//Set up socket where clients can communicate with server

**public void** startRunning(){  
 **try**{  
 server = **new** ServerSocket(5000,100);  
 showMessage(**"Waiting for someone to connect...\n"**);  
 createGroupList();  
 waitForConnection();  
 }  
  
 **catch**(Exception e){Server.showMessage(**"server line 61"** + e.toString());}  
}

//Create Group List on server startup by getting information from database

**private void** createGroupList(){  
 String query = **"select groupName, groupID, userID, groupImage from groups;"**;  
  
 **try**{  
 ResultSet resultSet = dbLib.selectQuery(query);  
 **while** (resultSet.next()) {  
 **byte**[] file;  
 String groupImage = resultSet.getString(**"groupImage"**);  
 **try**{  
 Path imagePath = Paths.get(groupImage);  
 file = java.nio.file.Files.readAllBytes(imagePath);  
 }  
 **catch**(Exception e){  
 file=**new byte**[0];  
 }  
 groups.add(**new** Group(server, resultSet.getString(**"groupName"**),

resultSet.getInt(**"groupID"**), resultSet.getInt(**"userID"**), file));}  
 }  
 **catch**(Exception e){Server.showMessage(**"server line 74"** + e.toString());}  
 *// The name of the file to open.*}

//Wait for Connection thread on Server Side

**private void** waitForConnection(){  
 t1 = **new** Thread(  
 **new** Runnable(){  
 **public void** run(){  
 **while**(**true**){  
 **try**{  
 Connection tempConn = **new**

Connection(server.accept(),dbLib);  
 connections.add(tempConn);  
 }  
 **catch**(Exception e){Server.showMessage(**"server line 98"** +

e.toString());}  
 }  
 }  
 });  
 t1.start();  
}

//Set up input/output streams when new connection joins and set up access to the database

**public** Connection(Socket socket, LoadDriver dbLib){  
 *connection* = socket;  
 Server.showMessage(**"\nNow connected to "** +

*connection*.getInetAddress().getHostName()+**"\n"**);  
 **try**{  
 output = **new** ObjectOutputStream(**new**

BufferedOutputStream(*connection*.getOutputStream()));  
 output.flush();  
 input = **new** ObjectInputStream(**new**

BufferedInputStream(*connection*.getInputStream()));  
 }  
 **catch**(Exception e){Server.showMessage(**"conn line 44"** + e.toString());  
 }  
 setUpThread();  
  
 **try**{  
 **this**.dbLib=dbLib;}  
 **catch**(Exception e){Server.showMessage(**"conn line 52"** + e.toString());  
 }  
}

//Each time a client connects to the server set up a thread to monitor for new messages from the client. //When a new message arrives check to see what kind of message it is (i.e. user join/exit, message, user //login, etc.) and use different logic based on the type. If client closes the connection catch the //exception and remove the client from all groups they are part of.

**private void** setUpThread(){  
 thread= **new** Thread(  
 **new** Runnable(){  
 **public void** run(){  
 **while**(**true**){  
 **try**{  
 Message message = (Message) input.readObject();  
 **if**(message.**type**.equals(**"CMD"**)){  
 **if**(message.**cmd**.equals(**"JOIN"**)){  
 …  
 }  
 **else if**(message.**cmd**.equals(**"EXIT"**)){  
 …  
 }  
 **else if**(message.**cmd**.equals(**"CREATE"**)){  
 …  
 }  
 }  
 **else if**(message.**type**.equals(**"MSG"**)){

**if**(message.**cmd**.equals(**"FILE"**) ||

message.**cmd**.equals(**"DOODLE"**)){  
 …  
 }  
 **else**{  
 …  
 }  
 }  
 **else if**(message.**type**.equals(**"SEND CODE"**)){  
 …  
 }  
 **else if**(message.**type**.equals(**"REGISTER"**)){  
 …  
 }  
 **else if**(message.**type**.equals(**"LOGIN"**)){  
 …  
 }  
 **else if**(message.**type**.equals(**"BAN"**)){  
 …  
 }  
 }  
 **catch**(Exception e){  
 **try**{  
 input.close();  
 *connection*.close();  
 removeConnection();  
 **break**;  
 }  
 **catch**(Exception ex){  
 Server.showMessage(**"conn261"**+ex.toString());  
 }  
 }  
 }  
 }  
 });  
 thread.start();  
}

