CP372 Assignment 1 RFC

Documentation written by
Chen Jin 170631720
Anthony Siprak 190903900
Group 04

1 - Introduction

1.1 - Purpose

This is a client-server application (single server, multiple clients) which is meant to simulate the uses and functions of a bulletin board, allowing for clients to store, request and manipulate notes based on different criterias. Clients interact with the board through a GUI with different buttons controlling different actions.

1.2 - Requirements

This is a multithreaded Java console application which can be booted from the command line with a set of arguments describing different attributes of the board. These attributes are port number, board width, board height, and accepted colors. Clients will establish socket connections to the server, and will make modifications to the global Board structure through a set of request messages.

Additionally, synchronization techniques will be used to ensure that no unwanted conflicts occur when multiple users try to access the board structure. Error handling will also be put in place to handle any unknown client requests.

1.3 Terminology

data - Two integers describing an x y coordinate on the board

note - A “post-it note” defined by its coordinates, width, height, color, message, and status attributes (formatted with parameters <x coordinate> <coordinate> <width> <height> <color> <message> <status>). Clients can post, get, and manipulate these notes through the use of different kinds of messages.

request - Field in GET message, used by client to get pins based on certain criteria and/or Attributes. Request accepts parameters formatted as color=<color>, contains=<data>, refersTo<string>.

1.4 Overall Operation

Once the server is online, users can connect to the server through their client. Once connected, the server will create a new thread to handle the user’s requests. A GUI will be loaded in the client, allowing the client to interact with the board by sending requests. These requests are transmitted from the client to the server as messages. All messages that can be sent conform to a predefined list of messages outlined in section 2 of this document. Clients are able to send requests freely, one at a time, in order to perform different operations on the global board structure. Once the user is finished using the app, they can use the DISCONNECT message to be disconnected from the server.

2 - Messages

2.1 - Request Message Types

POST <note> - The POST message is received by the server from the client. It describes a “post-it note” for the server to add to the board.

The POST message has 6 fields. Two describing the x and y coordinates of the lower left corner, two describing the height and width of the post it note, one which specifies the color of the note, and a sixth field which holds the message to go on the post it note.

GET <request> - The GET message is received by the server from the client. The client requests notes and describes the criteria for notes to be sent from the server to the client.

The GET message is conditional and may accept arguments in one of two formats. If the client sends the command GET PINS it will receive all pinned messages as a response. The client may also send a list of criteria. This list may be one or multiple items which are of the format “color=<color>”, which isolates all notes of a certain color, “contains=<data>” which isolates all notes which contain a certain coordinate on the board, and “refersTo=<string>” which isolates all notes which contain a given substring.

PIN <data> - The PIN message is received by the server from the client. The <data> condition of PIN describes x and y coordinates leading to a point on the board. It describes criteria for notes to be pinned by the server.

UNPIN <data> - The UNPIN message is received by the server from the client, and uses the same <data> (x, y coordinate) condition as PIN. It describes criteria for notes to be unpinned by the server.

SHAKE - The SHAKE message is received by the server from the client. The server deletes all unpinned notes on the board.

CLEAR - The CLEAR message is received by the server from the client. The server deletes all notes on the board.

DISCONNECT - The DISCONNECT message is received by the server from the client. The server sends a response message to confirm to the client the message was sent successfully. The server and client then both end the connection they have over that port.

2.2 - Server Message Information

SERVER - SERVER messages are standalone messages sent by the server to clients as a response to server events, as opposed to user requests. Server messages are accompanied by parameters describing the event, as well as any possible actions taken, such as if the client must be disconnected from the server.

ServerShutdown - This error is sent if the Server is shut down while the client is still connected. This error will display an error message as well as close the client’s connection to the server.

2.3 - Client Message Information

CLIENT - CLIENT messages are standalone messages generated by the client to notify the user and/or server of localized events occurring in user clients.

ClientDisconnect - This message is sent by the client if the client closes the connection to the server.

3 - Synchronization

3.1 - Rundown

This application stores the board as a global structure. Threaded clients will make requests in order to access the board. In the cases where the board must be modified the application will ensure that client access to the board structure is synchronized, so that there are no unwanted modifications, overwrites, or errors. Clients looking to POST, PIN, SHAKE, or CLEAR would require special write access to the board structure.

3.2 - Implementation

Synchronization is handled by the synchronize keyword. This means that only one thread can enter the method parseMessage() at one time. This is the method used for every single operation on the Bulletin Board (BBoard) structure so there is no chance of race condition errors.