Android i Bluetooth Low Energy

Laboratorium 9

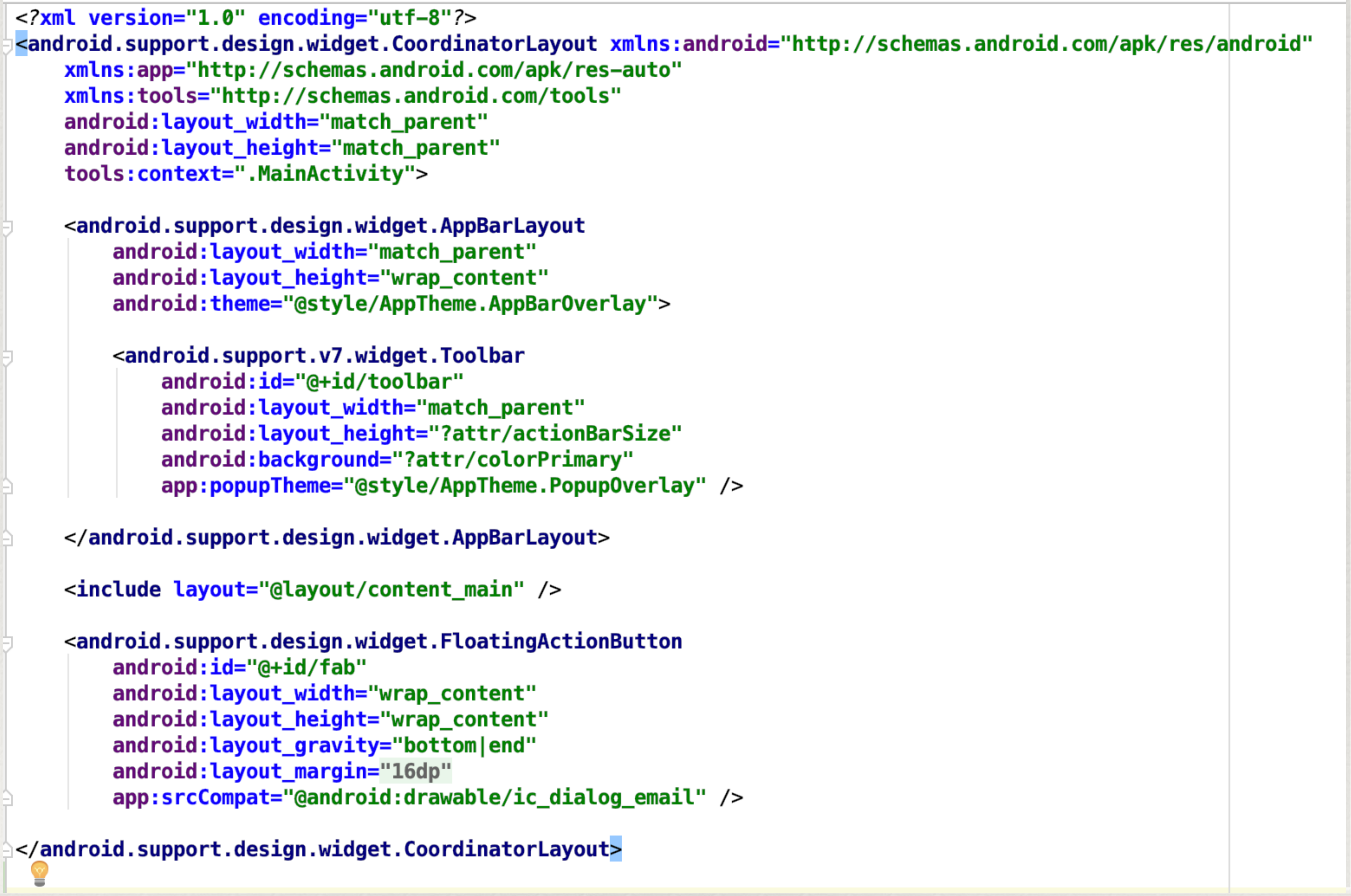
Agenda

- *Stworzymy dwie aplikacje działające jako klient-serwer komunikujące się poprzez Bluetooth LE*
- *Ze względu na ograniczenia emulatora potrzebne będą państwa telefony*
- *Nie każdy telefon posiada BLE*

Klient

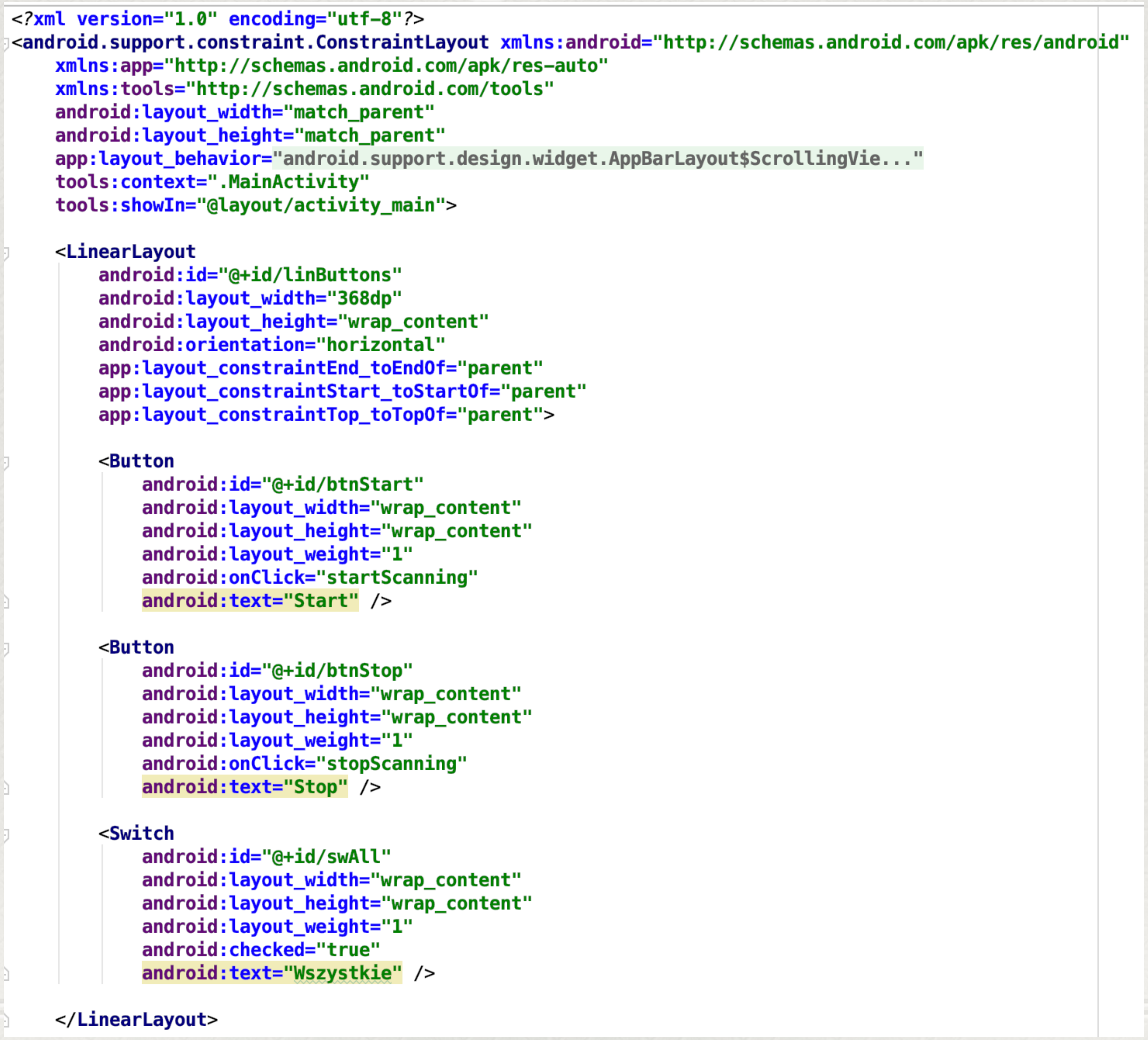
- *Tworzymy nowy projekt typu Basic Activity o nazwie BLEClient z API na poziomie Androida 6.*
- *Projekt aktywności na następnych slajdach*

