Prova finale di ingegneria del software 2024 – gruppo LB08

# Network Documentation

## Introduction

In this paper we will briefly discuss the implementation of the network functionality in our version of "Codex Naturalis".

## Structure

Our application operates following the client-server model.

In the server the acceptance of new connection from the clients is done by the ServerWelcomeSocket, which listens on a specified port. The client and the server communicate through an implementation of the NetworkInterface that offers functionalities to their controllers such as the "send" method that allows them to send a text message through the network to the other host. On the client this interface is implemented as ClientConnectionManager and on the server its counterpart is implemented as ClientHandler.

The NetworkInterface implements itself other two interfaces: the ConnectionObserver interface and the NetworkInputObserver interface.
The NetworkInputObserver interface is used by NetworkInputHandler which runs in its own thread to always check on the input stream of the socket and notifies the NetworkInterface whenever a new message from the network arrives.
The ConnectionObserver interface is used by the Pinger which also runs in its own thread and is used to check the functionality of the network by notifying the NetworkInterface in case the connection with the other host is lost. The Pinger implements the PongObserver interface which is used by the NetworkInterface to notify the pinger of an incoming pong message from the other host (in reply to a previous ping message sent by the pinger).

## Messages

The communication between client and server happens through the exchange of Json text messages. Every message has an identifier for the kind of message that is being transmitted and the rest of the message contains all the information and attributes that the specific message requires to be executed on the receiver host.

Most messages are meant for the application functionality, but there are some special ones like the ones that have the field "type" set to "ping" and "pong" that are messages exchanged between the two network components of the hosts to ensure the stability of the connection. When a message arrives, it's checked whether it's meant to be forwarded to the above controllers or it's a message meant for the network. If it's a ping message the network proceeds to send a pong reply automatically and if it's a pong reply from the other host the network notifies the pinger thread of the received pong.

The application-level messages are managed by the ClientMessageHandler in the client and by the LobbyRequestHandler and the GameRequestHandler by the lobby and game respectively in the server. These classes parse the incoming message and call the proper methods to update the model.

Example of a message sent from the client to the server for placing a card:

```
{
  "command" : "place",
  "placeableCardId" : 36,
  "x" : 2,
  "y" : 2,
  "facingUp" : true
}
```

These messages are generated by the relative message generator classes that take in all the parameters and translates them into their Json form.

## Pinger

The pinger detects a connection loss when a set amount of ping messages is sent but no reply is received. Whenever a ping message is sent the “ping tries” counter gets decreased from the set initial value and each ping is delayed between one another by a defined “PING_INTERVAL” value. If a pong reply is received the pinger gets notified and the “ping tries” counter is reset to the initial value. If that counter reaches 0, the pinger will notify the NetworkInterface that the connection is lost.

## Lobby

To implement the multiple games functionality the server has a lobby as a central hub for the clients to create or join the desired game.

When a new client connects, the lobby automatically assigns a random unique username (like Guest#####) and the thread of the client handler is started. When the player is in the lobby, they can change their username and it’s ensured that two players can’t have the same one. Every user can create a game whenever they want and the players who want to join are presented with a list of the available games that are waiting for the set number of players to start.