Table Of Content

Package controller

Class Summary

Controller

controller

Class Controller

```
java.lang.Object
    |
    +--controller.Controller
```

< Fields > < Constructors > < Methods >

public class **Controller**
extends java.lang.Object

Fields

network

`private NetworkHandler` **`network`**

running

`private boolean` **`running`**

session

`private Session` **`session`**

Constructors

Controller

`public` **`Controller`**`(Session session)`

Constructor with the session this controller talks to

Parameters:

session - - the session that the controller passes commands into

Methods

append

```
private byte[] append(byte[] data,
                     byte[] addition)
```

Parameters:

data - - first byte array
addition - - second byte array

Returns:

- a single byte array connected

---

append

```
private byte[] append(byte[] data,
                     java.lang.String addition)
```

Parameters:

data - - first byte array
addition - - string to be added

Returns:

- a single byte array connected

---

getDynamicData

```
private java.lang.String getDynamicData(byte[] data,
                                        int start)
```

Retrieves a given data from byte array where the start index is the size of the string to be received

Parameters:

data - - the byte[] array containing the information
start - - the index with the allocated size, start + 1 is where the string begins

Returns:

- the retrieved data

---

handshakeRequest

```
protected byte[] handshakeRequest(byte[] inFromClient)
```

CLIENT COMMAND - handshakeRequest

Parameters:

inFromClient - {'R' (1), IDlength (1), ID (x)}

Returns:

- the byte array to be returned

---

mediaPing

```
protected byte[] mediaPing(byte[] inFromClient)
```

CLIENT COMMAND - mediaPing

Parameters:

inFromClient - {'M' (1), 'P' (1)}

Returns:

- the byte array to be returned

---

mediaRequest

```
protected byte[] mediaRequest(byte[] inFromClient)
```

CLIENT COMMAND - mediaRequest

Parameters:

inFromClient - {'M' (1), 'R' (1), imageID (1), packet# (1)}

Returns:

- the byte array to be returned

---

parseNetworkCommand

```
public byte[] parseNetworkCommand(byte[] data)
              throws java.lang.IllegalArgumentException
```

CLIENT COMMAND CONTROL POINT - handles all incoming client commands

Parameters:

data - the byte array containing the command params

Returns:

- the byte array to be returned based on the initial command

Throws:

java.lang.IllegalArgumentException - - if the command given is invalid

start

```
public void start()
            throws java.net.SocketException
```

Starts the network socket to start accepting packets from clients

Throws:

java.net.SocketException - - if the port 9876 is in use

---

stop

```
public void stop()
            throws java.lang.IllegalArgumentException
```

Stops the network socket from accepting packets from clients

Throws:

java.lang.IllegalArgumentException - - if nothing is running

---

vote

```
protected byte[] vote(byte[] inFromClient)
```

CLIENT COMMAND - vote

Parameters:

inFromClient - {'V' (1), IDlength (1), ID (x), voteNumber (1), Votelength (1), Vote (x)}

Returns:

- the byte array to be returned

Package gui

Class Summary

ConsoleOutput

VotoDesktop

VotoDesktopFX

---

gui

Class ConsoleOutput

```
java.lang.Object
    |
    +--java.io.OutputStream
        |
        +--gui.ConsoleOutput
```

All Implemented Interfaces:
java.io.Closeable, java.io.Flushable

< Fields > < Constructors > < Methods >

---

public class **ConsoleOutput**
extends java.io.OutputStream

Fields

output

```
private final javax.swing.JTextArea output
```

---

sb

```
private final java.lang.StringBuilder sb
```

---

title

```
private java.lang.String title
```

Constructors

ConsoleOutput

```
public  ConsoleOutput(javax.swing.JTextArea out,
                      java.lang.String title)
```