**Client Side – Desktop**

//Set up connection on client side to server

**public void** startRunning(){  
 **try**{  
 *connection* = **new** Connection(**new**

Socket(InetAddress.*getByName*(**serverIP**),5000),**this**);  
 *connection*.setUpThread();  
 }  
 **catch**(Exception e){}  
}

//Set up input and output streams once connection to server is established

**public** Connection(Socket socket, LoginController loginForm){  
 login = loginForm;  
 *connection* = socket;  
 **try**{  
 output = **new** ObjectOutputStream(**new**

BufferedOutputStream(*connection*.getOutputStream()));  
 output.flush();  
 input = **new** ObjectInputStream(**new**

BufferedInputStream(*connection*.getInputStream()));}  
 **catch**(Exception e){}  
}

//Set up thread on client side to listen for messages from server and take different actions based on //type of message

**public void** setUpThread(){  
 thread= **new** Thread(  
 **new** Runnable(){  
 **public void** run(){  
 **while**(**true**){  
 **try**{  
 Message message = (Message) input.readObject();  
 System.***out***.println(message.**type**);  
 **if**(message.**type**.equals(**"CMD"**)){  
 **if**(message.**cmd**.equals(**"START"**)){  
 …

}  
 **else if**(message.**cmd**.equals(**"ADD"**)){  
 …

}  
 **else if**(message.**cmd**.equals(**"REMOVE"**)){  
 …

}  
 **else if**(message.**cmd**.equals(**"CREATE"**)){  
 …  
 }  
 }  
 **else if**(message.**type**.equals(**"BANNED"**)){  
 …  
 }  
 **else if**(message.**type**.equals(**"MSG"**)){  
 …  
 }  
 **else if**(message.**type**.equals(**"LOGIN SUCCESSFUL"**)){  
 …  
 }  
 **else if**(message.**type**.equals(**"LOGIN UNSUCCESSFUL"**)){  
 …  
 }  
 **else if**(message.**type**.equals(**"BAN"**)){  
 …  
 }}  
 **catch**(Exception e){}  
 }  
 }  
 });  
 thread.start();  
}

//Send message from client to server by writing to output stream

**public static void** sendMessage(Message message){  
 **try**{  
 output.writeObject(message);  
 output.flush();  
 }  
 **catch**(Exception e){}  
}

**Client Side – Android**

// Set up connection on client side to server

**public void** startRunning(){  
 Thread thread = **new** Thread(**new** Runnable(){  
 @Override  
 **public void** run() {  
 **try** {  
 *connection* = **new** Connection(**new** Socket (InetAddress.*getByName*

(**serverIP**) ,5000) ,getLoginActivity(),getContext());  
 *connection*.setUpThread();  
 } **catch** (Exception e) {  
 **final** Exception error = e;  
 runOnUiThread(**new** Runnable() {  
 @Override  
 **public void** run() {  
 **errorMessage**.setText(error.toString());  
 }  
 });  
 }  
 }  
 });  
  
 thread.start();  
}

//Set up input and output streams once connection to server is established

**public** Connection(Socket socket, LoginActivity loginForm, Context c){  
 **this**.*\_CONTEXT* = c;  
 *notificationCount* = 0;  
 *notificationGroupID* = 0;  
 *login* = loginForm;  
 *connection* = socket;  
 **try**{  
 *output* = **new** ObjectOutputStream(**new**

BufferedOutputStream(*connection*.getOutputStream()));  
 *output*.flush();  
 *input* = **new** ObjectInputStream(**new**

BufferedInputStream(*connection*.getInputStream()));}  
 **catch**(Exception e){e.toString();}  
}

//Set up thread on client side to listen for messages from server and take different actions based on //type of message

