Night Train to Budapest. Software Architecture

Purpose

This document aims to outline the architecture of the computer game *Night Train to Budapest* (NTtB).

Abstract

NTtB is a networked multiplayer game written mainly in Java. Even though the software is shipped as a unit, it is functionally divided into two parts: the client and the server. While the client and server share some of the same code, they are two separate programs and completely independently runnable. To be precise: even if a client and a server started from the same JAR are running on the same machine, those two processes have no knowledge of the other and might as well be running on opposite ends of the world. For communication, the server defines the NTtB-Protocol. The client adheres to that protocol and as such can conversate with the server via a network. They represent the two main components of the NTtB architecture; their connection is the protocol. Connection is established via TCP sockets.

Server

The server implements all the main functionality of the game. The main components of the game are client-handlers, lobbies, a chat, the game logic, and the game state. If accepted, a client attempting to connect to the server is put on a client handler thread. All network communication is done through the client handler. All incoming messages are first sent to a protocol parser. The parser interprets the messages and directly executes the appropriate action on the client handler.

If a client asks to start a game, the client handler creates a lobby object and associates the client alias with that lobby as an admin. Other clients on the server can now join that lobby if they so wish. If the admin asks to start a game, all relevant clients are informed of the start, and a new game is created on a new thread that is run within the client handler thread

associated with the admin. The server maintains static lists of all connected clients, lobbies, and running games. As such they are accessible to all client handler threads.

Within the server, the game logic handles all in-game data. When a game is started an instance of a game is created, which has a game state and various handlers. The game state maintains the state of the game and is the sole authority on this. The handlers implement the array of responses to client input while in-game.

In essence, the NTtB server does two things: maintain data, and send appropriate network responses dependent on incoming network messages. The game itself is not played on the server. It is merely maintained on it. The game is played via a TCP socket connection and a client sending the appropriate messages via that connection. *Night Train to Budapest* is shipped with such a client that implements an interface to the game for an end-user.

Client

The client has two main components: the main client program and a GUI. Additionally, it is possible to interact with the client software via the command line. In fact, the game is fully playable from the command line. If one chooses to do so, the end-user must adhere to the NTtB protocol. This means typing in the correctly formated protocol messages. For user convenience the syntax of the protocol is symplified. Messages sent from the command line are thus first formatted by a message-formatter as to adhere to the NTtB protocol before they are sent to the server. In theory, it is possible to send any message and even garbage to the server at any time even if the command shouldn't be used at the current time (e.g. it is pointless to try and open a lobby if one is in a running game, but the client will send the message if entered). Legality is mostly checked by the server. The main way to play the game is via the GUI. The GUI is implemented using JavaFX and in that sense formally is its own application and thus runs on its own thread within the client. The GUI itself has three types of components that loosely follow the MVC design pattern. First, the graphics themselves are implemented in views defined in FXML files. CSS files are used to style the view. JavaFX allows binding FXML files to Java controller classes that implement event handling such as when a button is pressed or an incoming chat message is to be displayed. While some of the game models are maintained by the client class, the GUI itself has a game state model that maintains information for a running game if the client is currently in one. It shall be said that all information regarding the game as a whole on the client-side is only updated based on messages from the server. That is: if the client is in a lobby, what his username is, what type of player he is in a running game, the state of a running game, a list of lobbies and clients on the server, incoming chat messages from other clients, which sound the GUI shall blay or what view should be displayed. The client just stores the respective information. Likewise, the users' messages to the server are generated based on the users' interactions with the GUI. Pressing buttons will generate correctly formatted NTtB-protocol messages which are then sent to the server via the main client class (possibly with additional

parameters like chat messages from a chatbox in the GUI).

Dependencies and Building

The codebase includes Gradle to handle software dependencies of the developed code to external libraries via maven central. The following external libraries are used: Apache Log4j; OpenFX JavaFX; Apache Commons Collections; JUnit Jupiter for testing. Gradle is also used for building shippable software, generating Javadocs, and testing.

Final Remarks

This document aims to give a high-level overview of the most important aspects of the software unified in *Night Train to Budapest*. Specifically, it intends to give an overview of how some of the main components interface with each other. It does not intend to be a complete description of the software architecture. The actual code will in fact look messier and have more components and subcomponents. The abstract architecture of the game does not necessarily perfectly overlap with the concrete file structure of the codebase. This is of course because there are always many ways to implement the same specifications. To recap: the game *Night Train to Budapest* consists of two functionally separate entities, the client and the server, who interact based on the NTtB protocol defined and validated by the server. The client offers a GUI for the user to easily interact with the server and enjoy a more visually appealing experience. The client and server always interact on a TCP socket connection.