contains data structures. The Player class contains the attributes and getter setter methods for the Player type of user. Similarly Administrator class for user type Admin. Please note that for admin type user default credential is always “Admin” for username and password.

Class Diagram:

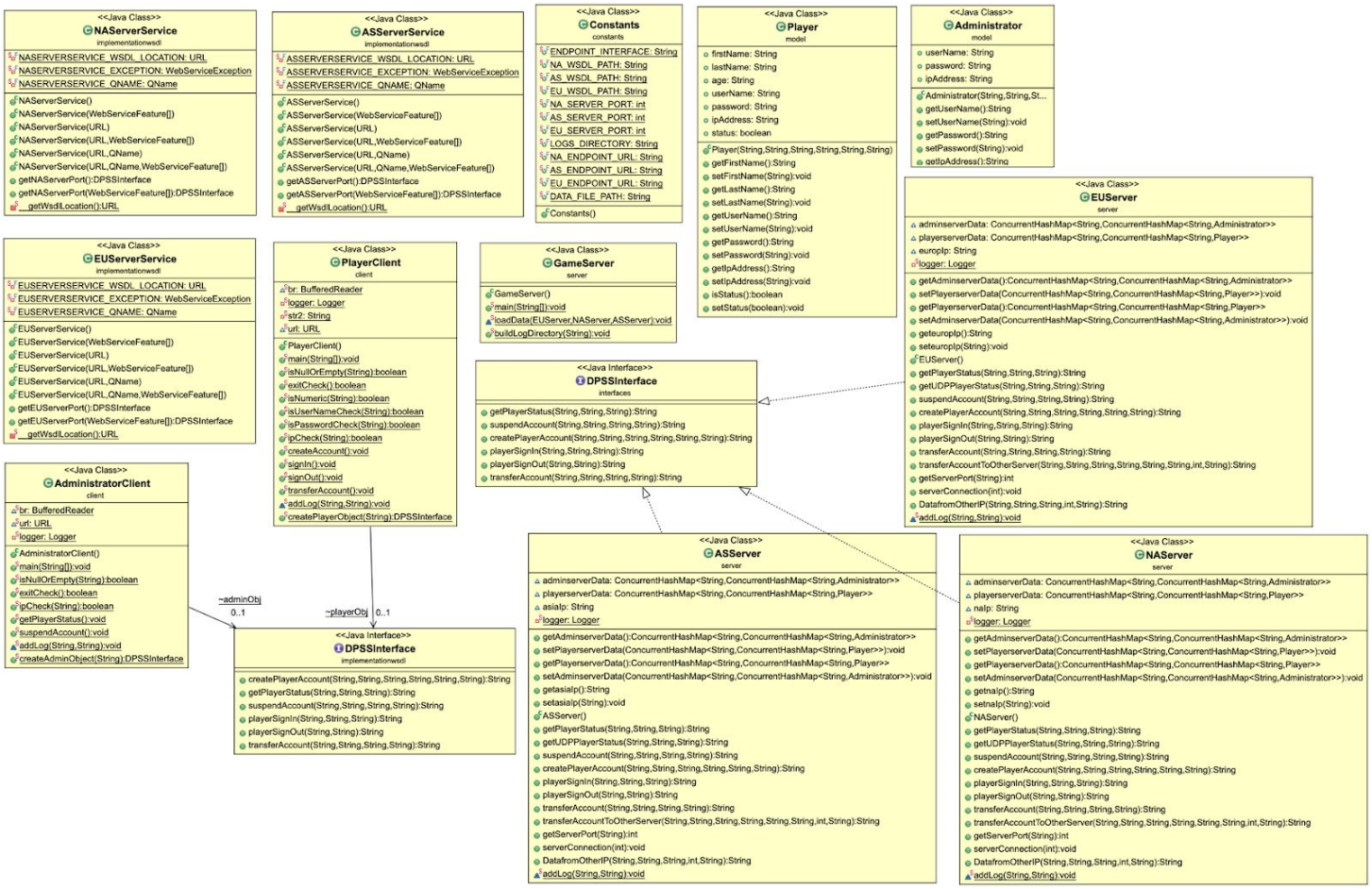


Figure 2: DPSS Class Diagram

AdministratorClient | PlayerClient

- These classes are client classes which interact with the console to get input from respective users and send it to the respective server class. Client classes identify the server based on ip value inserted and call the respective method of the server.

