SERVER-CLIENT

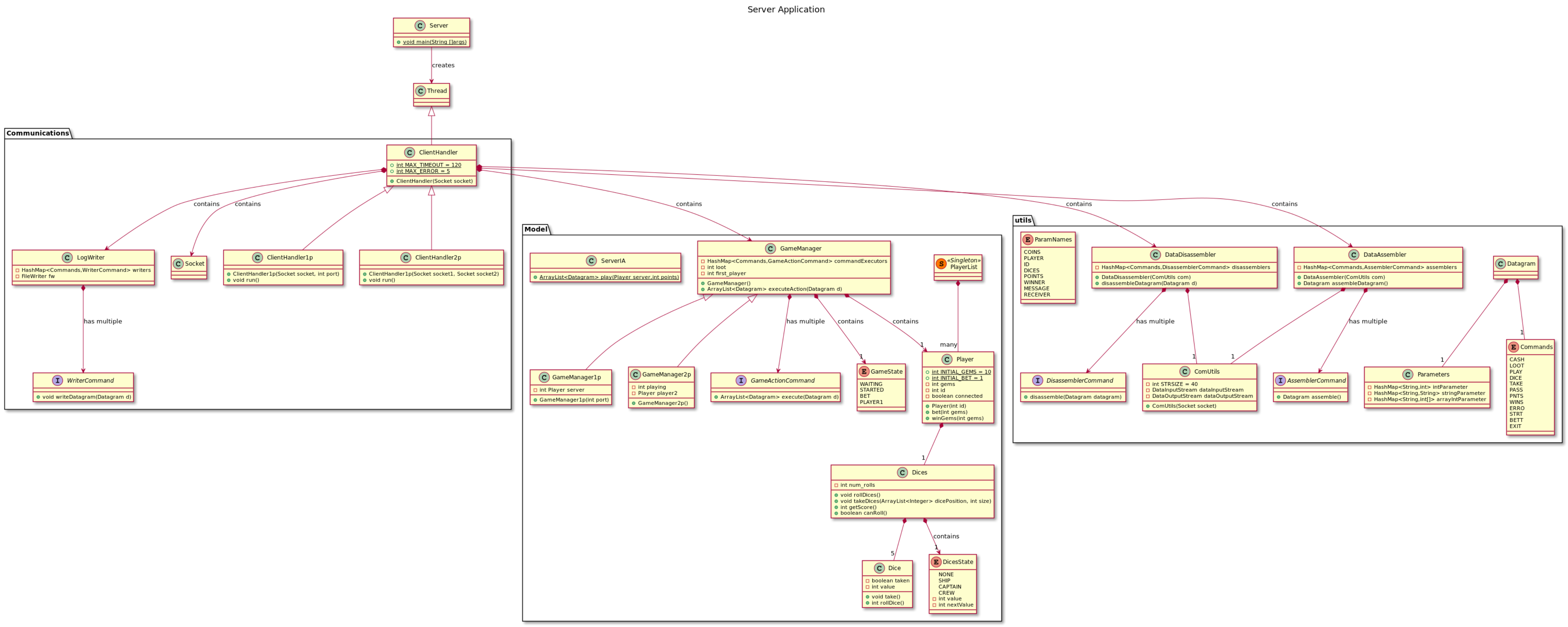
Memory for Server-Client practicum

Manuel Lecha & Oriol Rabasseda

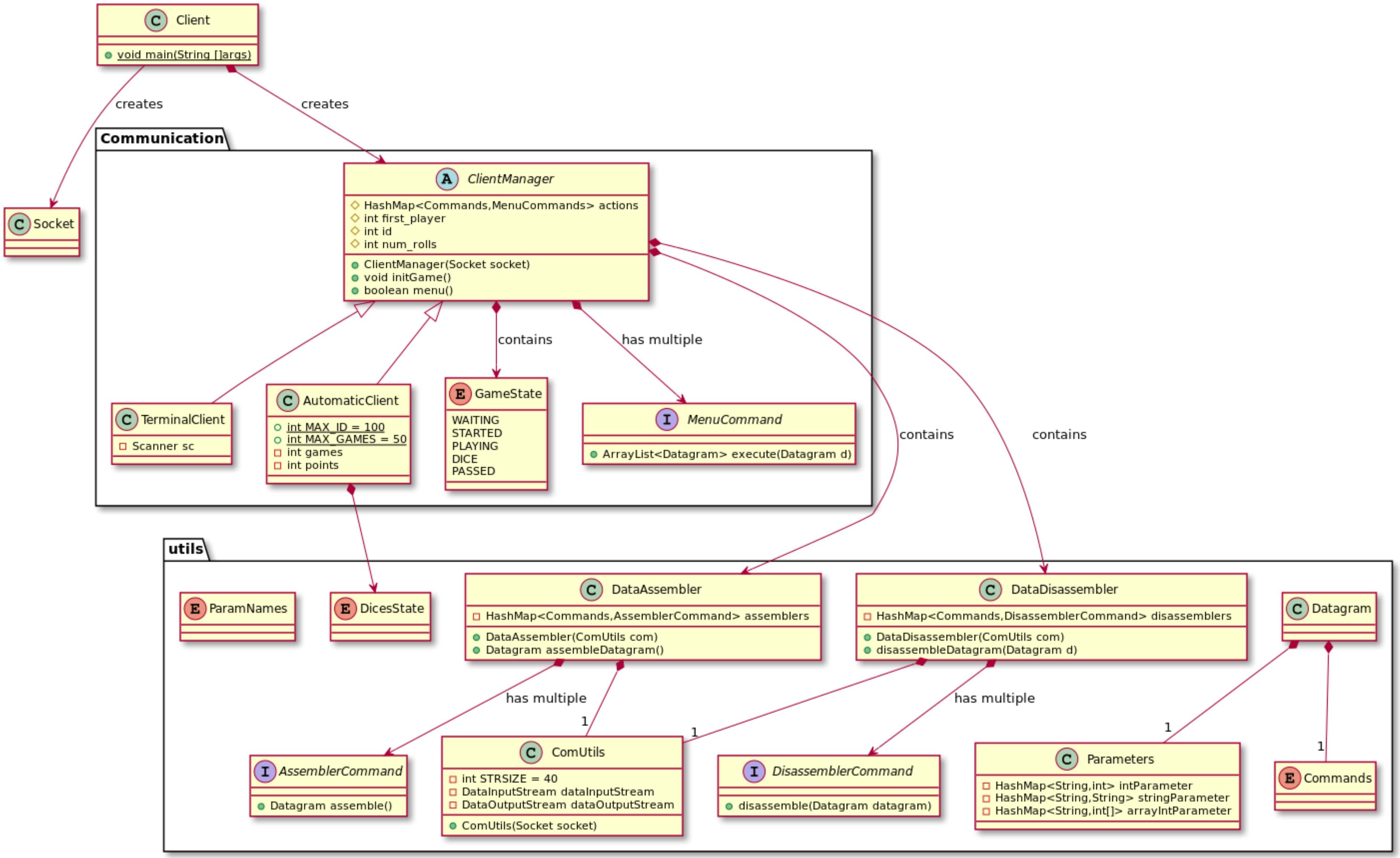
Main program decisions

1. For 1-player games and client, the read timeout to avoid deadlocks are 1 minute. In 2-player games, the timeout is set to 500 ms and we treat it by adding the number of timeout exceptions one of the players raise until they have sum 120 exceptions (1 minute)
2. Selector pattern: every time there is set of actions that depend on a command, we use a selector pattern implementation based on a HashMap where the key is a command and the value is a class. This class implements an interface (same interface for all handlers).
3. Error treatment: when an error is sent either by the server or client, we close the stream
4. To avoid creating one class for every type of datagram (command), we implemented a HashMap containing the parameters where the key is a specific identifier for every conceptual type of parameter. Additionally, we need 3 types of HashMaps to cover every type of parameter (but they aren't instanced if it is not necessary)
5. We have a list of players, so while the server is up, the same ID preserves its cash even if the client disconnects. We need to treat connection of players so a player cannot be connected for more than one client at a time
6. We have designed a wide set of exceptions so every possible exception inside the program is treated differently