Methods

close

```
public void close()
```

Overrides:

close in class java.io.OutputStream

---

flush

```
public void flush()
```

Overrides:

flush in class java.io.OutputStream

---

write

```
public void write(int b)
```

Overrides:

write in class java.io.OutputStream

---

gui

Class VotoDesktop

```
java.lang.Object
    |
    +--gui.VotoDesktop
```

All Implemented Interfaces:

java.awt.event.ActionListener, java.lang.Runnable

< Fields > < Constructors > < Methods >

---

public class **VotoDesktop**
extends java.lang.Object

implements java.awt.event.ActionListener, java.lang.Runnable

Fields

c

```
private Controller c
```

---

connectButton

```
private javax.swing.JButton connectButton
```

---

f

```
private javax.swing.JFrame f
```

---

fileChooser

```
private javax.swing.JFileChooser fileChooser
```

---

hostButton

```
private javax.swing.JButton hostButton
```

---

hostPanel

```
private javax.swing.JPanel hostPanel
```

---

ipField

```
private javax.swing.JTextField ipField
```

---

ipLabel

```
private javax.swing.JLabel ipLabel
```

---

joinButton

private javax.swing.JButton **joinButton**

---

openButton

private javax.swing.JButton **openButton**

---

s

private Session **s**

---

startPanel

private javax.swing.JPanel **startPanel**

---

t

private javax.swing.Timer **t**

Constructors

VotoDesktop

public **VotoDesktop**()

Methods

actionPerformed

public void **actionPerformed**(java.awt.event.ActionEvent e)

---

hostGUI

private void **hostGUI**()

---

joinGUI

```
private void joinGUI()
```

---

main

```
public static void main(java.lang.String[] args)
```

---

run

```
public void run()
```

---

startGUI

```
private void startGUI()
```

---

gui

Class VotoDesktopFX

```
java.lang.Object
    |
    +--javafx.application.Application
        |
        +--gui.VotoDesktopFX
```

All Implemented Interfaces:
java.awt.event.ActionListener, java.lang.Runnable

< Fields > < Constructors > < Methods >

---

public class **VotoDesktopFX**
extends javafx.application.Application
implements java.awt.event.ActionListener, java.lang.Runnable

Fields

fc

```
private javafx.stage.FileChooser fc
```

hostButton

private javafx.scene.control.Button **hostButton**

hostGrid

private javafx.scene.layout.GridPane **hostGrid**

hostStage

private javafx.stage.Stage **hostStage**

joinButton

private javafx.scene.control.Button **joinButton**

joinGrid

private javafx.scene.layout.GridPane **joinGrid**

joinStage

private javafx.stage.Stage **joinStage**

picPane

private javafx.scene.control.ScrollPane **picPane**

pics

private javafx.scene.layout.VBox **pics**

rootHost

private javafx.scene.layout.BorderPane **rootHost**

rootJoin

```
private javafx.scene.layout.BorderPane rootJoin
```

---

s

```
private Session s
```

---

votingButtons

```
private javafx.scene.control.Button[] votingButtons
```

Constructors

VotoDesktopFX

```
public  VotoDesktopFX()
```

Methods

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent e)
```

---

answerStage

```
public void answerStage()
```

---

hostGUI

```
private void hostGUI(javafx.stage.Stage p)
```

Host GUI displays IP address, allows user to open pictures, displays pictures, and lets the user select the correct answer for each picture

---

joinGUI

```
private void joinGUI(javafx.stage.Stage p)
```

---

main

```
public static void main(java.lang.String[] args)
```

---

openFile

```
private void openFile()
```

Open picture from file chooser to host pane

---

run

```
public void run()
```

---

start

```
public void start(javafx.stage.Stage primaryStage)
```

Start GUI has host or join options

Overrides:

start in class javafx.application.Application

Package networking

Class Summary

NetworkHandler

UDPSocket

---

networking

Class NetworkHandler

```
java.lang.Object
    |
    +--networking.NetworkHandler
```

All Implemented Interfaces:
java.io.Closeable, java.lang.Runnable

< Fields > < Constructors > < Methods >

---

public class **NetworkHandler**
extends java.lang.Object
implements java.io.Closeable, java.lang.Runnable

Author:

zomby This class controls whats coming in and out of the UDPSocket onPacketReceived, it passes it to the parser and finds the proper command

Fields

control

`private Controller` **`control`**

---

socket

`private UDPSocket` **`socket`**

Constructors

NetworkHandler