- It also handles the validation on input and ip address formats.

ASServer | NAServer | EUServer

- These classes directly communicate with client classes to get input. Also these classes are actual model classes which contain server wise data structure to store and retrieve information. It also contains the method implementation of the interface. It gets data from the client and after processing sends response data back to the client.

- These classes also contain UDP server interaction with respective servers. UDP servers are used in below cases.
    - Getting player status for admin user type
    - Transferring player account from one server to another

GameServer

- This class is used to create/initiate the endpoints for web services. The server classes objects are mapped, referenced and published with Web services.

- It creates all three server objects and is responsible to start all the three servers.

ASServerService | NAServerService | EUServerService

- These classes are used for mapping of web service implementations from the wsdl files. At runtime based on ip client classes can find respective service servers and call the respective method called by the user to that particular binded server.

- These classes are auto generated from the wsgen and wsimport commands.

Web Service Interface (DPSSInterface) :

- String getPlayerStatus(String userName, String password, String ipAddress)
- String suspendAccount(String AdminUsername, String AdminPassword, String AdminIP, String UsernameToSuspend)
- String createPlayerAccount(String firstName, String lastName, String age, String userName, String password, String ipAddress)
- String playerSignIn(String userName, String password, String ipAddress)
- String playerSignOut(String userName, String ipAddress)
- String transferAccount(String userName, String password, String OldIPAddress, String NewIPAddress)

Web Service Implementation (NAServerService|ASServerService|EUServerService):

A servant is the invocation target containing methods for handling the remote method invocations.

- Web services are implemented by these classes.
- Three instances of Web service implementation are created. One each for IP Server: EU (Europe), AS (Asia), NA (North-America).

Web Service Directory Service (GameServer):

Instances of server classes are published on the local server with different paths for WSDL objects to the client.

Examples of Servers are bound to the Web Services with three unique strings to open the items to the admin and player client.

- Endpoint.publish("http://localhost:8080/DPSS/EU", europe)
- Endpoint.publish("http://localhost:8080/DPSS/NA", northamerica)
- Endpoint.publish("http://localhost:8080/DPSS/AS", asia)

Data Models:

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

Ie. ConcurrentHashMap<String, ConcurrentHashMap<String, Player>>

Ex. username is **testuser**, it will get/create ConcurrentHashMap with key **t** (first character of username). The value of this HasConcurrentHashMapMap is another ConcurrentHashMap which will again get/create a new ConcurrentHashMap with the key **testuser** (actual username) and it will store the **Player class object/instance** in the value of Inner ConcurrentHashMap.

Logs:

To perform logging for investigating, I 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

Admin Flow:

- The Administratorclient communicates with the servers by sending a request to the 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 client with appropriate data.

Player Flow:

- The Playerclient communicates with the servers by sending a request to the respective server based on ip value.
- The server collects the request.
- The server which received the request responds to the client with appropriate data.
- In case of Transfer Player Account operation, It forks new requests to send the transferAccount 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 perform a create operation with respective data and respond to the request.
- The server which received the request responds to the client with appropriate response data.

Concurrency Control:

- The server classes create a new thread to communicate to each of UDP servers to handle requests for the same or different function at the same time.
- Addition to that all the operational methods are synchronized in nature so that it can handle the concurrency with the type of data structure used in DPSS to avoid inconsistency of data.

UDP Server Design:

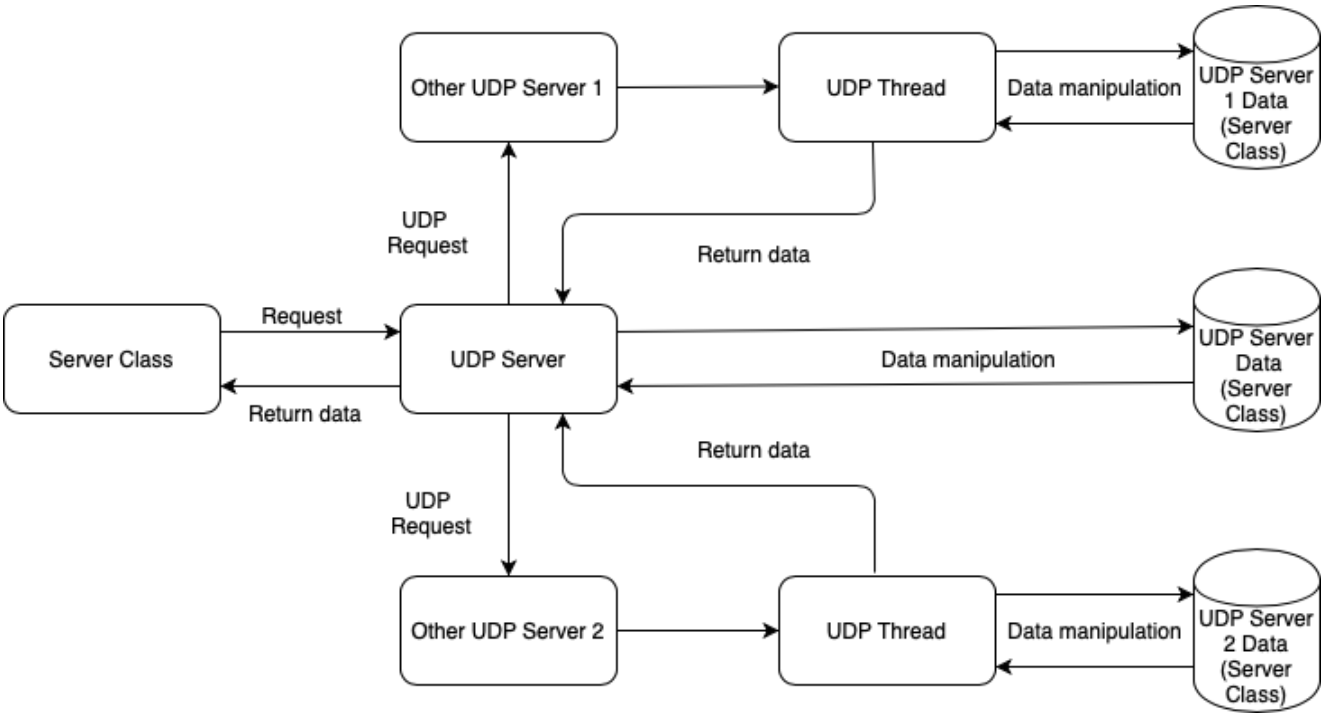


Figure 3: UDP Server Design Diagram

Challenges:

- Implementation of synchronization while managing multiple requests at the same time has been challenging. To implement this I have used synchronized method as this type of method automatically acquires the lock on the shared object such as ConcurrentHashMap in the DPSS and releases it when the method completes its execution.
- These synchronized methods are capable of handling multiple invocations from the numerous threads. I have used multithreading and serialization of methods to achieve concurrency and data integrity.
- Version 1 (RMI) of the system used HashMap, onwards I have used ConcurrentHashMap as ConcurrentHashMap is more synchronized in nature as well as Thread safe compared to HashMap
- **Re-structured the code from Version 1 (RMI) and Version 2 (CORBA) to remove the middleware class (Controller) and three different classes for UDP interaction (Asia, Europe, NorthAmerica). Instead of that, i have converted Data classes into Server classes and UDP class methods into server classes itself ie. AsiaServer.**

Code Structure:

Figure 4: Code Directory Structure

Run Program:

- Run **GameServer.java** to start the server with the below arguments in run configuration.
- Run **PlayerClient.java** to start the player client functionalities with the below arguments
- Run **AdministratorClient.java** to start the admin client functionalities with the below
- Run **Testing.java** to verify the test cases with the below arguments in run configuration. (Need to run GameServer first in order to run test cases)

Testing:

- Testing is divided into three parts.
    1. Console Test Scenarios
    2. Basic Test Scenarios
    3. Advanced/Concurrency Test Scenarios
    4. Web service implementation tested with soapUI tool.

Console Test Scenarios:

- **Before all the UI/Console 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.**
- **Below tests are the UI/Console Interaction Tests. Synchronization Test is added in the advanced tests category.**

- **Admin Get Player status to validate online/offline counts**
    - This test checks the number of player accounts with all servers with their status.

