1-public class BluetoothCommuunication: {

In this class we have a lot of variables and functions that will help us in make and mange Bluetooth connection and control message delivery between connected devices.

We also need thread to listen to the income connection that coming from master device (android application) to slave device (Bluetooth ship). }

2-public BluetoothCommuunicationService(Context context, Handler handler ){

first we need blueCommService constructor to establish new Bluetooth data transfer with 2 parameter :

1-Context: the data that exist in each message we call it context

2-handler: send messages to user interface ,

}

3-private synchronized void updateUserInterfaceTitle() {

it will change based on connection status that will send it to handler }

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

public synchronized int getStatus() {

it int function that will return the status of our connection }

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

public synchronized void startConnection() {

we use this function to establish the connection between master and slave

we need to accept on thread that will be on the master(server) side to listen(server mode) on incoming connection request.

So we will cancel each thread that try to connect with our master(android)except our thread

Also we cancel any thread running connection with our master.

Then our thread will start server mode

Finally, we will update UI through handler

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

public synchronized void connect(BluetoothDevice device, boolean secure) {

|  |
| --- |
| Start the ConnectThread to establishe a connection to a remote device. |
| We have 2 parameters :  1-Blutooth Device :we identify here the device that we want to connect with,  2-Sceure :here will determine if our connection if secure or not  If (ture)Secure  If(False):not\_secure  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| public synchronized void connected(BluetoothSocket socket, BluetoothDevice |
| device, final String socketType) { |

|  |
| --- |
| \* Start the ConnectedThread to begin managing a Bluetooth connection |
|  |
| socket The Bluetooth Socket on which the connection was made |
| Cancel the thread that completed the connection  Cancel any thread currently running a connection and cancel the accept thread process because we only want to connect with one device |
| run the thread that manage the connection and perform transmissions  Send the name of the connected device back to the UI Activity  To Update UI title through handler |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

public synchronized void stop() {

Stop all threads

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

public void write(byte[] out) {

Write to the ConnectedThread in an unsynchronized manne

out The bytes to write

let’s see ConnectedThread#write(byte[])

we will Create temporary object

then Synchronize a copy of the ConnectedThread

finally we will Perform the write unsynchronized

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |
| --- |
| private void connectionFailed() { |
| Indicate that the connection attempt failed, in this situation we will  Send a failure message back to the user interface Activity  Through handler to update user interface title  Then we will Start the service again to restart listen step through our thread |
|  |

}

|  |
| --- |
| private void connectionLost() { |
| Indicate that the connection attempt lost, in this situation we will  Send a failure message back to the user interface Activity  Through handler to update user interface title  Then we will Start the service again to restart listen step through our thread |
|  |

}

private class AcceptThread extends Thread {

\* This thread runs while try to make an outgoing connection

\* with a device. Its already runs through; the connection may

\* succes or fails.

public AcceptThread(boolean secure) {

|  |
| --- |
| \* This thread will run while they are listening for incoming connections. It’s look  \* like a server-side client. It runs until a connection is accepted  \* (or until refused)  Create a new listening server socket  } |
|  |

}

public void run() {

here we try to Listen to the server socket the socket was free( not connected)

// This is a blocking call and will only return on a

// successful connection or an exception

// If a connection was accepted

// Either not ready or already connected. Terminate

}

public void cancel() {

close socket on client side

}

\_\_\_\_\_\_\_\_\_\_\_\_}\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ private class ConnectThread extends Thread {

/\*\*

\* This thread runs while trying to make an outgoing connection

\* with a device. It’s already runs through; the connection either

\* succeed or failed.

\*/

// Get a BluetoothSocket for a connection with the

// given BluetoothDevice

public void run() {

here we will cancel the discovery process because it will reduce the speed of connection

// Make a connection to the BluetoothSocket

// This is a blocking call and will only return on a

// successful connection or an exception

// Close the socket

// Reset the ConnectThread because we're done

// Start the connected thread

}

|  |
| --- |
| public void cancel() { |
| Close socket on server side |

}

private class ConnectThread extends Thread {

\* This thread already runs through a connection with a remote device.

\* It handles all incoming and outgoing data

Get the BluetoothSocket input and output streams

public void run() {

// here we will stay listen to Input stream to receive data from paired device

But we have to share any incoming message with user interface activity through handler

}

public void write(byte[] buffer) {

/\*\*

\* Write to the connected OutStream

We write on the buffer (array)

We send a copy of any sent message to user interface (UI) activity

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ public void cancel() {}

Close Bluetooth socket

}