Ćwiczenia 19 – Android studio – bluetooth

Na koniec zajęć prześlij pliki źródłowe (.xml, .java)+ obrazek do zasobu w teams.

1. Utwórz projekt o nazwie BlueTooth na podstawie Empty Activity, dobierz odpowiednie API (28 – Android 9).
2. Otwórz dokumentację:
   https://developer.android.com/guide/topics/permissions/overview
   https://developer.android.com/guide/topics/connectivity/bluetooth/permissions
   https://developer.android.com/guide/topics/connectivity/bluetooth
   https://developer.android.com/guide/topics/connectivity/bluetooth/setup
   https://developer.android.com/guide/topics/connectivity/bluetooth/find-bluetooth-devices
   https://developer.android.com/guide/topics/connectivity/bluetooth/transfer-data
3. Zadeklaruj potrzebne stałe, np.:

```
private static final String TAG = "bluetooth111";
private static final int MY_REQUEST_PERMISSION_BLUETOOTH_CONNECT = 1;
private static final int MY_REQUEST_PERMISSION_BLUETOOTH = 2;
private static final int MY_REQUEST_CODE_DISCOVARABLE = 3;
private static final int MY_REQUEST_PERMISSION_BLUETOOTH_SCAN = 4;
private static final int MY_REQUEST_PERMISSION_BLUETOOTH_ADMIN = 5;
private static final int MY_REQUEST_PERMISSION_BLUETOOTH_ADVERTISE = 6;
private static final int MY_REQUEST_PERMISSION_ACCESS_FINE = 7;
private static final int MY_REQUEST_PERMISSION_ACCESS_COARSE = 8;
private static final int MY_REQUEST_PERMISSION_ACCESS_BACKGROUND_LOCATION = 9;
```

```
BluetoothAdapter bluetoothAdapter;

private ListView listViewPairedDevices;
private ListView listViewNewDevices;
private ArrayAdapter<String> strArrayAdapterPD;
private ArrayAdapter<String> strArrayAdapterND;
ArrayList<String> listPD;
ArrayList<String> listND;
```

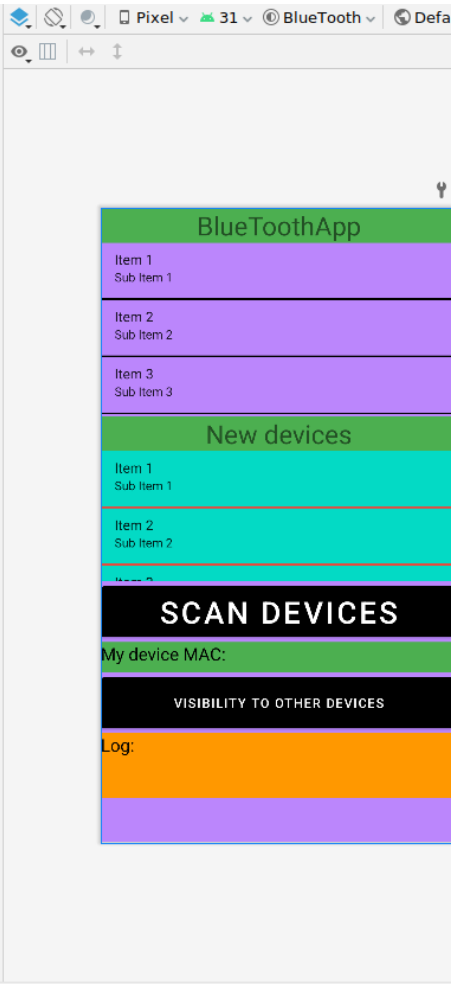
4. AndroidManifest.xml:

```
    <uses-feature android:name="android.hardware.bluetooth" android:required="false"/>

    <uses-permission android:name="android.permission.BLUETOOTH" />
    <!-- Needed only if your app looks for Bluetooth devices.
         If your app doesn't use Bluetooth scan results to derive physical
         location information, you can strongly assert that your app
         doesn't derive physical location. -->
    <uses-permission android:name="android.permission.BLUETOOTH_SCAN" />

    <!-- Needed only if your app makes the device discoverable to Bluetooth
         devices. -->
    <uses-permission android:name="android.permission.BLUETOOTH_ADVERTISE" />

    <!-- Needed only if your app communicates with already-paired Bluetooth
         devices. -->
    <uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />

    <!-- Needed only if your app uses Bluetooth scan results to derive physical location. -->
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />

    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-permission android:name="android.permission.ACCESS_BACKGROUND_LOCATION" />
```

```
<uses-permission
    android:name="android.permission.BLUETOOTH"
    android:maxSdkVersion="30" />
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
<uses-permission
    android:name="android.permission.BLUETOOTH_SCAN"
    android:usesPermissionFlags="neverForLocation" />

<uses-feature android:name="android.hardware.bluetooth" />
```