activity_main.xml

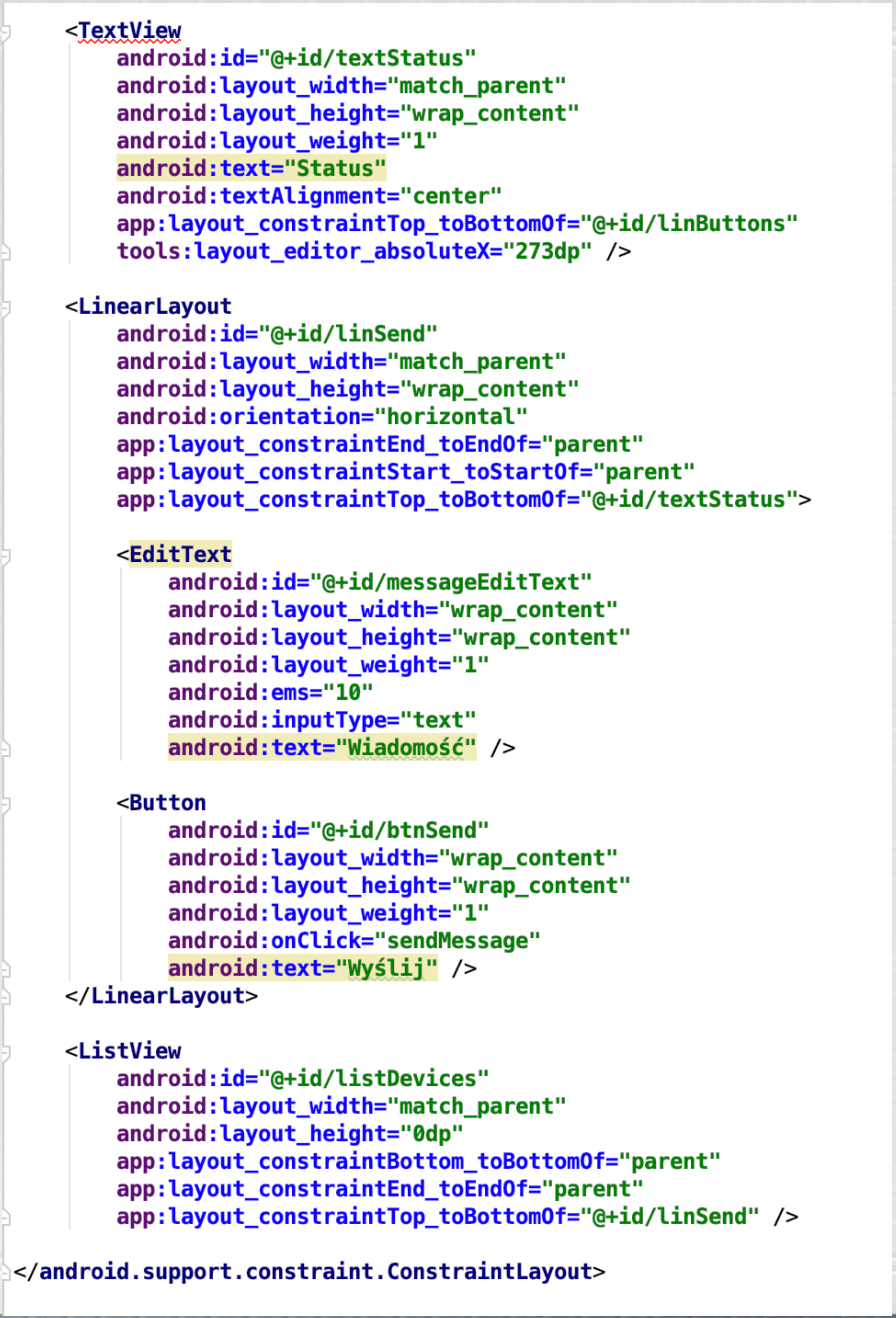


```xml
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.AppBarOverlay">

        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay" />

    </android.support.design.widget.AppBarLayout>

    <include layout="@layout/content_main" />

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="16dp"
        app:srcCompat="@android:drawable/ic_dialog_email" />

</android.support.design.widget.CoordinatorLayout>
```

content_main.xml cz. 1



```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout 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"
    app:layout_behavior="android.support.design.widget.AppBarLayout$ScrollingVie..."
    tools:context=".MainActivity"
    tools:showIn="@layout/activity_main">

    <LinearLayout
        android:id="@+id/linButtons"
        android:layout_width="368dp"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <Button
            android:id="@+id/btnStart"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:onClick="startScanning"
            android:text="Start" />

        <Button
            android:id="@+id/btnStop"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:onClick="stopScanning"
            android:text="Stop" />

        <Switch
            android:id="@+id/swAll"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:checked="true"
            android:text="Wszystkie" />

    </LinearLayout>
```

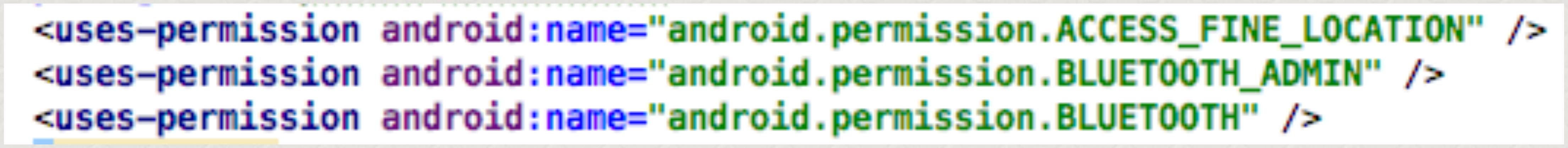


```
    <TextView
        android:id="@+id/textStatus"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Status"
        android:textAlignment="center"
        app:layout_constraintTop_toBottomOf="@+id/linButtons"
        tools:layout_editor_absoluteX="273dp" />

    <LinearLayout
        android:id="@+id/linSend"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textStatus">

        <EditText
            android:id="@+id/messageEditText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:ems="10"
            android:inputType="text"
            android:text="Wiadomość" />

        <Button
            android:id="@+id/btnSend"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:onClick="sendMessage"
            android:text="Wyślij" />
    </LinearLayout>

    <ListView
        android:id="@+id/listDevices"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/linSend" />

</android.support.constraint.ConstraintLayout>
```

