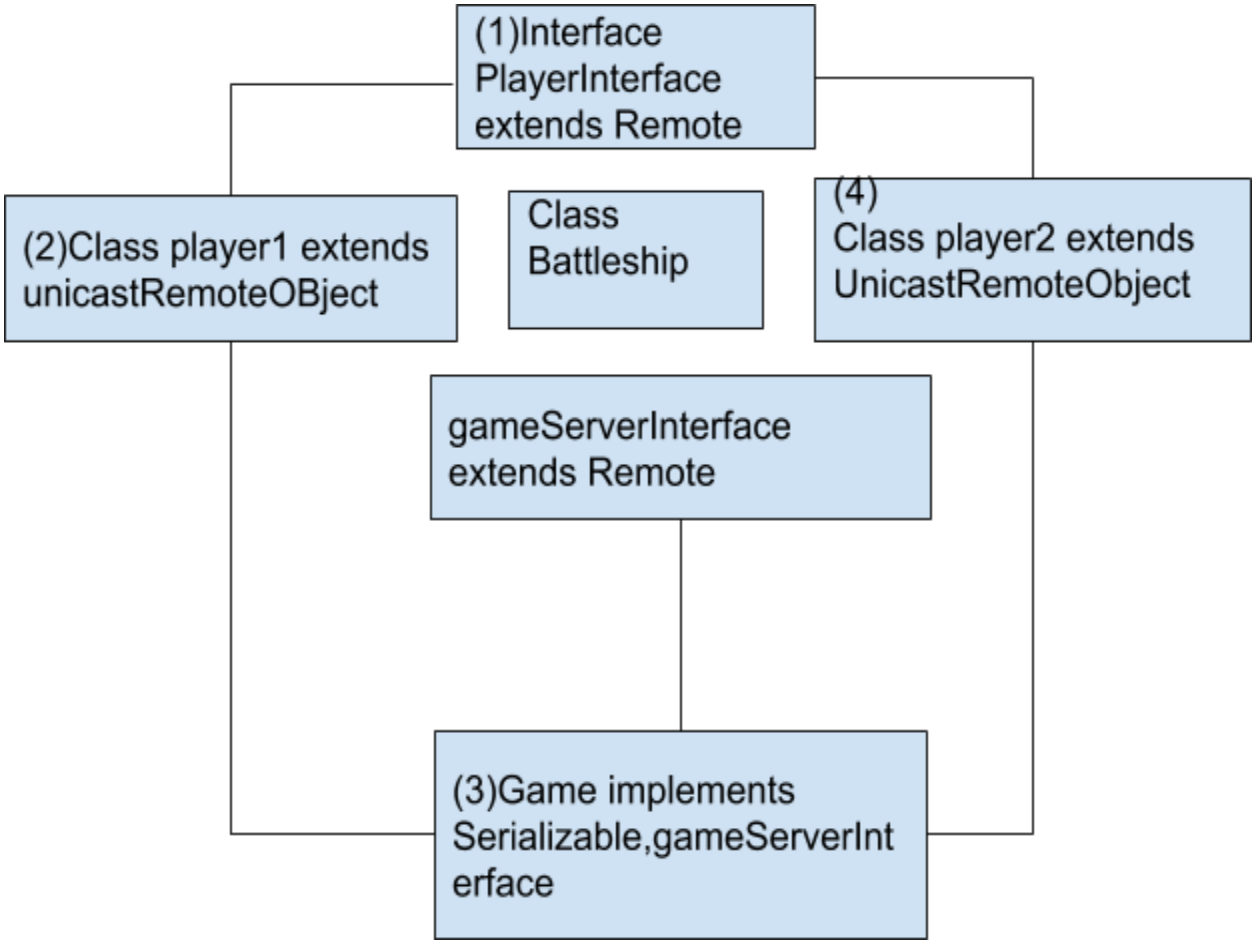
CSCI - 605 (Remote Method Invocation)



1.Design:
Remote Method Invocation for Battleship involved the following interface and Classes
playerInterface- This is the interface which is implemented by player1 and player2, where in they have these methods

- `public String getName() throws IOException;`
- `public void printBoards() throws IOException;`
- `boolean validPlacementPoint(double a,double b) throws IOException;`
- `public boolean PlaceShip(BattleShip playerShip) throws IOException;`
- `public void PlayerMove() throws IOException;`
- `public void init(playerInterface player)throws IOException;`

- `public void sendMessageToClient(String message) throws RemoteException;`

Note: The player1 and player 2 class acts as clients where they connect to the game which implements the gameServerInterface. Game acts as the server ,where it has functionalities to set ships, set the board, print the board. It acts as the middleman where broadcasting of messages can happen via the server from the client.

gameServerInterface- This interface is implemented by the game class, which acts as a server ,where it has the following methods.

- `String sendMsg(String msg) throws IOException;`
- `void player1Move() throws IOException;`
- `void player2Move() throws IOException;`
- `boolean PlaceShipPlayer1(BattleShip ship) throws IOException;`
- `boolean PlaceShipPlayer2(BattleShip ship) throws IOException;`
- `void shipCoordsPlayer1(String a,String b) throws IOException;`
- `public  void broadcastMessage(String message) throws RemoteException;`
- `void shipCoordsPlayer2(String a ,String b)throws IOException;`
- `void getPlayer(playerInterface player)throws IOException;`

Note: game class also acts as the stub, it forms a registry , and binds to it.

BattleShip Class- This class implements serializable and contains the orientation and life of ship details, after every blow, the ship life is reduced, and it denotes whether a ship is alive or not.

2. Learning and Difficulties:

- Learned the process of starting a RMI
- The server side process and client side process differences, where server creates the registry and binds to it, and client looks up in the registry.
- The fact that client casts itself to server side interface and calls its method to implement its own actions is intriguing
- Not sure, if RMI is the ideal approach, to a problem like battleship game, had tremendous difficulty in broadcasting message from server to client side, by the time i figured out , it was late.

- Couldn’t find a good resource online, which could relate to the approach of the battleship game in RMI, and had difficulty in translating it into the battleship problem (if they were slightly close).
- Couldn’t showcase the game logic, barely managed to form the connection , although a lot of game logic method stay unused.
- Tried the naming.rebind and naming.lookup method before, had trouble forming a connection, hence tried the Registry approach , which kind of worked.
- Only tried running it on localhost.