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

1 : Get Player status
2 : Suspend Player Account
3 : 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 : 132.111.222.333
Jun 26, 2020 6:19:16 PM client.AdministratorClient getPlayerStatus
INFO: IP : 132.111.222.333, username : Admin, start getPlayerStatus() operation.
Jun 26, 2020 6:19:16 PM client.AdministratorClient getPlayerStatus
INFO: IP : 132.111.222.333, username : Admin, Result getPlayerStatus() : North American : 0 online , 9 offline.  Europe : 0 online , 4 offline. Asian : 0 online , 5 offline.
North American : 0 online , 9 offline.  Europe : 0 online , 4 offline. Asian : 0 online , 5 offline.
```

- **Create Player to check username validation error**
    - This test checks validation on username while player account creation.

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

1 : Create Player Account
2 : Sign in
3 : Sign out
4 : Transfer Account
5 : 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**
    - This test checks validation on password while player account creation.

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

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

Select : 1
Enter firstname : Yash
Enter lastname : Pandya
Enter age : 24
Enter username : ypandya
Enter password : yp
===== 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**
    - This test checks validation on age while player account creation.

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

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

Select : 1
Enter firstname : Yash
Enter lastname : Pandya
Enter age : text
===== 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**
    - This test checks validation on IP while player account creation.

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

1 : Create Player Account
2 : Sign in
3 : Sign out
4 : Transfer Account
5 : 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 : 132.1234.123.32
===== 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**
    - This test performs the creation of a player account.

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

1 : Create Player Account
2 : Sign in
3 : Sign out
4 : Transfer Account
5 : 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 : 182.123.123.123
Jun 26, 2020 6:24:59 PM client.PlayerClient createAccount
INFO: IP : 182.123.123.123, username : ypandya, start createPlayerAccount() operation.
Jun 26, 2020 6:24:59 PM client.PlayerClient createAccount
INFO: IP : 182.123.123.123, username : ypandya, Result createPlayerAccount() : Player created successfully
Player created successfully
```

- **Create Player to validate existing user error**
    - This test checks validation on player existence while creating the player account.

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

1 : Create Player Account
2 : Sign in
3 : Sign out
4 : Transfer Account
5 : 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 : 182.123.333.444
Jun 26, 2020 6:25:56 PM client.PlayerClient createAccount
INFO: IP : 182.123.333.444, username : ypandya, start createPlayerAccount() operation.
Jun 26, 2020 6:25:56 PM client.PlayerClient createAccount
INFO: IP : 182.123.333.444, username : ypandya, Result createPlayerAccount() : Player already exists
Player already exists
```

- **Player Sign in**
    - This test performs the sign in of a player account.

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

1 : Create Player Account
2 : Sign in
3 : Sign out
4 : Transfer Account
5 : 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 : 182.123.123.123
Jun 26, 2020 6:27:16 PM client.PlayerClient signIn
INFO: IP : 182.123.123.123, username : ypandya, start playerSignIn() operation.
Jun 26, 2020 6:27:16 PM client.PlayerClient signIn
INFO: IP : 182.123.123.123, username : ypandya, Result playerSignIn() : Player sign in successfully
Player sign in successfully
```

- **Player Sign in to validate already signin error**
    - This checks sign in validation when a player trying to sign in again after first sign in is successful.

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

1 : Create Player Account
2 : Sign in
3 : Sign out
4 : Transfer Account
5 : 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 : 182.123.123.123
Jun 26, 2020 6:28:39 PM client.PlayerClient signIn
INFO: IP : 182.123.123.123, username : ypandya, start playerSignIn() operation.
Jun 26, 2020 6:28:39 PM client.PlayerClient signIn
INFO: IP : 182.123.123.123, username : ypandya, Result playerSignIn() : Player already signed in
Player already signed in
```

- **Admin Get Player status to validate online/offline counts**
    - This test checks the number of player accounts with all servers with their status.

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

1 : Get Player status
2 : Suspend Player Account
3 : 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 : 182.132.232.111
Jun 26, 2020 6:29:22 PM client.AdministratorClient getPlayerStatus
INFO: IP : 182.132.232.111, username : Admin, start getPlayerStatus() operation.
Jun 26, 2020 6:29:23 PM client.AdministratorClient getPlayerStatus
INFO: IP : 182.132.232.111, username : Admin, Result getPlayerStatus() : Asian : 1 online , 5 offline.  Europe : 0 online , 4 offline. North American : 0 online , 9 offline.
Asian : 1 online , 5 offline.  Europe : 0 online , 4 offline. North American : 0 online , 9 offline.
```

- **Player Sign out to validate username/ip error**
    - This test checks validation on username/ip error while sign out from the player.

