CP372

Assignment 1

RFC

Keven Iskander

ID: 160634540

Table of Contents

Introduction **3**
- Purpose 3
- Requirements 3
- Definitions 4
- Operation 5

Responsibilities **5**
- Client Responsibilities 5
- Server Responsibilities 5

Message Formatting **5**
- Format of Note Objects 5
- Client Requests 6
- Server Responses 6

Protocol **6**

Data Structures **7**

Border Cases **7**

Errors **7**

1 Introduction

1.1 Purpose

This application allows clients to store and retrieve digital notes on a server. The notes are posted to a simulated "board" with specific coordinates and properties such as colour. The note posting system is comparable to that of a physical post-it note. In a real-life setting, a user will write a message on a post it and pin it to a certain point on any surface. This platform will simulate that process through a client-server connection.

1.2 Requirements

The client should initially be capable of requesting a connection to the server. Without an established connection through the appropriate port, the client will not be able to send or receive any data. The user must then send different commands to the server for specific purposes. These commands are POST, GET, PIN, UNPIN, CLEAR, CONNECT and DISCONNECT. The server.py file needs to be running in a terminal so that the client.py file to establish a connection. When the server is initialized, it requires command line arguments to define the port number, width of the board, height of the board and all of the possible colour selections.

- If the user selects POST <note>, the client will be prompted to provide an x-coordinate, y-coordinate, width, height, colour and message. The message is initially posted without any pins. If all the parameters are provided and the colour corresponds to a valid color (in arguments), the server will return "NOTE POSTED". Otherwise the server will return "NOTE NOT POSTED"
- If the user selects GET <results>, the server will respond by retrieving all the notes corresponding to the user provided parameters. Here are some examples of the GET function:
  - GET colour = white
  - GET contains 4 6 refersTo Hello

  For the server to recognize the input, all terms must be space delimited. For example, if the user types GET colour=white, the server will not be able to retrieve the information. It must be written as GET colour = white.

- If the user selects PIN <coordinates>, the client will be prompted to provide an x y coordinate. The server will then PIN all notes containing those coordinates. If any notes are properly pinned, the client will receive a "NOTE PINNED"

message. Otherwise the server will return “NOTE NOT PINNED”. Here or some examples of the PIN function:
    - PIN 1 2
    - PIN 4 6

  Once again, all inputs should be space delimited so that the server to recognize the user request.

- If the user selects UNPIN <coordinates>, the client will be prompted to provide an x y coordinate. The server will then UNPIN all notes containing those coordinates. If any notes are properly unpinned, the client will receive a “NOTE UNPINNED” message. Otherwise, the server will return “NOTE NOT UNPINNED”. Here are some examples of the UNPIN function:
    - UNPIN 5 6
    - UNPIN 8 90
- If the user selects CONNECT, the client will establish a connection with the server. A connection must be established in order to send and receive any data
- If the user selects DISCONNECT, the client will disconnect from the server, and will exit.

1.3 Definitions

<coordinates> Is a pair of integers which refers to the x and y coordinates at any given point on the board (limited by board dimensions defined in arguments).

<note> Is an object used to assign values to any given note. It includes an x coordinate, y coordinate, width, height, colour and number of pins associated. Each note is initiated unpinned (0 pins associated).

<request> Is used for the GET command. A successful retrieval using the GET command can be done by providing relevant x y coordinates <coordinates>, colour <colour> or a string <string>. It can also be performed using a combination of the two.

<response> Is returned by the server once a GET command is called. It will either return nothing, since none of the parameters specified by the user refer to any given note stored on the server, or it will return that note in a string format containing all relevant NOTE data.

1.4 Operation

The server must be running for the client to establish a connection. The client is then able to perform all operations from their own terminal by selecting one of the six command options. All relevant data is returned to the client terminal via the server connection.

2 <u>Responsibilities</u>

2.1 Client Responsibilities

The client is responsible for requesting a connection to the server before engaging in any data transfer. Once a connection is established, all inputs must follow user protocol, otherwise the server will prompt the user to re-enter data. The message portion, x y coordinates all other required parameters must be entered by the user when engaging with the server. The results are displayed in the user's terminal

2.2 Server Responsibilities

The server is responsible for reading incoming messages and decoding them. It will then categorize all the information according to the format it was received in. If a note is being posted, the server is responsible for storing the note data in a note object. If the data is not submitted in the correct format, or it cannot be decoded, the server will return an error message. Data is only stored and updated so long as the server is online. If the user attempts to disconnect from the server, it must close the connection and close the client terminal safely.

3 <u>Message Formatting</u>

3.1 Format of Note Object

The note object must have the following format: x-coordinate y-coordinate width height colour and message. Each of these parameters must be separated by spaces for the server to be able to categorize them.

3.2 Client Requests

The client will initiate requests by entering a number corresponding to their request in the terminal. The format is as follows:

1-CONNECT

2-DISCONNECT

3-POST

4-GET

5-PIN

6-UNPIN

If selected, each of these options will initiate a response. CONNECT and DISCONNECT will immediately connect or disconnect the client from the server. POST, GET, PIN and UNPIN will prompt the user to enter the parameters requested by the server. If the request is recognized, the server will send the appropriate response.

3.3 Server Responses

If the client's data is properly formatted and the request is confirmed, the server will send confirmation back to the user. POST and PIN/UNPIN will respond either by confirming the action or notifying the user that the action could not be performed. Get will return a list of all notes corresponding to the user request. If an error occurs, the server will catch and prevent any crashes.

4 Protocol

The protocol being used is a TCP client to server connection protocol. Multithreading is also implemented, in order to allow multiple connections to the server without signals interfering with each other.

5 Data Structures

All notes are stored within a note class. This includes x-coordinates, y-coordinates, height, width, colour, message and pins. Each of these is stored as a string, except for pins which is stored as an integer. When comparing notes, certain values are converted to integers, and when returning notes, the server sends them as strings.

6 Border Cases

There are methods which prevent the user from crashing the server with their inputs. When a note cannot be posted because the user did not provide enough parameters, the server returns a "NOTE NOT POSTED" message rather than closing the user terminal and returning an index out of range error. If the user inputs an invalid entry when prompted to input a number (corresponding to a specific command) the client interface will simply prompt the user to enter a command once again, rather than sending the server any unformatted data. However, this does not apply to command inputs. Once the user selects one of six command options, and enters their data in an incorrect format, the server will simply be unable to categorize or store this data and will return a message similar to: "NOTE NOT POSTED".

7 Errors

The list of client errors are as follows

- Invalid input
- NOTE NOT POSTED
  NOT PINNED
  NOT UNPINNED