content_main.xml cz. 2

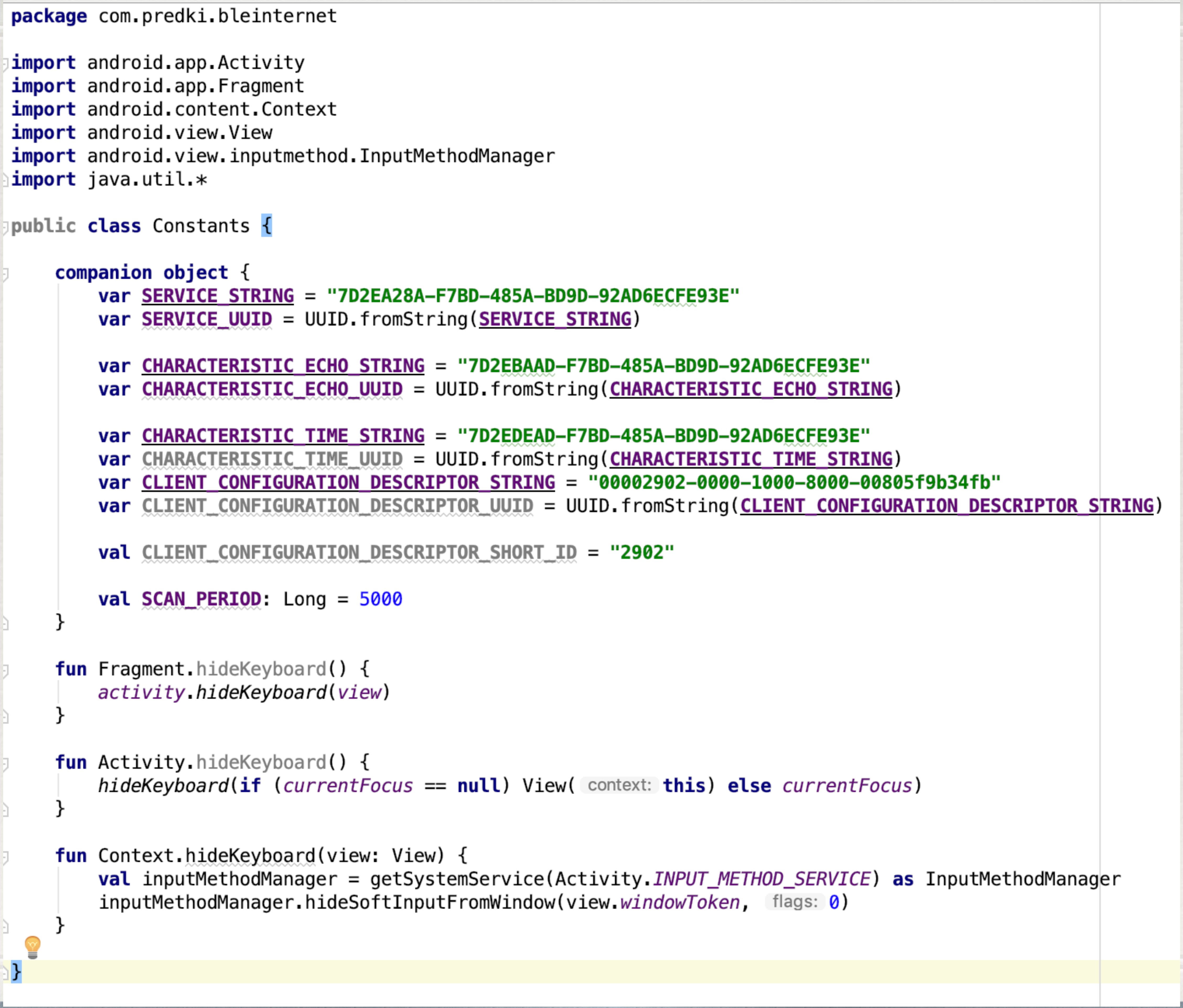
Klient

- *Do manifestu dodajemy wpisy:*



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

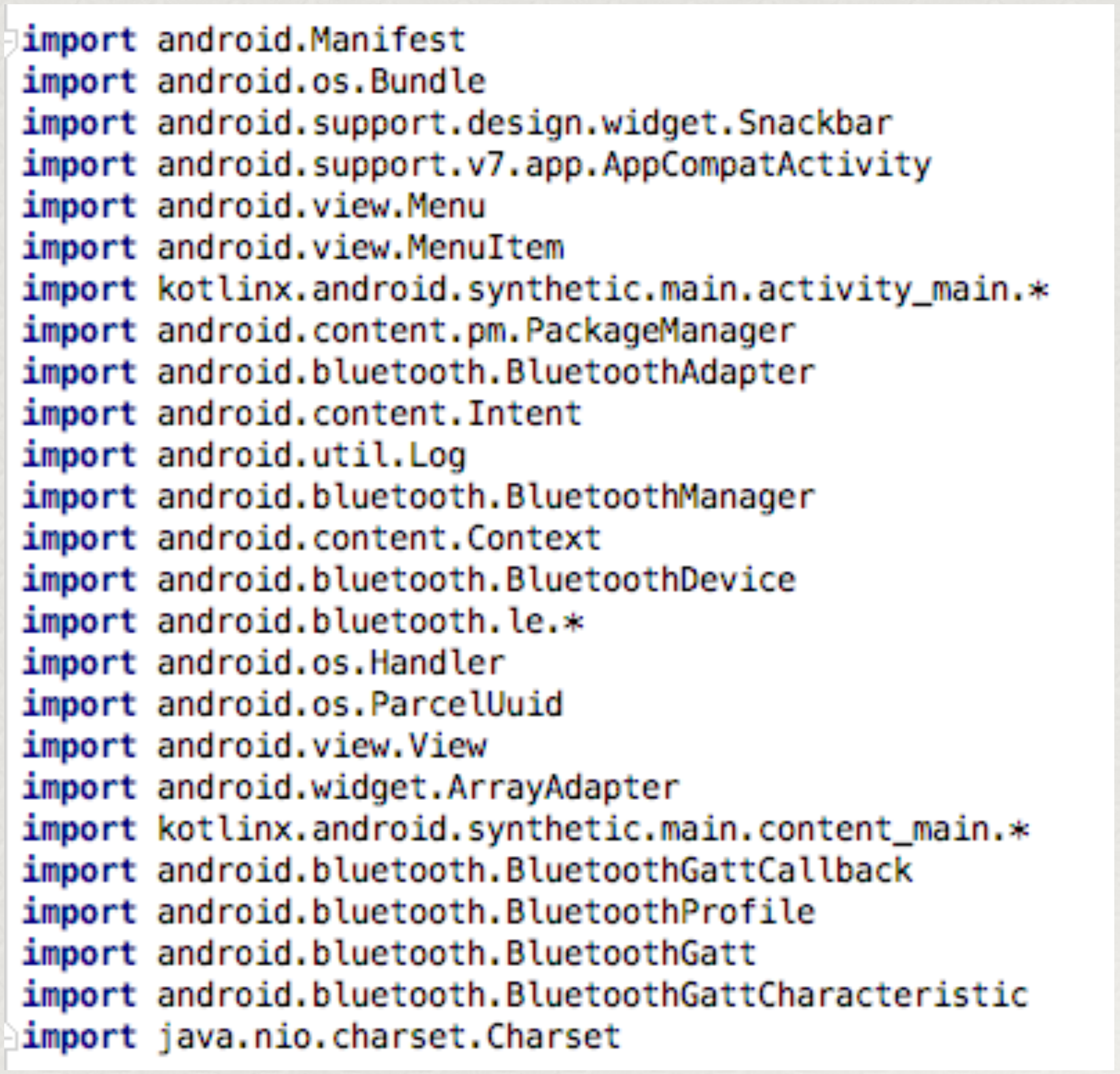
- *Do projektu dodajemy klasę Constants, której kod jest na następnym slajdzie*



```
package com.predki.bleinternet

import android.app.Activity
import android.app.Fragment
import android.content.Context
import android.view.View
import android.view.inputmethod.InputMethodManager
import java.util.*

public class Constants {

    companion object {
        var SERVICE_STRING = "7D2EA28A-F7BD-485A-BD9D-92AD6ECFE93E"
        var SERVICE_UUID = UUID.fromString(SERVICE_STRING)

        var CHARACTERISTIC_ECHO_STRING = "7D2EBAAD-F7BD-485A-BD9D-92AD6ECFE93E"
        var CHARACTERISTIC_ECHO_UUID = UUID.fromString(CHARACTERISTIC_ECHO_STRING)

        var CHARACTERISTIC_TIME_STRING = "7D2EDEAD-F7BD-485A-BD9D-92AD6ECFE93E"
        var CHARACTERISTIC_TIME_UUID = UUID.fromString(CHARACTERISTIC_TIME_STRING)
        var CLIENT_CONFIGURATION_DESCRIPTOR_STRING = "00002902-0000-1000-8000-00805f9b34fb"
        var CLIENT_CONFIGURATION_DESCRIPTOR_UUID = UUID.fromString(CLIENT_CONFIGURATION_DESCRIPTOR_STRING)

        val CLIENT_CONFIGURATION_DESCRIPTOR_SHORT_ID = "2902"

        val SCAN_PERIOD: Long = 5000
    }

    fun Fragment.hideKeyboard() {
        activity.hideKeyboard(view)
    }

    fun Activity.hideKeyboard() {
        hideKeyboard(if (currentFocus == null) View( context: this) else currentFocus)
    }

    fun Context.hideKeyboard(view: View) {
        val inputMethodManager = getSystemService(Activity.INPUT_METHOD_SERVICE) as InputMethodManager
        inputMethodManager.hideSoftInputFromWindow(view.windowToken,  flags: 0)
    }

}
```

