**DISCONNECTION AND RECONNECTION BEHAVIOR**

The network architecture of the program requires that:

* There is a Client [the phaseClient]
* There is a Lobby
* There are **ClientHandlers** in the Lobby, each of one talks directly to a single Client.
  + Note that the ClientHandler must interact with the game only through the lobby, and not directly on the controller. This is by design.
* There is a ServerApp that creates ClientHandlers after accepting a new connection.

The Client, in the [MenuPhase], can choose to:

* Create a new lobby (specifying his nickname and the lobby “size”)
* Join an existing lobby (specifying his nickname and the lobby number)
* RE-JOIN a lobby, if he crashed.

What happens when a Lobby is initialized?

* The GameManager is instantiated (which basically is the Controller-Model part)
* In the gameManager, a **IdlePlayers** list is created.
* The ClientHandlers, after waking up, give a starting full-model (MSG\_UPD\_FULL) to the Clients. The Clients copy that update message in their Simplified Model.
* The ClientHandlers then go in a Forever-Listening-On-The-Socket scenario.

What happens if a Client Crashes?

* First of all, the ClientHandler that was listening on the sockets gets an exception. He wakes up, and after controlling that the game is not over (a part which is used in the endgame), goes into **disconnected** phase. The Disconnected phase performs some actions:
  + Closes the streams().
  + Sets a **pendingConnection** Boolean variable to true.
  + Asks the Lobby to put his **playerNumber** in the **IdlePlayers** list.
  + IF the idlePlayers list contains all the playerNumbers, the lobby is basically killed.
  + ELSE, it is asked the lobby to **disconnectPlayer(** himself **)**
    - The lobby calls a specific method in the **actionManager**, called **disconnectPlayer().** That method basically saves the states of the player and his not-performed initial actions, if there are any. It notifies all the players that <player> has crashed, using the messageHelper.
    - IF the player was the current player, the game must advance. Any Middle-object is disabled, and the endTurn() is called.
      * The endTurn(), in this perspective, skips the idlePlayers.
      * Or, either, can use the saved state of the player to re-enable the LeaderCardsObject or the ResourceObject.
  + The ClientHandler then goes to **wait** on a **pendingLock**.

What happens if the game ends while a player is Disconnected?

* The game status is set to GAME\_OVER by the actionManager.
* The ClientHandler are either
  + Listening to the Player (Listening Phase), on the objectInputStream.readObject() method
  + Waiting on the PendingLock.
* The actionManager, when the game ends (condition met if a MSG\_STOP is generated),
  + Sends the Leaderboards to the Clients (last message)
  + Eliminates the Lobby from the Lobbies List.
  + Sets the Boolean variable **LobbyDestroyed** to true.
  + Cuts the streams of any ClientHandler
  + Wakes up any Pending ClientHandlers, setting to **false** the **pendingConnection** variable and notifying() on their **pendingLock**
* The ClientHandler that were listening to their player:
  + Simply checks if status==GameOver (which is true) and kill themselves.
* The ClientHandler that were waiting on the pendingLock
  + Simply checks if lobby is Deleted (true in this case) and kill themselves.

What happens if a player Reconnects mid-game?

* Another ClientHandler will receive a REJOIN message, in which are present the number of the lobby and the player nickname.
* It is checked if the lobby exists.
* **It is checked if there is any ClientHandler in that lobby, whose player has the same nickname has the one requested, and the ClientHandler pendingConnection variable is true.**
  + That means that the player is trying to reconnect.
* If all of this is true, this new ClientHandler..
  + “passes” his streams to the old ClientHandler using substituteStreams()
  + Wakes up the old ClientHandler, setting pendingConnection to false and notifying()
  + Kills himself.
* The old ClientHandler, wakes up! He sees that the lobby has not been deleted, so:
  + He sends a full-model to the client, to update him up.
  + Removes his playerNumber from the IdlePlayers
  + Notifies everyone that the player has reconnected
  + Gets back to the Listening Phase.
* Eventually, the endTurn will perform some check on disconnectedPlayer.

**That’s it.**