```
public  NetworkHandler(Controller control)
        throws java.net.SocketException
```

Create the controller

Throws:

java.net.SocketException - - If something is already using port 9876

Methods

close

```
public void close()
```

Close the socket

---

onPacketReceived

```
public void onPacketReceived(java.net.DatagramPacket inFromClient)
```

Parses the DatagramPacket into a set of keyword arguments, passes them onto a command parser

Parameters:

inFromClient - - The datagram packet received from client

---

reply

```
public void reply(byte[] data,
                  java.net.DatagramPacket in)
```

Replies the given byte array to the location of the datagram packet

Parameters:

data - - the byte array to be sent
in - - the datagram packet to have the byte array sent too

---

run

```
public void run()
```

Start the socket

networking

Class UDPSocket

```
java.lang.Object
    |
    +--networking.UDPSocket
```

All Implemented Interfaces:
java.io.Closeable, java.lang.Runnable

< Fields > < Constructors > < Methods >

---

public class **UDPSocket**
extends java.lang.Object
implements java.io.Closeable, java.lang.Runnable

Author:

zomby This class is designed to open a Datagram socket on a defined port and then both send and receive

Fields

PORT

```
private final int PORT
```

---

isListening

```
private volatile boolean isListening
```

---

listener

```
private NetworkHandler listener
```

---

socket

```
private java.net.DatagramSocket socket
```

Constructors

UDPSocket

`public` **`UDPSocket`**`(NetworkHandler l)`

Create a new datagram socket, catch the error of something else using the port.

Methods

close

`public void` **`close`**`()`

Wait a second for the isListening to take affect then close it cause the loop in run will have stopped This is not necessary but guarantees no error

---

run

`public void` **`run`**`()`

Start listening and in an infinite loop constantly receive packets, sending them up to the listener.

---

send

`public void` **`send`**`(java.net.DatagramPacket outToClient)`

Send DatagramPacket to client

Parameters:

outToClient - - Datagram packet to be sent

Package session

Class Summary

Client

Question

Session

Vote

---

session

Class Client

```
java.lang.Object
    |
    +--session.Client
```

< Fields > < Constructors > < Methods >

---

public class **Client**
extends java.lang.Object

Fields

ID

`private java.lang.String` **`ID`**

---

voteList

`private java.util.ArrayList` **`voteList`**

Constructors

Client

`public` **`Client`**`(java.lang.String clientID)`

Client constructor

Parameters:

clientID - - this client's ID

Methods

equals

`public boolean` **`equals`**`(java.lang.String ID)`

Tells whether or not the sent arg is the current client

Parameters:

ID - - a client ID to validate

Returns:

True if the ID sent in matches the current client's ID False if the IDs don't match

---

getClientVoteList

`public java.util.ArrayList` **`getClientVoteList`**`()`

Returns a list of all the clients final votes for current session

Returns:

List of client votes

---

getLastVote

`public` Vote **`getLastVote`**`()`

Returns the last vote sent by the specified client

Returns:

last received vote

---

setLastVote

```
public void setLastVote(Vote lastVote,
                        Vote oldVote)
```

Sets the clients most recently sent vote as their current vote

Parameters:

lastVote - - the most recently sent vote
oldVote - - reference to previous client vote to be discarded

---

session

Class Question

```
java.lang.Object
    |
    +--session.Question
```

< Fields > < Constructors > < Methods >

---

public class **Question**
extends java.lang.Object

Fields

answer

```
private Vote answer
```

---

answerSet

```
private java.util.HashMap answerSet
```

---

choices

```
private java.util.HashMap choices
```

---

currentSession

```
private Session currentSession
```

---

imageID

```
private int imageID
```

---

questionImg

```
public java.util.ArrayList questionImg
```

Constructors

Question

```
public  Question(Session s,
                 java.util.ArrayList img,
                 int imageID)
```

Question constructor

Parameters:

s - - current session of this question
img - - image loaded with this question
imageID - - image ID for image param

Methods

addVote