Klient

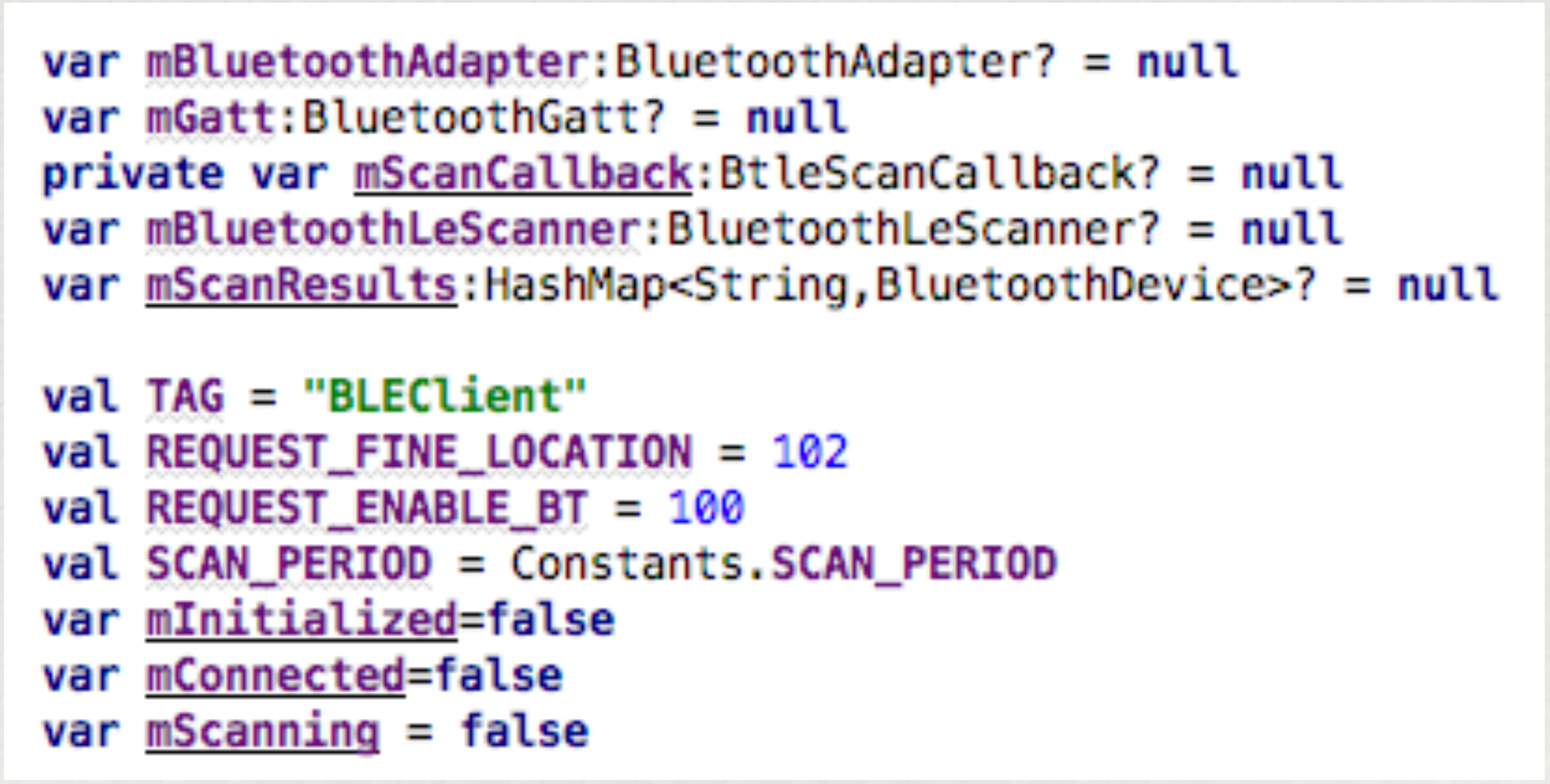
- *Będziemy potrzebować wiele importów (powinny dodawać się automatycznie)*



```
import android.Manifest
import android.os.Bundle
import android.support.design.widget.Snackbar
import android.support.v7.app.AppCompatActivity
import android.view.Menu
import android.view.MenuItem
import kotlinx.android.synthetic.main.activity_main.*
import android.content.pm.PackageManager
import android.bluetooth.BluetoothAdapter
import android.content.Intent
import android.util.Log
import android.bluetooth.BluetoothManager
import android.content.Context
import android.bluetooth.BluetoothDevice
import android.bluetooth.le.*
import android.os.Handler
import android.os.ParcelUuid
import android.view.View
import android.widget.ArrayAdapter
import kotlinx.android.synthetic.main.content_main.*
import android.bluetooth.BluetoothGattCallback
import android.bluetooth.BluetoothProfile
import android.bluetooth.BluetoothGatt
import android.bluetooth.BluetoothGattCharacteristic
import java.nio.charset.Charset
```