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

1 : Create Player Account
2 : Sign in
3 : Sign out
4 : Transfer Account
5 : 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 : 132.123.321.123
Jun 26, 2020 6:30:40 PM client.PlayerClient signOut
INFO: IP : 132.123.321.123, username : ypandya1, start playerSignOut() operation.
Jun 26, 2020 6:30:40 PM client.PlayerClient signOut
INFO: IP : 132.123.321.123, username : ypandya1, Result playerSignOut() : Player account (ypandya1) doesn't exists
Player account (ypandya1) doesn't exists
```

- **Player Sign out**
  - This test performs the sign out of a player account.

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

1 : Create Player Account
2 : Sign in
3 : Sign out
4 : Transfer Account
5 : 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 : 182.123.123.123
Jun 26, 2020 6:31:17 PM client.PlayerClient signOut
INFO: IP : 182.123.123.123, username : ypandya, start playerSignOut() operation.
Jun 26, 2020 6:31:17 PM client.PlayerClient signOut
INFO: IP : 182.123.123.123, username : ypandya, Result playerSignOut() : Player sign out successfully
Player sign out successfully
```

- **Player Sign out to validate not yet signed in error**
  - This test checks validation on not signed in while sign out player account.

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

1 : Create Player Account
2 : Sign in
3 : Sign out
4 : Transfer Account
5 : 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 : 182.123.123.123
Jun 26, 2020 6:31:49 PM client.PlayerClient signOut
INFO: IP : 182.123.123.123, username : ypandya, start playerSignOut() operation.
Jun 26, 2020 6:31:49 PM client.PlayerClient signOut
INFO: IP : 182.123.123.123, username : ypandya, Result playerSignOut() : Player is not signed in
Player is not signed in
```

- **Admin Get Player Status to validate username and password**
  - This test checks validation on wrong admin username and password.

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

1 : Get Player status
2 : Suspend Player Account
3 : 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 26, 2020 6:32:25 PM client.AdministratorClient getPlayerStatus
INFO: IP : 93.123.123.123, username : Admin1, start getPlayerStatus() operation.
Jun 26, 2020 6:32:25 PM client.AdministratorClient getPlayerStatus
INFO: IP : 93.123.123.123, username : Admin1, Result getPlayerStatus() : Invalid username or password
Invalid username or password
```

- **Admin Suspend Player Account**
    - This test performs suspend player account operation of admin user.

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

1 : Get Player status
2 : Suspend Player Account
3 : Exit

Select : 2
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 : 182.123.123.123
Enter Player username to suspend account : ypandya
Jun 26, 2020 6:33:34 PM client.AdministratorClient suspendAccount
INFO: IP : 182.123.123.123, username : Admin, start suspendAccount() operation.
Jun 26, 2020 6:33:34 PM client.AdministratorClient suspendAccount
INFO: IP : 182.123.123.123, username : Admin, Result suspendAccount() : Player account (ypandya) is suspended
Player account (ypandya) is suspended
```

- **Admin Suspend Player Account with error**
    - This test checks validation on player existence before suspending account.

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

1 : Get Player status
2 : Suspend Player Account
3 : Exit

Select : 2
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 : 182.123.123.123
Enter Player username to suspend account : ypandya
Jun 26, 2020 6:34:31 PM client.AdministratorClient suspendAccount
INFO: IP : 182.123.123.123, username : Admin, start suspendAccount() operation.
Jun 26, 2020 6:34:31 PM client.AdministratorClient suspendAccount
INFO: IP : 182.123.123.123, username : Admin, Result suspendAccount() : Player account (ypandya) doesn't exists
Player account (ypandya) doesn't exists
```

- **Player Transfer Account**
    - This test performs transfer player accounts from one server to another.