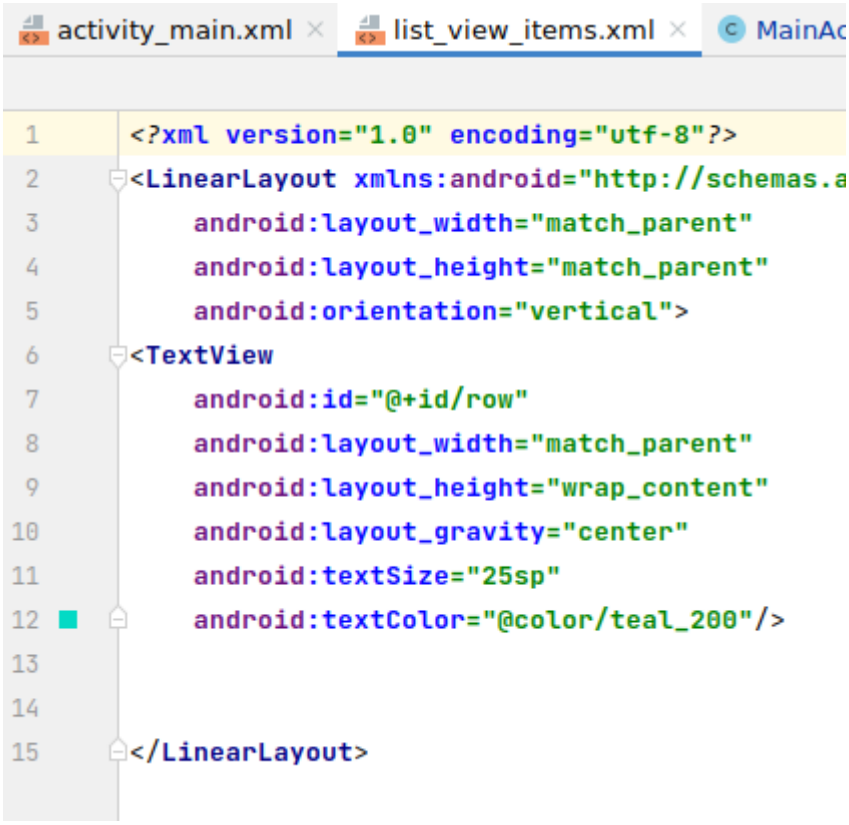
5. Przygotuj w activity_main.xml:

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="@color/purple_200"
    android:id="@+id/linearLayout"
    tools:context=".MainActivity">

<ScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent">
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/text_app"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="30sp"
        android:background="@color/green"
        android:textAlignment="center"
        android:text="BlueToothApp"
         />
    <ListView
        android:id="@+id/listPairedDevices"
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:background="@color/purple_200"
        android:divider="@color/black"
        android:dividerHeight="2dp"/>