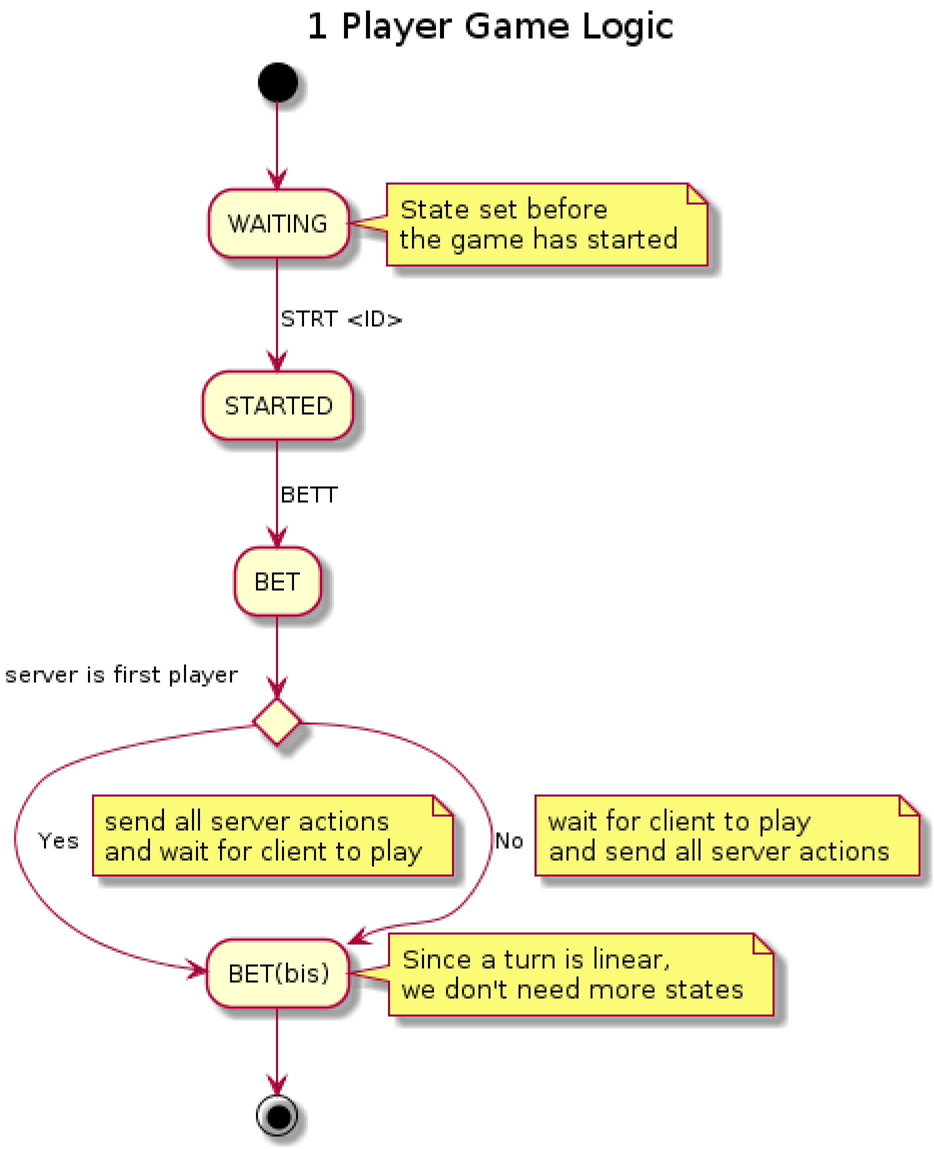
Class diagrams



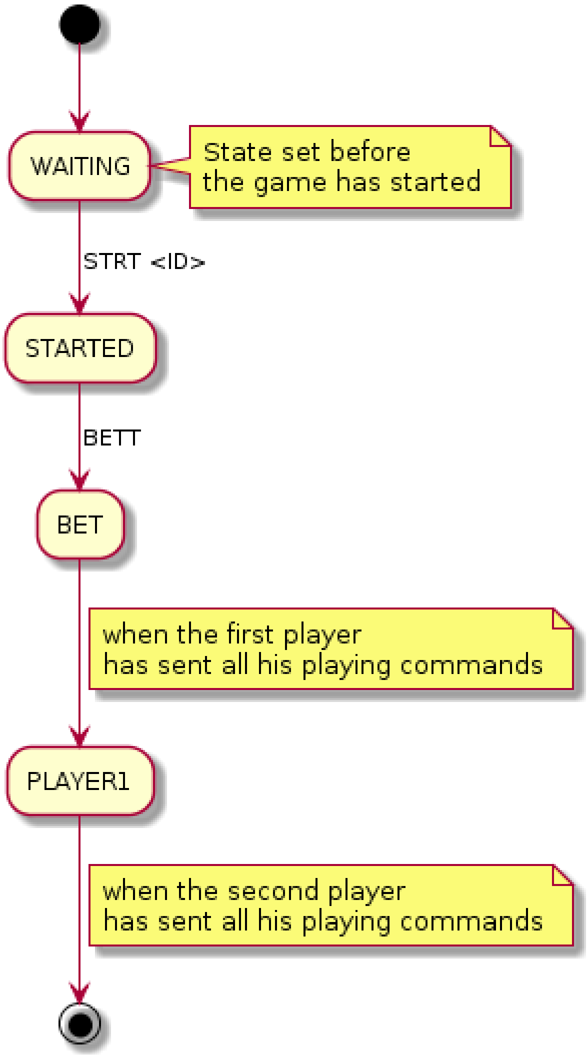
Client Diagram



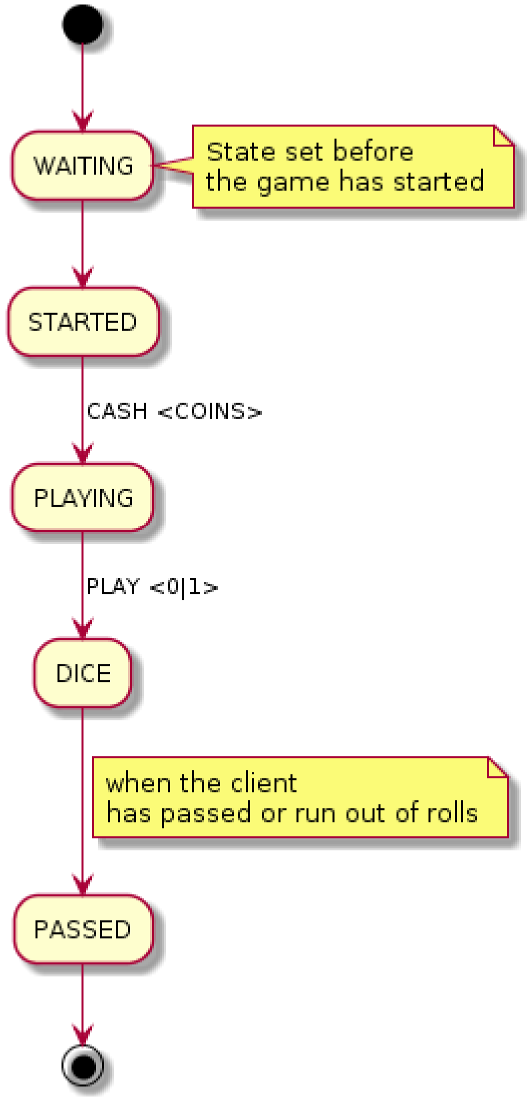
Flow diagrams


1 Player Game Logic
WAITING
State set before the game has started
STRT <ID>
STARTED
BETT
BET
server is first player
Yes
send all server actions and wait for client to play
No
wait for client to play and send all server actions
BET(bis)
Since a turn is linear, we don't need more states

2 Player Game Logic



Client Game Logic


WAITING
State set before
the game has started
STARTED
CASH <COINS>
PLAYING
PLAY <0|1>
DICE
when the client
has passed or run out of rolls
PASSED