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

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

Select : 4
Enter username : ypandya
Enter password : ypandya
Enter Old ip-address in following format 132.XXX.XXX.XXX or 93.XXX.XXX.XXX or 182.XXX.XXX.XXX : 182.123.123.123
Enter New ip-address in following format 132.XXX.XXX.XXX or 93.XXX.XXX.XXX or 182.XXX.XXX.XXX : 93.321.321.321
Jun 26, 2020 6:37:20 PM client.PlayerClient transferAccount
INFO: IP : 182.123.123.123, username : ypandya, start transferAccount() operation.
Jun 26, 2020 6:37:20 PM client.PlayerClient transferAccount
INFO: IP : 182.123.123.123, username : ypandya, Result transferAccount() : Player account (ypandya) is transfered from 182.123.123.123 to 93.321.321.321
Player account (ypandya) is transfered from 182.123.123.123 to 93.321.321.321
```

- **Player Transfer Account with error**
    - This test checks transfer player validation if the transfer server ip is the same as the current server.

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

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

Select : 4
Enter username : ypandya
Enter password : ypandya1
Enter Old ip-address in following format 132.XXX.XXX.XXX or 93.XXX.XXX.XXX or 182.XXX.XXX.XXX : 182.123.123.123
Enter New ip-address in following format 132.XXX.XXX.XXX or 93.XXX.XXX.XXX or 182.XXX.XXX.XXX : 93.123.123.123
Jun 26, 2020 6:35:32 PM client.PlayerClient transferAccount
INFO: IP : 182.123.123.123, username : ypandya, start transferAccount() operation.
Jun 26, 2020 6:35:32 PM client.PlayerClient transferAccount
INFO: IP : 182.123.123.123, username : ypandya, Result transferAccount() : Invalid IP address or Password
Invalid IP address or Password
```

- **Player Transfer Account in same server/IP**
  - This test checks transfer player validation if the transfer server ip is the same as the current server.

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

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

Select : 4
Enter username : ypandya
Enter password : ypandya
Enter Old ip-address in following format 132.XXX.XXX.XXX or 93.XXX.XXX.XXX or 182.XXX.XXX.XXX : 182.123.123.123
Enter New ip-address in following format 132.XXX.XXX.XXX or 93.XXX.XXX.XXX or 182.XXX.XXX.XXX : 182.123.123.123
===== The player can't be transfered within same server. =====
```

Basic Test Scenarios:

- **Below test cases are run directly on the server no interaction with the client side is involved in it to verify the correctness of the methods without console integration.**
- **No initial data was loaded while running these test cases.**

```
--------------------------- Basic Test Cases ---------------------------

Test case detail
----------------
Username : testuserdata1
Password : testuserdata1
Old IP : 182.123.123.123 (Asian)
New IP : 93.123.123.123 (Europe)


Test 1 : Create player account with 182.123.123.123 (Asian) Server
Method : createPlayerAccount() , Player created successfully

Test 2 : Sign in player account in 182.123.123.123 (Asian) Server
Method : playerSignIn() , Player sign in successfully

Test 3 : Get player status
Method : getPlayerStatus() , Asian : 1 online , 0 offline.  Europe : 0 online , 0 offline. North American : 0 online , 0 offline.

Test 4 : Sign out player account from 182.123.123.123 (Asian) Server
Method : playerSignOut() , Player sign out successfully

Test 5 : Transfer player account 182.123.123.123 (Asian) to 93.123.123.123 (Europe) Server
Method : transferAccount() , Player account (testuserdata1) is transfered from 182.123.123.123 to 93.123.123.123

Test 6 : Try Sign in player account into old server (182.123.123.123 Asian) after transferting
Method : playerSignIn() , Player account (testuserdata1) doesn't exists

Test 7 : Sign in player account into new server (93.123.123.123 Europe) after transferting
Method : playerSignIn() , Player sign in successfully

Test 8 : Get player status
Method : getPlayerStatus() , Europe : 1 online , 0 offline.  Asian : 0 online , 0 offline. North American : 0 online , 0 offline.

Test 9 : Suspend player account from 93.123.123.123 (Europe) Server
Method : suspendAccount() , Player account (testuserdata1) is suspended

Test 10 : Try Sign in player account after suspended in 93.123.123.123 (Europe) Server
Method : playerSignIn() , Player account (testuserdata1) doesn't exists

