Design Documentation

Matthew Arbesfeld, Matthew Tancik, Anubhav Jain

Datatypes

Packet

Packet is an abstract class that implements the Serializable interface, which is used to represent the all the types of data that is sent between the client and the server.

process(PacketHandler handler) - Abstract method to handle processing of packets

Many subclasses of Packet:

PacketBoardIdentifierList(BoardIdentifier[] boards)

A Packet that represents all the Boards currently known to the server. Takes a list of boards as input.

boards() – returns a deep copy of the boards stored in the packet

PacketBoardModel(BoardModel boardModel)
A packet that represents the current boardModel
boardModel() – returns the boardModel stored in the packet

PacketBoardUsers(Identifiable[] Users)
A packet that represents all the users associated with a boardModel
boardUsers() – returns the users stored in the packet

PacketClientReady()
A packet that represents a client is ready to accept drawing commands

PacketDrawCommand(DrawCommand drawCommand)
A packet that represents a draw command on a board
drawCommand() – returns the DrawCommand associated with the packet

PacketExitBoard()
A packet that represents a client quitting a board

PacketJoinBoard(BoardIdentifier boardName)
A packet that represents a client joining a board
boardName() – returns the boardName that was joined by the client

PacketMessage(String text)
A packet that represents a message, used for chat client
Text() – returns the text stored in the message

PacketNewBoard(BoardIdentifier boardName, int width, int height)

A packet that represents all the information from client of a new board
boardName() – returns the boardName
width() – returns the width of the board
height() – returns the height of the board

PacketNewClient(ClientIdentifier senderName)
A Packet that represents new client information
senderName() – returns ClientIdentifier senderName

PacketHandler – interface to handle different types of Packets

PacketType – enum for all the different packets
New_Client, New_Board, Client_Ready, Join_Board, Exit_Board, Board_Model, Board_Users, Board_Identifier_List, Draw_Command

Identifiable – interface to create different identifier objects for Users and Boards
Identifier(int id, String name)
Abstract class that represents a way of identifying boards and users. Implements Identifiable
Id() – returns the integer id
Name() – returns the String name
identifier() – returns this Identifier
hashCode() – returns a hash value of the identifier
equals(Object obj) – returns a Boolean of whether this Identifier is equal to another object
toString() – returns a String representation of the Identifier

ClientIdentifier – subclass of Identifier, representing a client
BoardIdentifier – subclass of Identifier, representing a board

BoardModel(BoardIdentifier boardName, DrawableBase canvas)
BoardModel(BoardIdentifier boardName, DrawableBase canvas, Identifiable[] initUsers)
BoardModel is the model used to represent a board, with a DrawableBase representing the canvas on top of which we draw and a synchronizedSet<Identifiable> of users. It implements Identifiable, Drawable, and Serializable
addUser(Identifiable user) – adds the user to the synchronizedSet of users
containsUser(Identifiable user) – Boolean value if the board contains the user
removeUser(Identifiable user) – removes the user from the synchronizedSet of users
users() – returns a deep copy array of Identifiable[] users
drawPixel(Pixel pixel) – draws pixel on the canvas
width() – returns the canvas' width
height() – returns the canvas' height
canvas() – returns the Drawable canvas
identifier() – returns the boardName

Protocol

We used a client/server architecture as our network architecture to handle data transfer between the clients and our server.

Grammar for messages from client to server:

PACKET: PACKET_NEW_CLIENT | PACKET_NEW_BOARD |PACKET_JOIN_BOARD | PACKET_EXIT_BOARD | PACKET_DRAW_COMMAND | PACKET_CLIENT_READY | PACKET_MESSAGE

PACKET_NEW_CLIENT : SENDER_NAME

PACKET_NEW_BOARD : BOARD_NAME INT INT // boardName, width, height

PACKET_JOIN_BOARD: BOARD_NAME

PACKET_EXIT_BOARD: NONE

PACKET_DRAW_COMMAND: DRAW_COMMAND

PACKET_CLIENT_READY: NONE

PACKET_MESSAGE: TEXT

SENDER_NAME: TEXT

BOARD_NAME: TEXT

DRAW_COMMAND: INT INT TEXT // x, y, color as string

TEXT: ('\-'|[~A-Za-z0-9.,'"?\\!&@#$\^();/=\+\]\[])+;

Grammar for messages from server to client:

PACKET : PACKET_JOIN_BOARD | PACKET_EXIT_BOARD | PACKET_DRAW_COMMAND | PACKET_BOARD_MODEL | PACKET_BOARD_USERS | PACKET_BOARD_IDENTIFIER_LIST | PACKET_MESSAGE

PACKET_JOIN_BOARD : BOARD_NAME

PACKET_EXIT_BOARD : NONE

PACKET_DRAW_COMMAND : DRAW_COMMAND

PACKET_BOARD_MODEL: BOARD_MODEL

PACKET_BOARD_USERS : NAME*

PACKET_BOARD_IDENTIFIER_LIST: BOARD*

PACKET_MESSAGE: TEXT

BOARD: ID NAME

ID: INT

NAME: TEXT

BOARD_NAME: TEXT

BOARD_MODEL: BOARD_NAME “canvas” NAME*

DRAW_COMMAND: INT INT TEXT // x, y, color as string

TEXT: ('\-'|[~A-Za-z0-9.,'"?\\!&@#$\^();/=\+\]\[])+;

The client sends data in packets letting the server know what is happening on the client side. This includes:

- new client packet, meaning that a new client has joined the server
- new board packet, meaning that a client has created a new board

- join board packet, meaning that a client has joined a board
- exit board packet, meaning that a client has left a board
- draw command packet, meaning that a client has drawn on their board
- client ready packet, meaning that a client is ready to receive draw commands
- message packet, which holds chat information

The server sends data back to the client in packets as well, giving it updates to the new state that it should reflect as other users work on the same board. These packets include:

- join board packet, meaning that a client has joined a board
- exit board packet, meaning that a client has left the board
- draw command packet, meaning that a client has drawn on the board and the board should be updated
- board model packet, which sends the current board to the client so it can update itself to the appropriate point
- board users packet, which sends the current users connected to the board to the client
- board identifier packet, which sends the current boards on the server to the client
- message packet, which holds a new chat message to be added to the chat menu

Concurrency Strategy

Testing Strategy

We plan on testing the following objects we create: Packets, BoardModel, Identifiers, Client, ClientController, ClientState, DrawCommand, DrawCommandPixel, Drawable, Canvas2d, DrawableCanvas2d, DrawableBase, Pixel, Server, ServerSocketHandler, ServerSocketState, SocketHandler, StrokeTypes, and StrokeProperties