Klient

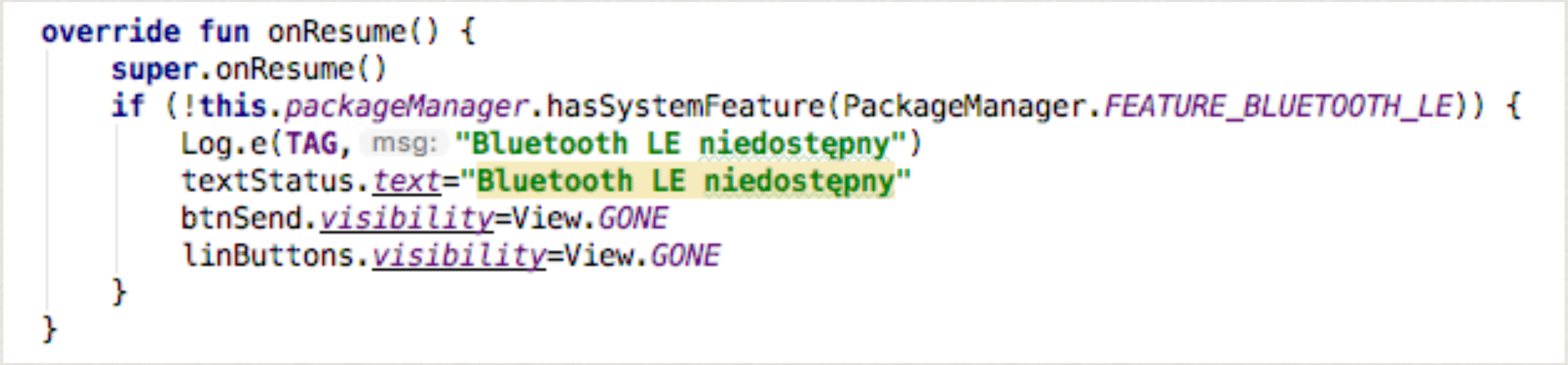
- *W klasie zdefiniujemy pola:*



```
var mBluetoothAdapter:BluetoothAdapter? = null
var mGatt:BluetoothGatt? = null
private var mScanCallback:BtleScanCallback? = null
var mBluetoothLeScanner:BluetoothLeScanner? = null
var mScanResults:HashMap<String,BluetoothDevice>? = null

val TAG = "BLEClient"
val REQUEST_FINE_LOCATION = 102
val REQUEST_ENABLE_BT = 100
val SCAN_PERIOD = Constants.SCAN_PERIOD
var mInitialized=false
var mConnected=false
var mScanning = false
```

Klient

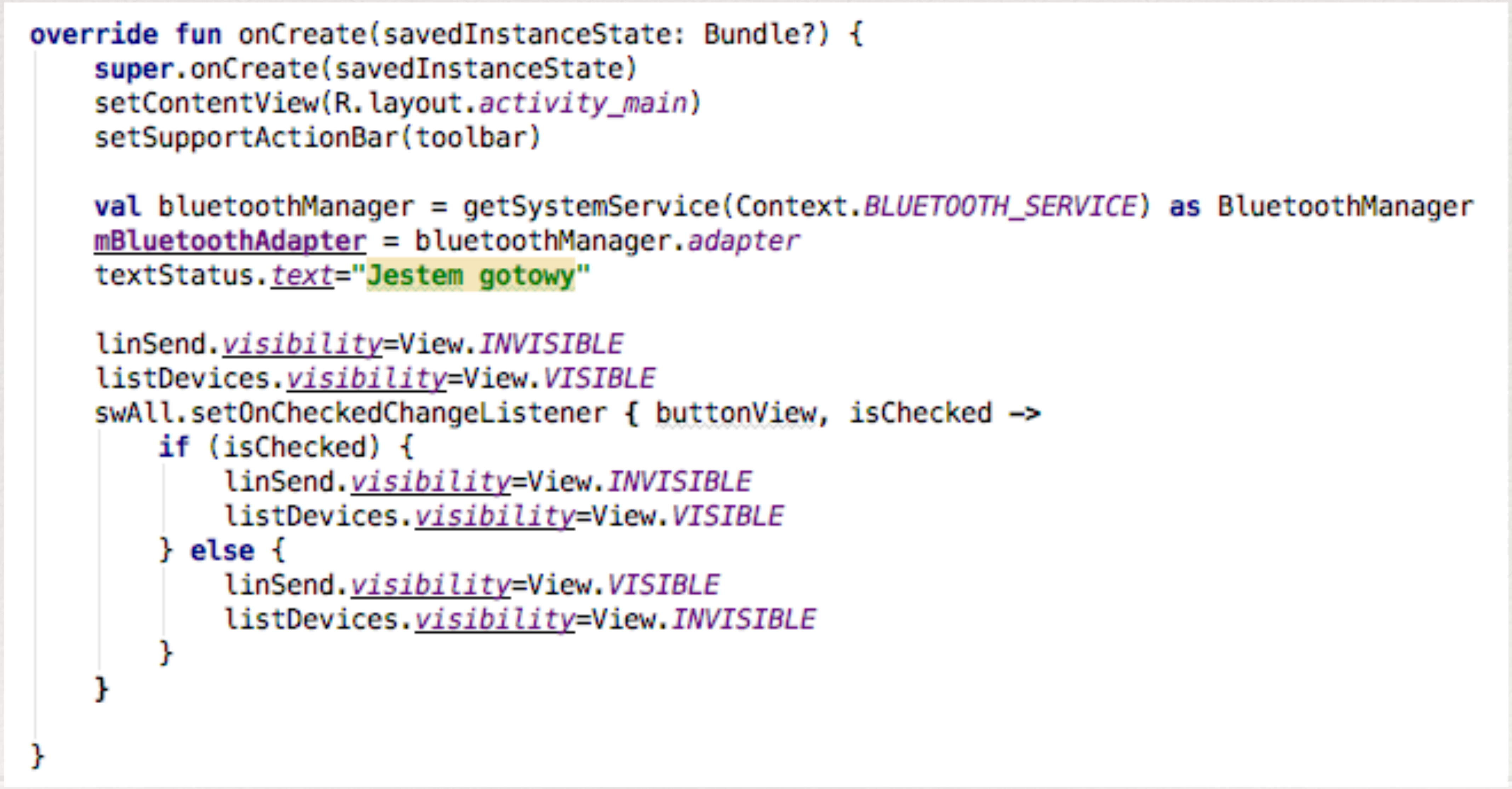
- *W metodzie onResume sprawdzimy, czy BLE jest dostępne*



```
override fun onResume() {
    super.onResume()
    if (!this.packageManager.hasSystemFeature(PackageManager.FEATURE_BLUETOOTH_LE)) {
        Log.e(TAG, msg: "Bluetooth LE niedostępny")
        textStatus.text="Bluetooth LE niedostępny"
        btnSend.visibility=View.GONE
        linButtons.visibility=View.GONE
    }
}
```