Test 11 : Get player status
Method : getPlayerStatus() , Europe : 0 online , 0 offline.  Asian : 0 online , 0 offline. North American : 0 online , 0 offline.
```

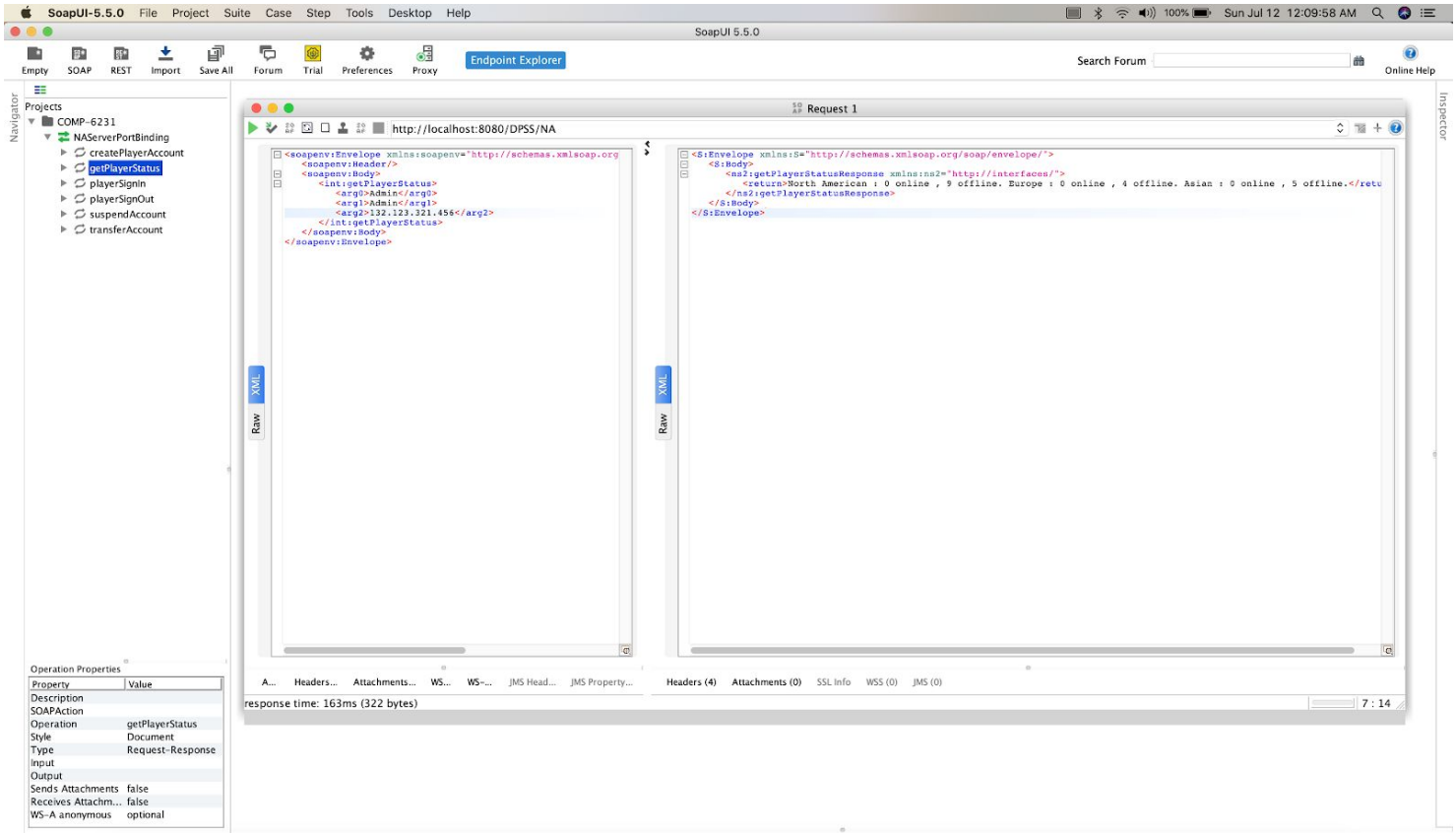
Advanced/Concurrency Test Scenarios:

- **In Advanced/Concurrency tests, no data added into the server.**
- **Below tests cases are run directly on the server no interaction with the client side is involved in it. Synchronization test is added with CreatePlayer, SuspendPlayer and TransferPlayer account.**
- I have made 3 threads that run on Asian server and transfer operations are performed on Europe server
- Each threads try to create 3 different players with username testuserdata1, testuserdata2 and testuserdata3
- Out of 9 thread calls only 3 will be successful and rest of them contains an error
- Each threads try to suspend testuserdata1 player account only 1 out of 3 will be successful
- Each threads try to transfer testuserdata1 player account if its already suspended than none of thread will be successful otherwise only 1 out of 3 will be successful
- Each threads try to transfer testuserdata2 player account only 1 out of 3 will be successful
- Each threads try to suspend testuserdata2 player account if its already transferred than none of thread will be successful otherwise only 1 out of 3 will be successful

