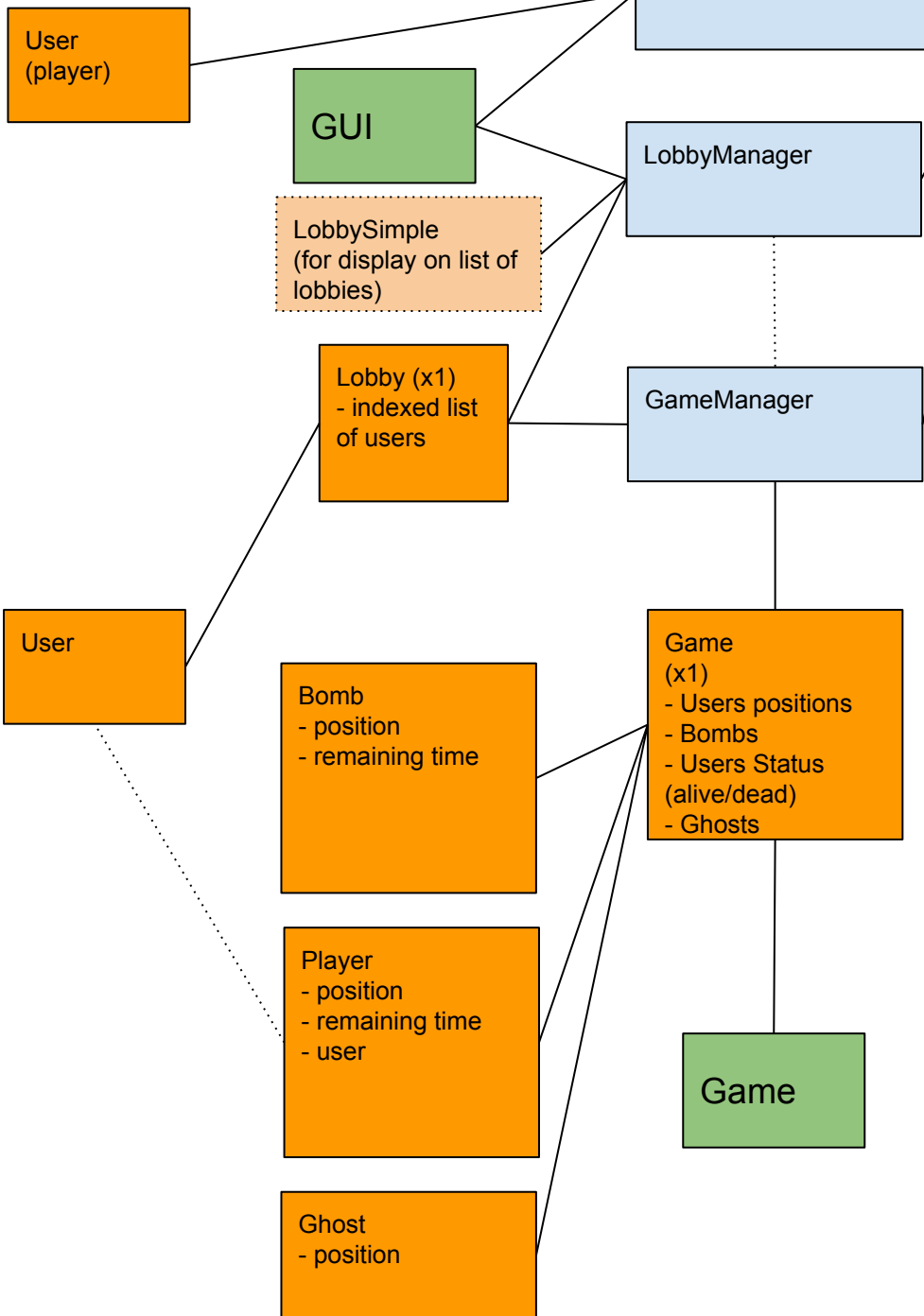


```
# MESSAGE : Testing, testing testing... #
# PLAYERPOSITION : 10 20 # int [2] = [10, 20]
player.setposition(int[])
```

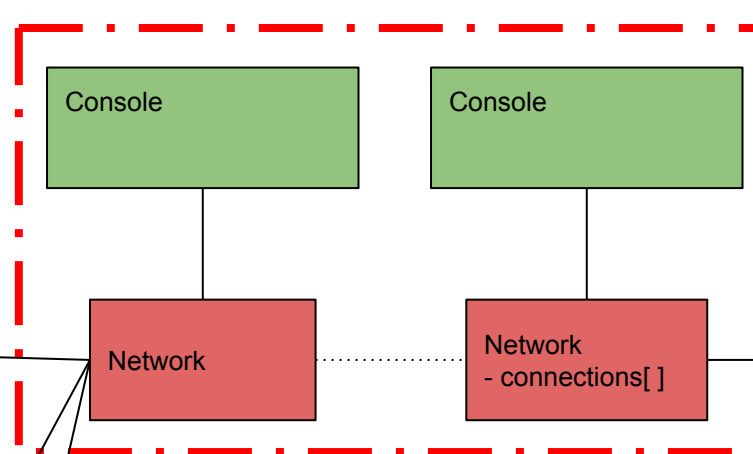
```
User = Network.resolveUser()
```

```
switch (TYPE) {
    int[] array Network.PlayerPosition(string input)
    User.Player.setPosition(array)
}
```