Klient

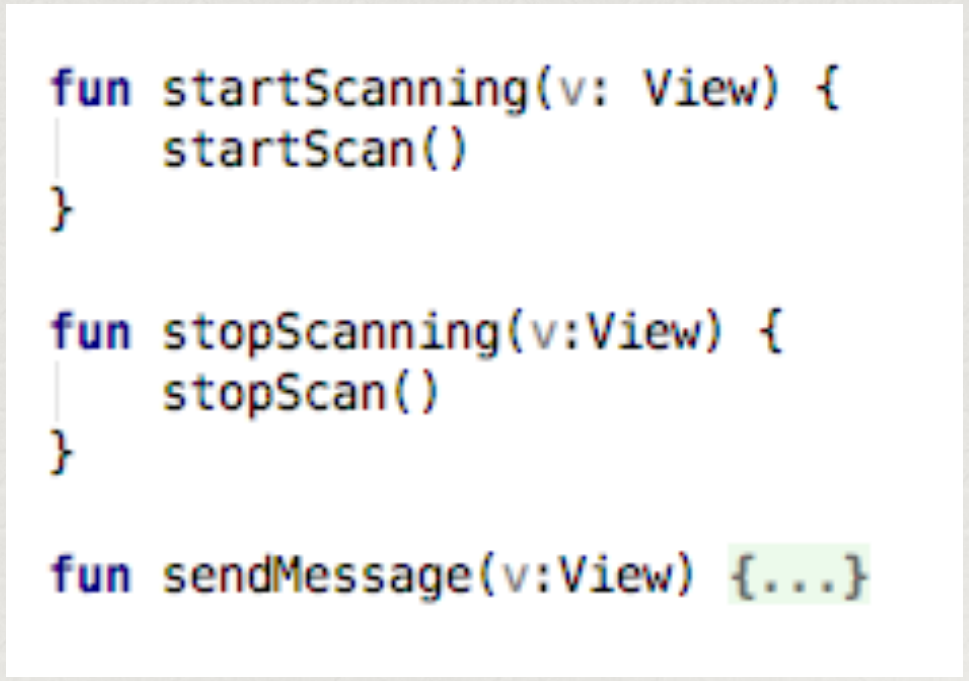
- *Modyfikujemy metodę onCreate*



```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    setSupportActionBar(toolbar)

    val bluetoothManager = getSystemService(Context.BLUETOOTH_SERVICE) as BluetoothManager
    mBluetoothAdapter = bluetoothManager.adapter
    textStatus.text="Jestem gotowy"

    linSend.visibility=View.INVISIBLE
    listDevices.visibility=View.VISIBLE
    swAll.setOnCheckedChangeListener { buttonView, isChecked ->
        if (isChecked) {
            linSend.visibility=View.INVISIBLE
            listDevices.visibility=View.VISIBLE
        } else {
            linSend.visibility=View.VISIBLE
            listDevices.visibility=View.INVISIBLE
        }
    }

}
```

Klient

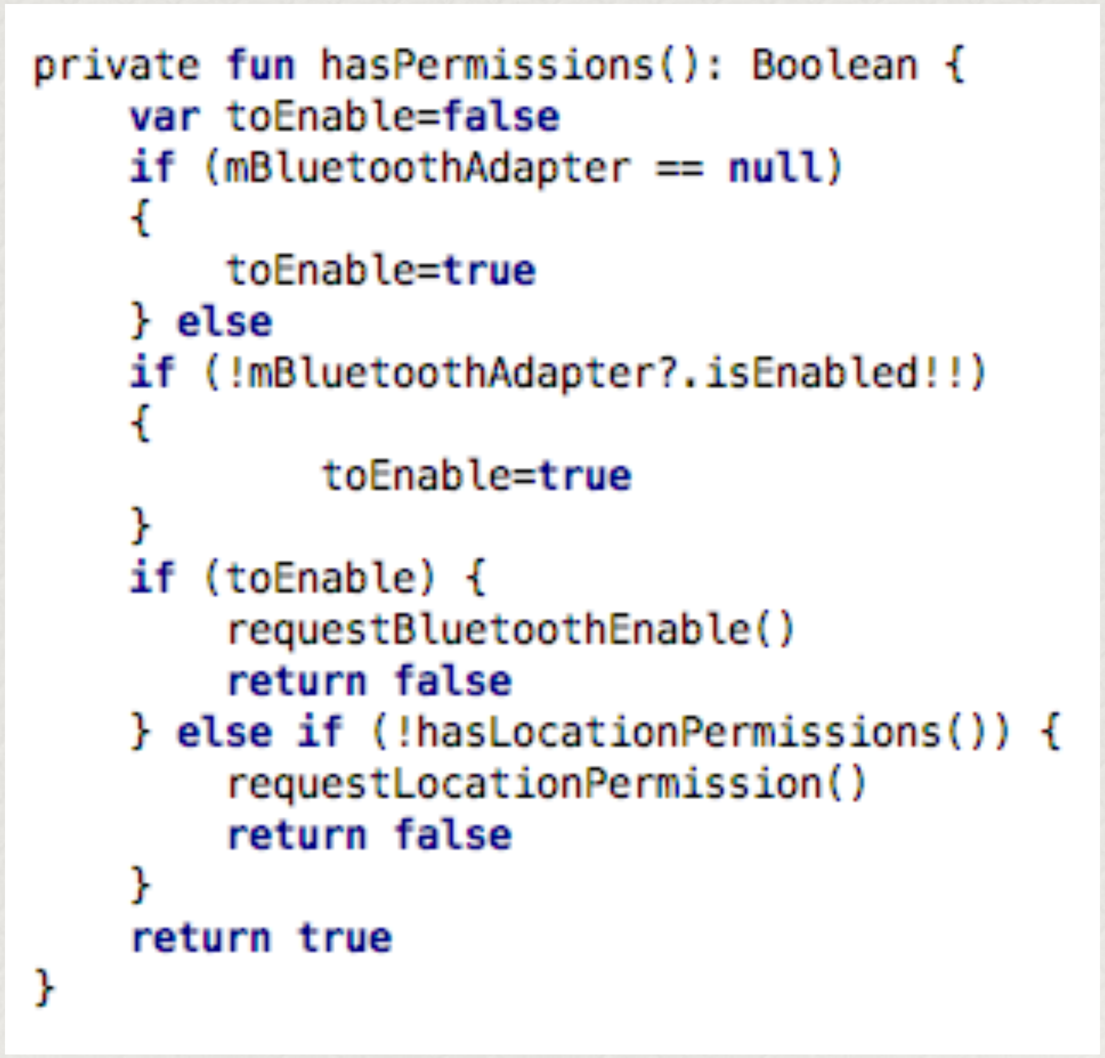
- *Dodajemy metody obsługi przycisków Start/Stop - odpowiednio startScanning i stopScanning oraz dla przycisku Wyślij - sendMessage*



```
fun startScanning(v: View) {
    startScan()
}

fun stopScanning(v:View) {
    stopScan()
}

fun sendMessage(v:View) {...}
```