```
---------------------------- Advanced Test Cases ----------------------------

Test case detail
----------------
- I have made 3 threads that run on Asian server and transfer operations are performed on Europe server

- Each threads try to create 3 different players with username testuserdata1, testuserdata2 and testuserdata3

- Out of 9 thread calls only 3 will be successful and rest of them contains an error

- Each threads try to suspend testuserdata1 player account only 1 out of 3 will be successful

- Each threads try to transfer testuserdata1 player account if its already suspended than none of thread
  will be successful otherwise only 1 out of 3 will be successful

- Each threads try to transfer testuserdata2 player account only 1 out of 3 will be successful

- Each threads try to suspend testuserdata2 player account if its already transfered than none of thread
  will be successful otherwise only 1 out of 3 will be successful

- I have made 3 threads that run on Asian server

Thread 14 Method : createPlayerAccount() ,  Username : testuserdata1 , Player created successfully
Thread 15 Method : createPlayerAccount() ,  Username : testuserdata1 , Player already exists
Thread 16 Method : createPlayerAccount() ,  Username : testuserdata1 , Player already exists
Thread 16 Method : createPlayerAccount() ,  Username : testuserdata2 , Player created successfully
Thread 15 Method : createPlayerAccount() ,  Username : testuserdata2 , Player already exists
Thread 14 Method : createPlayerAccount() ,  Username : testuserdata2 , Player already exists
Thread 15 Method : createPlayerAccount() ,  Username : testuserdata3 , Player created successfully
Thread 16 Method : createPlayerAccount() ,  Username : testuserdata3 , Player already exists
Thread 14 Method : createPlayerAccount() ,  Username : testuserdata3 , Player already exists
Thread 16 Method : suspendAccount() , Player account (testuserdata1) is suspended
Thread 14 Method : suspendAccount() , Player account (testuserdata1) doesn't exists
Thread 15 Method : suspendAccount() , Player account (testuserdata1) doesn't exists
Thread 16 Method : transferAccount() , Player account (testuserdata1) doesn't exists
Thread 15 Method : transferAccount() , Player account (testuserdata1) doesn't exists
Thread 14 Method : transferAccount() , Player account (testuserdata1) doesn't exists
Thread 16 Method : transferAccount() , Player account (testuserdata2) is transfered from 182.123.123.123 to 93.123.123.123
Thread 14 Method : transferAccount() , Player account (testuserdata2) doesn't exists
Thread 15 Method : transferAccount() , Player account (testuserdata2) doesn't exists
Thread 16 Method : suspendAccount() , Player account (testuserdata2) doesn't exists
Thread 15 Method : suspendAccount() , Player account (testuserdata2) doesn't exists
Thread 14 Method : suspendAccount() , Player account (testuserdata2) doesn't exists

- testuserdata1 and testuserdata2 is either suspended or transfered depending on operation result,
 however testuserdata3 is always created and offline. result of get player status after all thread execution as below


Method : getPlayerStatus() , Asian : 0 online , 1 offline.  Europe : 0 online , 1 offline. North American : 0 online , 0 offline.
```

Web service implementation tested with soapUI :

- SoapUI is an open-source web service testing application/tool for service-oriented architectures and representational state transfers.
- I have tested DPSS web service implementation for getPlayerStatus web service/API with the SoapUI tool.



WSDL Paths:

- All the WSDL files are placed **bin/** directory.
  ASServerService.wsdl
  EUServerService.wsdl
  NAServerService.wsd**l**

- North America : http://localhost:8080/DPSS/NA?wsdl
- Asian : http://localhost:8080/DPSS/AS?wsdl
- Europe : http://localhost:8080/DPSS/EU?wsdl

References:

- https://systembash.com/a-simple-java-udp-server-and-udp-client/
- https://www.geeksforgeeks.org/multithreading-in-java/
- https://www.geeksforgeeks.org/synchronized-in-java/
- https://docs.oracle.com/javaee/6/tutorial/doc/gijti.html
- https://www.javatpoint.com/web-services-tutorial