    <TextView
```



```
        android:layout_height="70dp"
        android:textColor="@color/white"
        android:text="@string/scanDevice"
        android:textSize="35sp"
        android:backgroundTint="@color/black"
        />
    <TextView
        android:id="@+id/text_my_MAC"
        android:layout_width="match_parent"
        android:layout_height="35dp"
        android:textColor="@color/black"
        android:textSize="20sp"
        android:text="@string/myDeviceMac"
        android:background="@color/green"
        />
    <Button
        android:id="@+id/makeVisibleMyDevice"
        android:layout_width="match_parent"
        android:layout_height="70dp"
        android:textColor="@color/white"
        android:text="@string/makeVisibleMyDevice"
        android:textSize="15sp"
        android:backgroundTint="@color/black"
        />
    <TextView
        android:id="@+id/text_log"
        android:layout_width="match_parent"
        android:layout_height="75dp"
        android:textColor="@color/black"
        android:textSize="20sp"
        android:text="@string/log"
        android:background="@color/orange"
        />
    </LinearLayout>
</ScrollView>
</LinearLayout>
```

6. Utwórz nowy layout o nazwie list_view_items.xml



7. Dokończ tworzenie list dla sparowanych urządzeń i nowych urządzeń, np.:

```
strArrayAdapterPD = new ArrayAdapter<String>( context: this, R.layout.list_view_items, R.id.row, listPD);
listViewPairedDevices.setAdapter(strArrayAdapterPD);

strArrayAdapterND = new ArrayAdapter<String>( context: this, R.layout.list_view_items, R.id.row, listND);
listViewNewDevices.setAdapter(strArrayAdapterND);

bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
```

8. Jeśli bluetooth na urządzeniu jest wyłączony ustaw kolor czerwony na wybranych komponentach, a kolor niebieski jeśli bluetooth jest włączony, np.:

poniższy fragment wywołuje również okno z ustawieniami do włączenia przez użytkownika!!!

```
163         if (!bluetoothAdapter.isEnabled()) {
164             Log.v(TAG,  msg: "Bluetooth is OFF");
165             linearLayout.setBackgroundColor(getResources().getColor(R.color.red));
166             textApp.setBackgroundColor(getResources().getColor(R.color.red));
167             textNewDevices.setBackgroundColor(getResources().getColor(R.color.red));
168
169             Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
170             startActivityForResult(enableBtIntent, MY_REQUEST_PERMISSION_BLUETOOTH);
171
172         }else {
173             Log.v(TAG,  msg: "Bluetooth is ON");
174             linearLayout.setBackgroundColor(getResources().getColor(R.color.bluetooth));
175             textApp.setBackgroundColor(getResources().getColor(R.color.bluetooth));
176             textNewDevices.setBackgroundColor(getResources().getColor(R.color.bluetooth));
177         }
```

9. Napisz metodę sprawdzającą dostępność bluetooth na urządzeniu, fragment poniżej:

```
432     private void checkBlueToothAvailable() {
433         // Use this check to determine whether Bluetooth classic is supported on the device.
434         // Then you can selectively disable BLE-related features.
435         if (!getPackageManager().hasSystemFeature(PackageManager.FEATURE_BLUETOOTH)) {
436             Toast.makeText( context: this, R.string.bluetooth_not_supported, Toast.LENGTH_SHORT).show();
```

10. Zaimplementuj metody onResume(), onPause() itd.:.
11. Zaimplementuj także metodę onDestroy(), np.:

```
@Override
protected void onDestroy() {
    super.onDestroy();
    Log.v(TAG,  msg: "onDestroy");
    // unregister the receivers
    unregisterReceiver(receiver);
    unregisterReceiver(receiver2);
    unregisterReceiver(receiver3);
    // Make sure we're not doing discovery anymore
    if (bluetoothAdapter != null) {
       if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.BLUETOOTH_SCAN)
               != PackageManager.PERMISSION_GRANTED) {
           ActivityCompat.requestPermissions( activity: this, new String[]{Manifest.permission.BLUETOOTH_SCAN},
                   MY_REQUEST_PERMISSION_BLUETOOTH_SCAN);
         return;
       }
       bluetoothAdapter.cancelDiscovery();
    }
}
```

12. Utwórz listę sparowanych urządzeń:

```
private void listAllPairedDevices() {

    checkAllPermissions();
    if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.BLUETOOTH_CONNECT) != PackageManager.P
    Set<BluetoothDevice> pairedDevices = bluetoothAdapter.getBondedDevices();

    if (pairedDevices.size() > 0) {
        // There are paired devices. Get the name and address of each paired device.
        for (BluetoothDevice device : pairedDevices) {