Klient

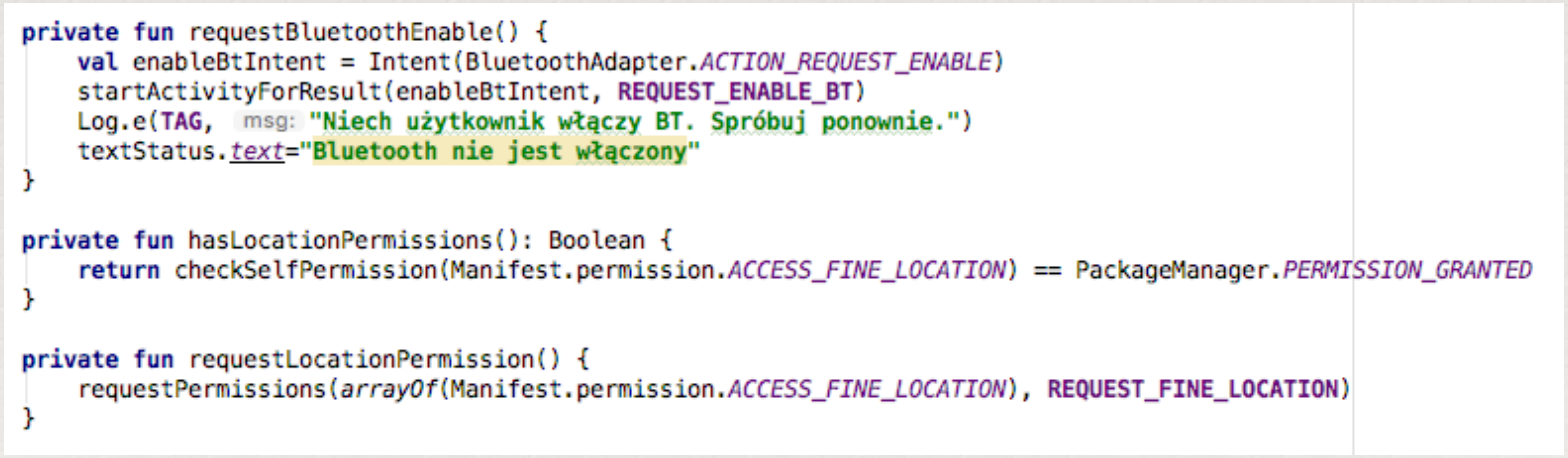
- *Dodajemy metodę sprawdzającą, czy mamy uprawnienia*



```
private fun hasPermissions(): Boolean {
    var toEnable=false
    if (mBluetoothAdapter == null)
    {
        toEnable=true
    } else
    if (!mBluetoothAdapter?.isEnabled!!)
    {
            toEnable=true
    }
    if (toEnable) {
        requestBluetoothEnable()
        return false
    } else if (!hasLocationPermissions()) {
        requestLocationPermission()
        return false
    }
    return true
}
```

Klient

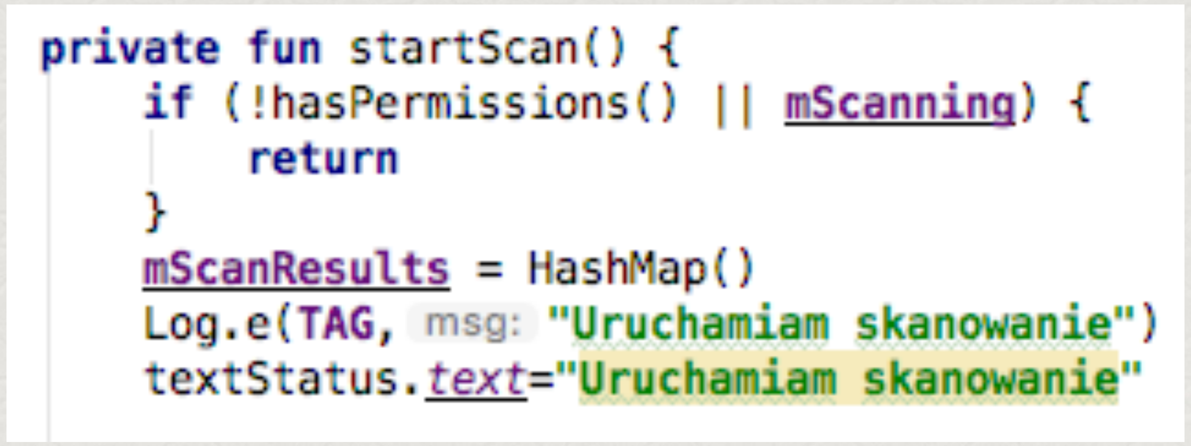
- *I jej metody pomocnicze*



```
private fun requestBluetoothEnable() {
    val enableBtIntent = Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE)
    startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT)
    Log.e(TAG,  msg: "Niech użytkownik włączy BT. Spróbuj ponownie.")
    textStatus.text="Bluetooth nie jest włączony"
}

private fun hasLocationPermissions(): Boolean {
    return checkSelfPermission(Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED
}

private fun requestLocationPermission() {
    requestPermissions(arrayOf(Manifest.permission.ACCESS_FINE_LOCATION), REQUEST_FINE_LOCATION)
}
```

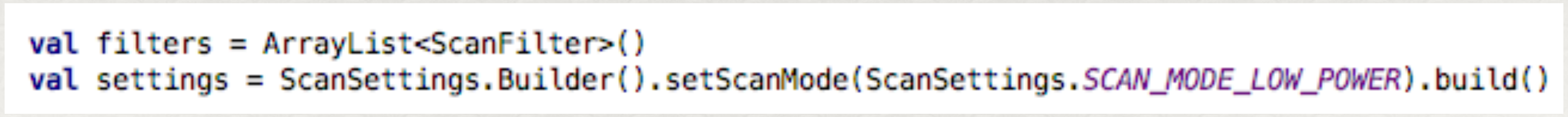
Klient

- *Zaczynamy pisać metodę startScan*



```
private fun startScan() {
    if (!hasPermissions() || mScanning) {
        return
    }
    mScanResults = HashMap()
    Log.e(TAG, msg: "Uruchamiam skanowanie")
    textStatus.text="Uruchamiam skanowanie"
```