**public void** setUpThread(){  
 *thread*= **new** Thread(  
 **new** Runnable(){  
 **public void** run(){  
 **while**(**true**){  
 **try**{  
 Message message = (Message) *input*.readObject();  
 **if**(message.**type**.equals(**"CMD"**)){  
 **if**(message.**cmd**.equals(**"START"**)){  
 *…* }  
 **else if**(message.**cmd**.equals(**"ADD"**)){  
 …}  
 **else if**(message.**cmd**.equals(**"REMOVE"**)){  
 …}  
 **else if**(message.**cmd**.equals(**"CREATE"**)){  
 *…*  
 }  
 }  
 **else if**(message.**type**.equals(**"BANNED"**)){  
 *…*  
 }  
 **else if**(message.**type**.equals(**"MSG"**)){  
 …  
 }  
 **else if**(message.**type**.equals(**"LOGIN SUCCESSFUL"**)){  
 *…*  
 }  
 **else if**(message.**type**.equals(**"LOGIN UNSUCCESSFUL"**)){  
 *login*.unsuccessfulLogin(message.**message**);  
 }  
 **else if**(message.**type**.equals(**"BAN"**)){  
 …  
 }}  
 **catch**(Exception e){e.toString();}  
 }  
 }  
 });  
 *thread*.start();  
}

//Send a file on Android client

**private void** sendFile(File file){  
 **byte** [] bytearray = **new byte** [(**int**)file.length()];  
 String extension = **""**;  
  
 **int** i = file.getPath().lastIndexOf(**'.'**);  
 **int** p = Math.*max*(file.getPath().lastIndexOf(**'/'**),

file.getPath().lastIndexOf(**'\\'**));  
  
 **if** (i > p) {  
 extension = file.getPath().substring(i+1);  
 }  
 **try**{  
 FileInputStream fin = **new** FileInputStream(file);  
 BufferedInputStream bin = **new** BufferedInputStream(fin);  
 bin.read(bytearray, 0, bytearray.**length**);  
 Message newMessage = **new** Message(**"MSG"**, **"FILE"**, **groupID**, **chat**.**username**,

bytearray, extension);  
 newMessage.**userID** = **chat**.**userID**;  
 sendMessage(newMessage);  
 }  
 **catch**(Exception ex){}  
}

//Open Existing Doodle on Android

**private void** setupDrawing(){  
 **drawPath** = **new** Path();  
 **drawPaint** = **new** Paint();  
 **drawPaint**.setColor(**paintColor**);  
 **drawPaint**.setAntiAlias(**true**);  
 **drawPaint**.setStrokeWidth(20);  
 **drawPaint**.setStyle(Paint.Style.***STROKE***);  
 **drawPaint**.setStrokeJoin(Paint.Join.***ROUND***);  
 **drawPaint**.setStrokeCap(Paint.Cap.***ROUND***);  
 **canvasPaint** = **new** Paint(Paint.***DITHER\_FLAG***);  
}

//Draw Doodle on Android

@Override  
**public boolean** onTouchEvent(MotionEvent event) {  
 **float** touchX = event.getX();  
 **float** touchY = event.getY();  
  
 **switch** (event.getAction()) {  
 **case** MotionEvent.***ACTION\_DOWN***:  
 **drawPath**.moveTo(touchX, touchY);  
 **break**;  
 **case** MotionEvent.***ACTION\_MOVE***:  
 **drawPath**.lineTo(touchX, touchY);  
 **break**;  
 **case** MotionEvent.***ACTION\_UP***:  
 **drawCanvas**.drawPath(**drawPath**, **drawPaint**);  
 **drawPath**.reset();  
 **break**;  
 **default**:  
 **return false**;  
 }  
  
 invalidate();  
 **return true**;  
}

//Send Doodle on Android

**public void** sendClicked(View view){  
 **doodleView**.setDrawingCacheEnabled(**true**);  
 Bitmap bitmap = **doodleView**.getBitmap();  
 ByteArrayOutputStream stream = **new** ByteArrayOutputStream();  
 bitmap.compress(Bitmap.CompressFormat.***PNG***, 100, stream);  
 **byte**[] byteArray = stream.toByteArray();  
 Message message = **new** Message(**"MSG"**, **"DOODLE"**, **groupID**, Connection.*username*,

byteArray, **"png"**);  
 message.**userID** = **this**.**userID**;  
 Connection.*sendMessage*(message);  
 finish();  
}