            String deviceName = device.getName();
            String deviceHardwareAddress = device.getAddress(); // MAC address
            Log.v(TAG,  msg: "deviceName=" + deviceName + " " + "deviceHardwareAddress= " + deviceHardwareAddress);
            listPD.add(device.getName() + " " + device.getAddress());
        }
    } else {
        Log.v(TAG,  msg: "pairedDevices.size()=" + pairedDevices.size());
        listPD.add("no devices found");
    }
}
```

13. Przetestuj aplikację, uruchom na urządzeniu. Wyświetl listę sparowanych urządzeń.

14. Stwórz metodę wyszukującą nowe urządzenia, poniżej przykładowy szkielet (patrz następny punkt):

```
Log.d(TAG, msg: "Scanning for 12 seconds ...");
setTitle(R.string.scanning);

//  checkAllPermissions();
// Request discover from BluetoothAdapter
        bluetoothAdapter.startDiscovery();
if (bluetoothAdapter.isDiscovering()) {
   // bluetoothAdapter.cancelDiscovery();
    Log.v(TAG, msg: "find device");
} else {
    Log.v(TAG, msg: "not discovering, not find device");
}
```

15. Zarejestruj receiver:

```
//  checkAllPermissions();
// Request discover from BluetoothAdapter
// Register for broadcasts when a device is discovered.
IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
filter.addAction(BluetoothAdapter.ACTION_DISCOVERY_STARTED);
filter.addAction(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
registerReceiver(receiver, filter);
Log.v(TAG, msg: "1 bluetoothAdapter.getState()="+bluetoothAdapter.getState()+
        " bluetoothAdapter.getScanMode()="+bluetoothAdapter.getScanMode());
```

```
    // Create a BroadcastReceiver for ACTION_FOUND.
    private final BroadcastReceiver receiver = new BroadcastReceiver() {
        public void onReceive(Context context, Intent intent) {
            String action = intent.getAction();

            if (BluetoothDevice.ACTION_FOUND.equals(action)) {
                // Discovery has found a device. Get the BluetoothDevice
                // object and its info from the Intent.
```

```
    BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
    String deviceName = device.getName();
    String deviceHardwareAddress = device.getAddress(); // MAC address
    Log.v(TAG, msg: "BroadcastReceiver -------> deviceName=" + deviceName + " " + "deviceHardwareAddr
    // add new device to list
    listND.add(deviceName +" "+deviceHardwareAddress);

    if (BluetoothAdapter.ACTION_DISCOVERY_STARTED.equalsIgnoreCase(action)) {
        Log.v(TAG, msg: "BroadcastReceiver -------> BluetoothAdapter.ACTION_DISCOVERY_STARTED");
    }
    if (BluetoothAdapter.ACTION_DISCOVERY_FINISHED.equalsIgnoreCase(action)) {

        Log.v(TAG, msg: "BroadcastReceiver -------> BluetoothAdapter.ACTION_DISCOVERY_FINISHED");
        listND.add(" no new device found");
        //scanResult();
    }
}
```

16. Przetestuj wykrywalność nowych urządzeń:

17. Widoczność urządzenia dla innych urządzeń:

```
// makeVisibleForAnotherDevices();

Intent discoverableIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
discoverableIntent.putExtra(BluetoothAdapter.EXTRA_DISCOVERABLE_DURATION, value: 180);
startActivityForResult(discoverableIntent, MY_REQUEST_CODE_DISCOVARABLE);
textLog.setText("visible for 180 seconds");
```

18. Dodaj receiver podający stany:

```
IntentFilter filter2 = new IntentFilter(BluetoothAdapter.ACTION_STATE_CHANGED);
registerReceiver(receiver2, filter2);
```

