SUMANT PATEL (6796841) and ROHAL KABIR (7105836)

COSC 3P91 – Assignment 4

1 SERVER CLASS

In the simulation, the server side of a TCP-based networking solution is represented by the Server class. It starts by defining a port number, after which it tries to create a socket on that port so that it may watch for client connections coming in. When a client connects, the server uses a hardcoded list of valid usernames to compare the client's given username to confirm the client's authentication. It delivers a confirmation message in the event that authentication is successful and rejects the connection otherwise. After a successful authentication attempt, the server awaits a response from the client specifying the chosen vehicle. It uses a GameController instance to start the game and passes the chosen vehicle choice to begin the simulation after parsing this response into an integer indicating the vehicle choice. Any mistakes made during this procedure are recorded, and the client receives the relevant error messages. By creating a communication connection between the client and the traffic simulation game, this server class efficiently enables network-based gaming and interactivity.

2 CLIENT CLASS

In an implementation of TCP-based networking, the Client class acts as the server's client-side equivalent. By connecting to the designated host and port, it uses a socket to create a network connection with the server. The client uses the GameView class to provide prompts and messages to the user in addition to using PrintWriter and BufferedReader for text-based communication with the server. The client employs an authentication system that restricts playback to only predetermined usernames (user1, user2, user 3). The client asks the user to choose a car, transmits the selection to the server, and shows a confirmation message after successful authentication. The client breaks off the connection if authentication is unsuccessful. Users may play the traffic simulation game via the network thanks to this client class, which makes it easier for them to communicate with the server.

3 THE UPDATE GAMEVIEW CLASS

The interface used to show the user prompts and notifications in the traffic simulation game is the GameView class. It takes user input via the Scanner class. The class has methods to show messages, receive input from the user, and prompt the user to perform several tasks such choosing a car, changing lanes, choosing an action for an intersection, and inputting a username. These techniques guarantee that player interactions with the game are organized and simple to utilize. The class also has error handling to verify user input and make sure that only legitimate selections are processed. All things considered, the GameView class improves the user experience by giving prompts and explicit instructions throughout the simulation game.

Authors' address: Sumant Patel (6796841); Rohal Kabir (7105836).

Update date: 9 April 2024

4 THE GAME CONTROLLER CLASS

The main element in charge of overseeing the game logic is the GameController class. It adheres to the Model-View-Controller (MVC) architectural pattern, in which it plays the role of controller, arranging interactions between the view (user interface) and the model (game map, player, bots, etc.). The user chooses a car and the game advances via rounds of lane work and intersection work until the player's vehicle's health drops to zero when the startGame() function is invoked. While working on a lane, the player can advance, change lanes, see the map, or end the game. When at a junction, however, the player must make decisions based on the flow of traffic. The Client class, which talks with the server, provides input to the GameController class, which uses it instead of interacting with it directly. The way the game progresses is determined by this input, which also affects how lanes and intersections behave. This ultimately shapes how the player experiences the traffic simulation game.

5 USING TCP

The traffic simulation system's client and server components communicate with each other using TCP (Transmission Control Protocol). A ServerSocket is produced in the Server class and is used to receive inbound connections from clients. A socket is accepted by the server upon a client connection, enabling bidirectional communication between the two. Text-based communication between the server and client is made possible by the establishment of input and output streams (BufferedReader and PrintWriter) on the input and output streams of the socket, respectively. The client sends messages to the server across the network, and the server receives them, reacts appropriately, authenticates, sends vehicle selection prompts, and starts the game. Similar to this, a Socket is made in the Client class to establish a connection with the server using its hostname and port number. The socket is configured with input and output streams so that messages can be sent to the server and replies can be received. The user's input, including the username and vehicle preference, is sent by the client to the server, which responds with game prompts and an authentication confirmation. The traffic simulation system's gaming is made easier by the bidirectional message flow that TCP facilitates between the client and server.

6 CLIENT SERVER PARADIGM

We Restructured the current classes to divide the client-side and server-side operations and enable TCP communication across a network is the coherent alteration into a Client/Server paradigm. By constructing a ServerSocket and providing logic for connection acceptance, authentication, and client request handling, the Server class is updated to listen for new client connections and manage communication with numerous clients concurrently. The Client class is modified in parallel to manage user interaction, create a TCP connection with the server via a Socket, and transfer user input to the server while presenting the user with server replies. The client and server components of the traffic simulation system communicate data properly and reliably over the network to enable interactive gameplay. This is accomplished by creating a communication protocol and making sure that it is followed. This reorganization improves scalability by enabling several clients to engage with the simulation at once, all the while preserving stability and resilience in the client-server relationship.

7 USAGE OF SOCKETS

Both the client and server classes have been implemented with correct usage of sockets. The endpoints for network-based communication between the client and server are sockets. A ServerSocket is created in the Server class and assigned a port to listen for incoming client connections. A new socket is accepted by the server upon a client connection, creating a channel of communication. Similar to this, a Socket is made in the Client class to establish a connection with the server using its hostname and port number. After that, the socket's input and output streams are configured to allow for bidirectional communication between the client and server. The client and server components create and maintain connections by appropriately using sockets, which makes it possible for data to be exchanged and real-time network interaction in the traffic simulation system. This guarantees effective and dependable communication between the client and server, facilitating smooth gaming and simulation engagement.

8 COMMUNICATION BETWEEN CLIENT AND SERVER

User authentication and simulation execution are covered by a well defined protocol that facilitates appropriate communication between the client and server. The user credentials are sent by the client to the server upon connection, and the server checks them against a pre-established list of legitimate usernames. Secure access to the simulation environment is ensured by this authentication procedure. The client chooses a vehicle for simulation after authenticating, and the server receives this decision. After processing this data, the server starts the simulation with the chosen car. To keep the simulation state and user activities, including car movements and intersection actions, in sync, the client and server communicate with each other throughout the simulation. Following this communication protocol allows the client and server to communicate with each other in an easy-to-use manner. This allows for the cooperative execution of the traffic simulation, authentication, and synchronization of the client-side and server-side components.