Randy Dang
Yi Ling Pang

Notes Board Protocol

Table Of Contents

Introduction

1. <u>Description of Protocol</u>

   The purpose of this protocol is to implement a server and client based Message Board. Consider an office space that contains a bulletin board in which notes and messages can be pinned upon. The different messages can vary from advertisements, event notices, important information, etc. Each message is written on its own separate post-it note and then pinned onto the board. Each pin occupies a space on the board and cannot occupy the space of another. In this case, a single pin is used instead. Notes on the other hand can occupy the same space, and have the option of using the same pin as the other note, or use another pin with the pin already being used (This will mean the corresponding message will have two pins on it). Pins will have to be unpinned and then pinned again to account for multiple notes on the same space. Notes vary in the message it contains, location on the board, and the colour of the note itself.

2. <u>Terminology</u>

   Client:

   A person or entity that establishes connections for the purpose of sending requests to the server.

   Server:

   An application program that accepts connections in order to service requests by sending back responses to clients.

   Connection:

   A transport layer virtual circuit established between two application programs for the purpose of communication

   Request:

   A request message sent by the client to initiate a transaction (See section 4.3 for different types of requests)

   Response:

   A response message sent by the server to indicate the status of request

   Message Board:

   A virtual board that will contain the coordinates of all the notes and the pins

   Note:

   A virtual note that consists of its size (width and height), position on the message board (x and y coordinates), color of the note, and the message it contains

   Message:

   A string composed by the client that will be stored within its own note, accompanied by the other attributes of the note

3. <u>Architecture</u>

   3.1 Data Structures

All entries (notes) will be stored individually as an object in a Vector. Each object will contain the notes x and y position, size (width and height) of note, color, message and status (Pinned/Unpinned). This storage message will allow the server to easily identify items that match with each GET request from the client. The use of a Vector is also thread safe, allowing for multiple clients to make requests without problems.

All pins will be stored as well as an object in a Vector. Each object will contain the x and y coordinates of the pins

The Note class will contain a notes x and y position, size (width and height), color, message, and status (as an integer 0=unpinned or 1=pinned)
The Pin class will contain a pins x and y coordinates.

3.2 Client and Server class

The Server class will contain its own Client class that acts as a thread for the purpose of allowing multiple clients to connect to the same server

The Client class will contain its own GUI upon running. This GUI is used to send requests to the server. A function will be present to formulate a single line command that will be sent to the server based on the data filled in within the GUI

4. Client

4.1 GUI Requirements

The Graphics User Interface is implemented through Java's Javax Swing API. When the program is first run, there will be a window which contains a user-friendly form for the user to connect to the server and make requests. The following will be displayed:

1. Text fields to input IP address of the server and the port number, accompanied with a CONNECT/DISCONNECT button
2. A text field to type a command to be sent to the server, accompanied with a POST button
3. A GET button accompanied by seperate text boxes to provide the appropriate data: Color (of the note), Contains (x and y coordinates that the note lie on), Refers To (a substring within the message of the note). Any text boxes left blank will be disregarded.
4. A PIN/UNPIN button which will invoke a dialogue to enter the following in seperate text boxes: X (x-coordinate for a point on the board), and Y (y-coordinate for a point on the board)
5. A text field to display the result of the request, sent by the server (This will include request responses and error messages)

6. A CLEAR button that will remove all notes that are not pinned
7. A GET PINS button that will send a request to acquire all the coordinates of every pin to the server

4.2 Method Definitions

1. GET PINS: Server must return all the coordinates of pins
2. GET: Client requests to retrieve notes based on color, coordinate on the board, and content of message
3. POST: Client can create a note to post on the board and specify the coordinates, dimensions, color of note, and the message
4. PIN/UNPIN: Client can specify coordinates of the board, in which notes that lie on said coordinates will be pinned or unpinned.
5. CLEAR: Server will delete the entire board of notes save for the notes which are pinned.
6. CONNECT/DISCONNECT: Client has the option to connect or disconnect with server at any given time

4.3 Request Syntax

1. GET color = <color> contains = <coordinates> refersTo = <substring>
   (Note that not all text boxes have to be filled. Requests like the following are applied to the corresponding data:
   i. GET color=white: Will force the server to provide all notes that are colored white
   ii. GET contains=3 7: Will force the server to provide all notes that contain points with (3, 7)
   iii. GET color=yellow refersTo=Dog: Will force the server to provide all notes that are colored yellow and have the substring “Dog” in its message
2. POST <coordinates of lower left corner of note> <width> <height> <color> <message>
3. PIN/UNPIN <x coordinate>,<y coordinate>

5. <u>Server</u>

5.1 Initialization

1. In order to start the server, the user can enter in the command line **java NoteServer <port number> <width> <height> <color> [<color>] …**
2. The <width> and <height> refers to the width and height of the message board
3. The <color> attributes will be the list of colors that will be available for the user to choose from
4. The first <color> attribute will be the default color in the case that the client does not enter a color

5.2 Connections

1. Server must be able to handle multiple client connections (Multithreading)
2. It is important to make sure the clients chosen IP address and port number match that of the server in order to successfully connect
3. Upon successful connection, server will send the client a list of available colors to choose from and the size of the board (Determined from initialization)

5.3 Responses

1. Handshake: Initial connection response (This will include a welcome message, list of available colors and the size of the board that will be sent to the client)
2. Valid Request: If the request sent from the client is invalid, message will be sent that the request was invalid, otherwise the server will send a message stating that the request was valid and begin processing the request
   Responses:
   i. Connection Response: <board_width> <board_height> <colors>
   ii. POST Response: “Note Successfully posted!”
   iii. GET Valid Response: <status> <color> Note at <x, y>: <message>
      - Status syntax: 0 = unpinned, 1 = pinned
   iv. GET Invalid Response: “There are no notes satisfying your request!”
   v. GET PINS Valid Response: <x, y>
   vi. GET PINS Invalid Response: “There are currently no relevant PINs on the board”
   vii. PIN Valid Response: “Pin is successfully pinned!”
   viii. PIN Invalid Response: “A pin already exists at this location”
   ix. CLEAR Response: “All unpinned notes have been removed”
   x. ERROR Response: “Error handling connection”

5.4 Message Processing

1. Each request require different parameters that are passed to the server (See Section 4.3 for request parameters and syntax)
2. Server must be able to determine the type of request and then process the data according to that request

6. Synchronization Policy

Requests being sent to the server will be processed on a first come first serve basis. A client that sends in a request first will be processed before any other requests being made afterwards. Requests will be processed one at a time.

7. Error Handling

7.1 Connection

Initial Connection must be successful, where the IP address matches with the server and the correct port number is inputted

7.2 Input restrictions

Unrecognized colors, commands, invalid coordinates (non-numeric or out of bounds), and a message exceeding a length of 142 characters or an empty message will result in a respective error message printed on the GUI

7.3 Retrieval Errors

A request for content that does not exist (i.e. no existing substring, color of note does not exist, no note at coordinates) will result in no action taken and a respective error message

7.4 Pinning Errors

Pinning at an existing pin coordinate, Unpinning at a coordinate with no notes, Unpinning at a coordinate with no pins will result in no action taken and a respective error message

7.5 Server shutdown

Upon an unexpected shutdown of the server, clients must be notified that the connection has unexpectedly been interrupted, and client will be instructed to reconnect

7.6 Empty Board

The very first client on the server will not be able to retrieve any notes because the board is empty. An error message will be sent back in the result box.

8. Security Considerations

Data is not persistent (Notes will be deleted upon shutdown of the server)