```
// Create a BroadcastReceiver for ACTION_STATE_CHANGED
private final BroadcastReceiver receiver2 = new BroadcastReceiver() {
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();

        if (action.equals(BluetoothAdapter.ACTION_STATE_CHANGED)) {
            final int state = intent.getIntExtra(BluetoothAdapter.EXTRA_STATE, BluetoothAdapter.ERROR);
            switch (state) {
                case BluetoothAdapter.STATE_OFF:
                    Log.v(TAG, msg: "BroadcastReceiver2 -------> BluetoothAdapter.STATE_OFF=" + BluetoothAdapter.STATE_OFF);
                    linearLayout.setBackgroundColor(getResources().getColor(R.color.red));
                    textApp.setBackgroundColor(getResources().getColor(R.color.red));
                    textNewDevices.setBackgroundColor(getResources().getColor(R.color.red));
                    break;
                case BluetoothAdapter.STATE_TURNING_OFF:
                    Log.v(TAG, msg: "BroadcastReceiver2 -------> BluetoothAdapter.STATE_TURNING_OFF=" + BluetoothAdapter.STATE_TURNING_OFF);
                    break;
                case BluetoothAdapter.STATE_ON:
                    Log.v(TAG, msg: "BroadcastReceiver2 -------> BluetoothAdapter.STATE_ON=" + BluetoothAdapter.STATE_ON);
                    linearLayout.setBackgroundColor(getResources().getColor(R.color.bluetooth));
                    textApp.setBackgroundColor(getResources().getColor(R.color.bluetooth));
                    textNewDevices.setBackgroundColor(getResources().getColor(R.color.bluetooth));
                    break;
                case BluetoothAdapter.STATE_TURNING_ON:
                    Log.v(TAG, msg: "BroadcastReceiver2 -------> BluetoothAdapter.STATE_TURNING_ON=" + BluetoothAdapter.STATE_TURNING_ON);
                    break;
            }

        }
    }
};
```

19. Dodaj receiver dla zmiany stanu skanowania:

IntentFilter:

```
IntentFilter filter3 = new IntentFilter();
filter3.addAction(BluetoothAdapter.ACTION_DISCOVERY_STARTED);
filter3.addAction(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
filter3.addAction(BluetoothAdapter.ACTION_SCAN_MODE_CHANGED);
registerReceiver(receiver3, filter3);
```

Receiver:

```
// Create a BroadcastReceiver for ACTION_SCAN_MODE_CHANGED
private final BroadcastReceiver receiver3 = new BroadcastReceiver() {
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();

        if (action.equals(BluetoothAdapter.ACTION_SCAN_MODE_CHANGED)) {
            int mode = intent.getIntExtra(BluetoothAdapter.EXTRA_SCAN_MODE, BluetoothAdapter.ERROR);
            switch (mode) {
                case BluetoothAdapter.SCAN_MODE_CONNECTABLE_DISCOVERABLE:
                    Log.v(TAG, msg: "BroadcastReceiver3 -------> BluetoothAdapter.SCAN_MODE_CONNECTABLE_DISCO
                            + BluetoothAdapter.SCAN_MODE_CONNECTABLE_DISCOVERABLE);
                    break;
                case BluetoothAdapter.SCAN_MODE_CONNECTABLE:
                    Log.v(TAG, msg: "BroadcastReceiver3 -------> BluetoothAdapter.SCAN_MODE_CONNECTABLE="
                            + BluetoothAdapter.SCAN_MODE_CONNECTABLE);
                    break;
                case BluetoothAdapter.SCAN_MODE_NONE:
                    Log.v(TAG, msg: "BroadcastReceiver3 -------> BluetoothAdapter.SCAN_MODE_NONE="
                            + BluetoothAdapter.SCAN_MODE_NONE);
                    break;

            }

        }
    }
};
```

20. Część druga: napisanie komunikatora tekstowego po bluetooth.
https://developer.android.com/guide/topics/connectivity/bluetooth/connect-bluetooth-devices

Example

The following is a basic example of a client thread that initiates a Bluetooth connection:

Kotlin Java

```
private class ConnectThread extends Thread {
    private final BluetoothSocket mmSocket;
    private final BluetoothDevice mmDevice;

    public ConnectThread(BluetoothDevice device) {
        // Use a temporary object that is later assigned to mmSocket
        // because mmSocket is final.
        BluetoothSocket tmp = null;
        mmDevice = device;

        try {
            // Get a BluetoothSocket to connect with the given BluetoothDevice.
            // MY_UUID is the app's UUID string, also used in the server code.
            tmp = device.createRfcommSocketToServiceRecord(MY_UUID);
        } catch (IOException e) {
            Log.e(TAG, "Socket's create() method failed", e);
        }
        mmSocket = tmp;
    }

    public void run() {
        // Cancel discovery because it otherwise slows down the connection.
        bluetoothAdapter.cancelDiscovery();

        try {
            // Connect to the remote device through the socket. This call blocks
            // until it succeeds or throws an exception.
            mmSocket.connect();
```