Client:  
Gdy Network wykryje utracone połączenie,  
wyrzucamy gracza z gry, do ekranu rejestracji z  
komunikatem błędu

W pierwszej wersji  
zastąpimy  
managerów po  
prostu klasami



Przez sieć mogą iść różne typy komunikatów:

- Message (for console)
- GameState (s) info about all objects in game (players, bombs, ghosts)
- PlayerPosition (c)
- RequestNewBomb (c)
- RequestNewUser (c)
- NewUser (s)
- RequestNewLobby (c)
- NewLobby(s)
- RequestLobbyList (c)
- LobbyList (s)
- RequestJoin (c)
- Join (s) (also to others in lobby)
- Ready (c)
- AnotherPlayerDisconnected (s)

#### Client

Establishing a connection:  
- ask server if a 'username' is available

#### Lobby:

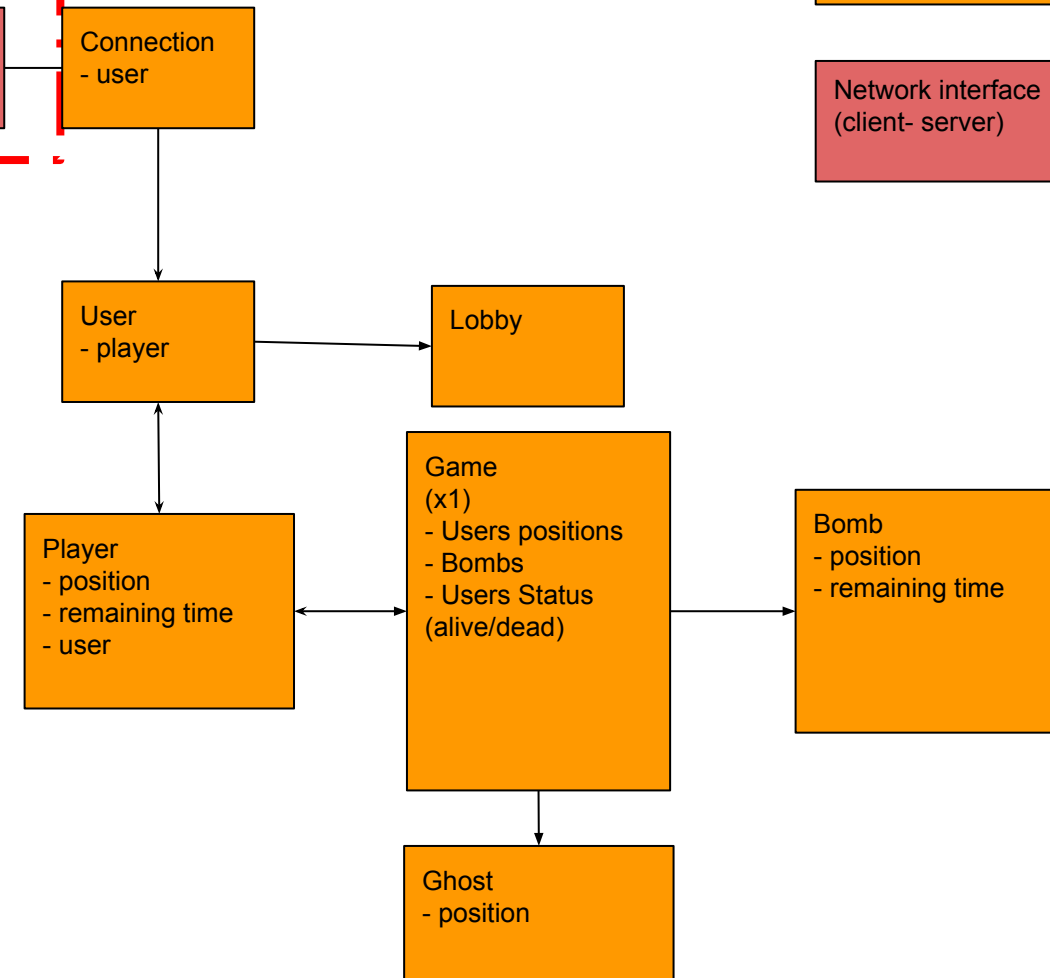
- host sets the settings for new lobby
- players can see the list of open lobbies
- player asks server if they can join lobby
- all players set their status to READY

- (auto) confirmation that everything is loaded

#### Game SETUP:

- send map to the server (host)
- receive map from the server
- handling player input
- sending player position to the server
- displaying enemies position (info from server)

Na razie pracujemy w:  
sk2-bomberman/src/client/Network.cpp#main()  
sk2-bomberman/src/server/Network.cpp#main()



#### Server

#### Lobby:

- server opens the lobby (adds to the list)
- server adds player to lobby

- server starts the game
  - send start position to each player
  - send player color
  - send map

- server is listening for player position and bombs
- server sends position of any placed bombs
- server checks if bomb has affected any players (and sets their status)

- Lobby
- Map
- Position of things

Endpoints

Handler / Manager

Class

Network interface  
(client- server)