```
public void addVote(java.lang.String clientID,
                    java.lang.String clientVote)
```

Adds a vote from a client to the current question

Parameters:

clientID - - ID of the client sending the vote
clientVote - - the actual vote sent by the client

---

endQuestion

```
public void endQuestion()
```

Ends the current question and sets each client's last vote to be their final recorded vote

---

getAnswer

`public Vote getAnswer()`

Returns the correct answer for this question

Returns:

the Vote corresponding to the correct answer

---

getImagePacket

```
public byte[] getImagePacket(int packetNum)
              throws java.lang.IllegalArgumentException
```

Returns a byte array containing the image for the current question

Parameters:

packetNum - - the packet number for the desired image

Returns:

image byte array

Throws:

java.lang.IllegalArgumentException -

---

imageID

`public int imageID()`

Returns the image ID for current question

Returns:

the image ID

---

imageSize

`public int imageSize()`

Returns the size of the current questions image

Returns:

current image size

---

setAnswer

`public void` **`setAnswer`**`(java.lang.String ans)`

Sets the correct answer for current question

Parameters:

ans - - String for the correct answer

---

session

Class Session

```
java.lang.Object
    |
    +--session.Session
```

< Fields > < Constructors > < Methods >

---

public class **Session**
extends java.lang.Object

Fields

ID

`public java.lang.String` **`ID`**

---

clientList

`private java.util.ArrayList` **`clientList`**

---

control

`private Controller` **`control`**

---

currentQuestion

`public Question` **`currentQuestion`**

Constructors

Session

```
public  Session()
```

Methods

addClient

```
public void addClient(java.lang.String ID)
```

Adds a new client to the session's client list with their ID

Parameters:

ID - - the new client's ID

---

getClient

```
public Client getClient(java.lang.String clientID)
```

Returns a client object based off of the client ID passed in

Parameters:

clientID - - ID of the desired client

Returns:

a client object

---

getCurrentImageID

```
public int getCurrentImageID()
```

Returns the ID for the image of the current question

Returns:

ID of current question image

---

getCurrentImagePacketCount

```
public int getCurrentImagePacketCount()
```

Returns the number of packets for the current question image

Returns:

image packet count

---

getCurrentImageSize

```
public int getCurrentImageSize()
```

Returns the size (in bytes) of the current session image

Returns:

image size in bytes

---

getImagePacket

```
public byte[] getImagePacket(int imageID,
                             int packetNumber)
              throws java.lang.IllegalArgumentException
```

Returns the image packet for the current question of the session

Parameters:

imageID - - ID of the question image
packetNumber - - corresponding packet number

Returns:

byte array packet for the current image

Throws:

java.lang.IllegalArgumentException -

---

hasImage

```
public boolean hasImage()
```

If a current Question is loaded

Returns:

- True if loaded, false if not (null)

---

loadImage

```
public java.util.ArrayList loadImage(java.lang.String filename)
                           throws java.io.IOException
```

Loads an image into an arraylist of bytearray of 64KB each

Parameters:

filename - - The filename of the image to be loaded

Returns:

- An arraylist of 60KB byte arrays

Throws:

java.io.IOException - - if the filename is invalid

start

```
public void start()
            throws java.net.SocketException
```

Starts a Voto session

Throws:

java.net.SocketException -

stop

```
public void stop()
            throws java.lang.IllegalArgumentException
```

Stops (or ends) the current Voto session

Throws:

java.lang.IllegalArgumentException -

session

Class Vote

```
java.lang.Object
    |
    +--session.Vote
```

< Fields > < Constructors >

public class **Vote**
extends java.lang.Object

Fields

ID

```
private final java.lang.Integer ID
```

Constructors

Vote

```
public  Vote(java.lang.Integer i)
```

Vote constructor

Parameters:

i - - ID for the vote object

INDEX

Q

R

S

T

U

V

W