```
            Log.e(TAG, "Could not close the client socket", closeException);
        }
        return;
    }

    // The connection attempt succeeded. Perform work associated with
    // the connection in a separate thread.
    manageMyConnectedSocket(mmSocket);
}

// Closes the client socket and causes the thread to finish.
public void cancel() {
    try {
        mmSocket.close();
    } catch (IOException e) {
```

https://developer.android.com/guide/topics/connectivity/bluetooth/transfer-data

```
// Call this from the main activity to send data to the remote device.
public void write(byte[] bytes) {
    try {
        mmOutStream.write(bytes);

        // Share the sent message with the UI activity.
        Message writtenMsg = handler.obtainMessage(
                MessageConstants.MESSAGE_WRITE, -1, -1, mmBuffer);
        writtenMsg.sendToTarget();
    } catch (IOException e) {
        Log.e(TAG, "Error occurred when sending data", e);

        // Send a failure message back to the activity.
        Message writeErrorMsg =
                handler.obtainMessage(MessageConstants.MESSAGE_TOAST);
        Bundle bundle = new Bundle();
        bundle.putString("toast",
                "Couldn't send data to the other device");
        writeErrorMsg.setData(bundle);
        handler.sendMessage(writeErrorMsg);
    }
```

```java
public class MyBluetoothService extends ActivityCompat{

    // Debugging

    private static final String TAG = "MY_APP_DEBUG_TAG";
    private Handler handler; // handler that gets info from Bluetooth service

    // Defines several constants used when transmitting messages between the
    // service and the UI.
    private interface MessageConstants {
        public static final int MESSAGE_READ = 0;
        public static final int MESSAGE_WRITE = 1;
        public static final int MESSAGE_TOAST = 2;

        // ... (Add other message types here as needed.)
    }

    private class AcceptThread extends Thread {
        private BluetoothServerSocket mmServerSocket;
        BluetoothAdapter bluetoothAdapter;
        // Name for the SDP record when creating server socket
        private static final String NAME = "BluetoothChat";
        // Unique UUID for this application
        private final UUID MY_UUID = UUID.fromString("fa87c0d0-afac-11de-8a39-0800200c9a66");
```

```
90          // Closes the connect socket and causes the thread to finish.
91          public void cancel() {
92              try {
93                  mmServerSocket.close();
94              } catch (IOException e) {
95                  Log.e(TAG,  msg: "Could not close the connect socket", e);
96              }
97          }
98      }
99          private class ConnectedThread extends Thread {
00              private final BluetoothSocket mmSocket;
01              private final InputStream mmInStream;
02              private final OutputStream mmOutStream;
03              private byte[] mmBuffer; // mmBuffer store for the stream
04
05  @           public ConnectedThread(BluetoothSocket socket) {
06                  mmSocket = socket;
07                  InputStream tmpIn = null;
08                  OutputStream tmpOut = null;
09
10                  // Get the input and output streams; using temp objects because
11                  // member streams are final.
12                  try {
13                      tmpIn = socket.getInputStream();
14                  } catch (IOException e) {
15                      Log.e(TAG,  msg: "Error occurred when creating input stream", e);
16                  }
17                  try {
18                      tmpOut = socket.getOutputStream();
19                  } catch (IOException e) {
20                      Log.e(TAG,  msg: "Error occurred when creating output stream", e);
21                  }
```

21. Wykonaj zadania
    a) parowanie urządzeń towarzyszących:

    https://developer.android.com/guide/topics/connectivity/companion-device-pairing

    b) przetestuj działanie komunikatora

    c) dodaj przycisk zatrzymujący wyszukiwanie innych urządzeń

22. KONIEC.