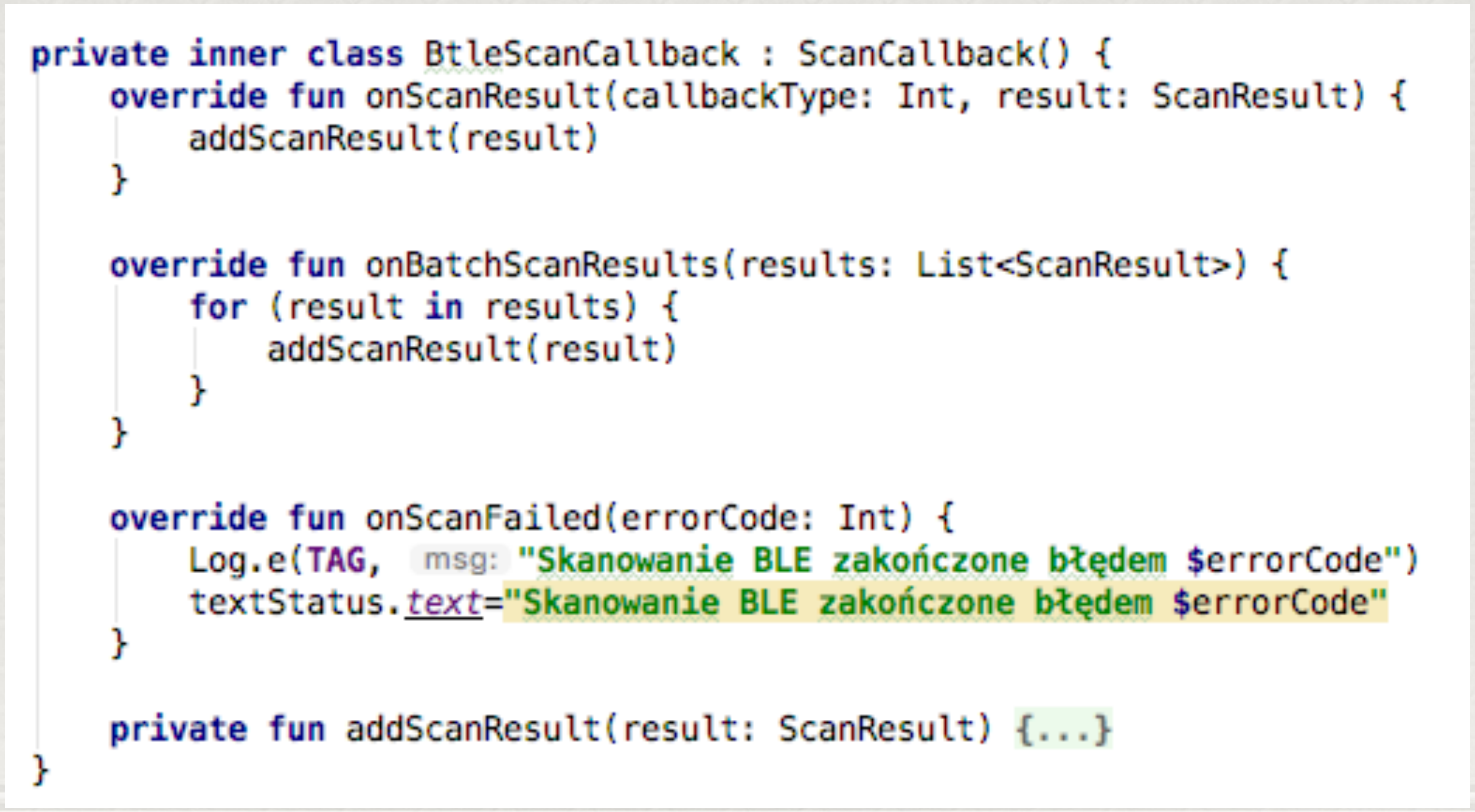
- *Skanując musimy zdefiniować filtry (na razie puste) i tryb pracy (Low Power)*



```
val filters = ArrayList<ScanFilter>()
val settings = ScanSettings.Builder().setScanMode(ScanSettings.SCAN_MODE_LOW_POWER).build()
```

Klient

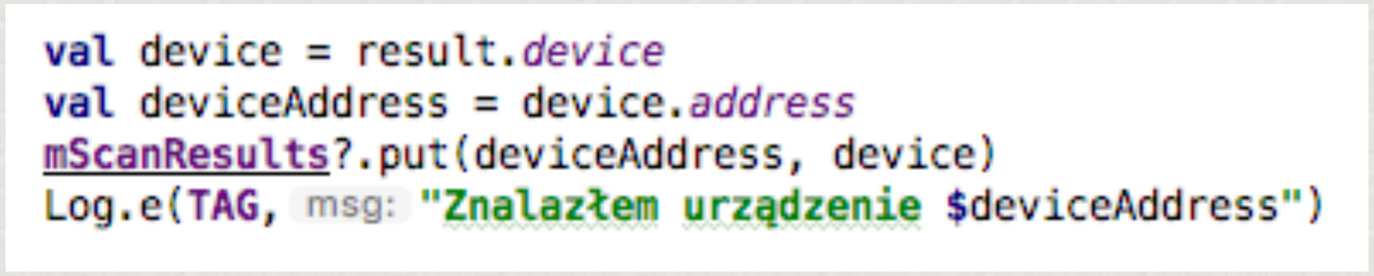
- *Ponieważ proces skanowania jest asynchroniczny musimy zdefiniować klasę, która obsłuży CallBack wewnątrz naszej klasy aktywności*



```
private inner class BtleScanCallback : ScanCallback() {
    override fun onScanResult(callbackType: Int, result: ScanResult) {
        addScanResult(result)
    }

    override fun onBatchScanResults(results: List<ScanResult>) {
        for (result in results) {
            addScanResult(result)
        }
    }

    override fun onScanFailed(errorCode: Int) {
        Log.e(TAG,  msg: "Skanowanie BLE zakończone błędem $errorCode")
        textStatus.text="Skanowanie BLE zakończone błędem $errorCode"
    }

    private fun addScanResult(result: ScanResult) {...}
}
```

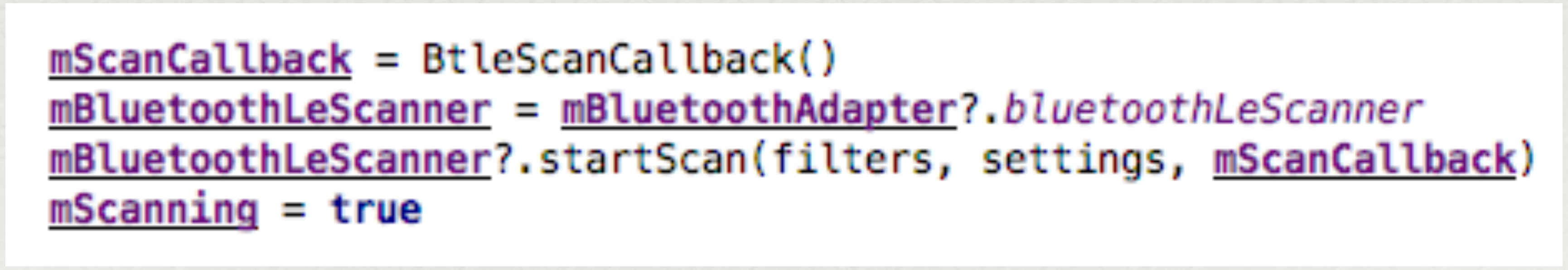
Klient

- *W metodzie addScanResult dodamy znalezione urządzenia do listy*



```
val device = result.device
val deviceAddress = device.address
mScanResults?.put(deviceAddress, device)
Log.e(TAG, msg: "Znalazłem urządzenie $deviceAddress")
```

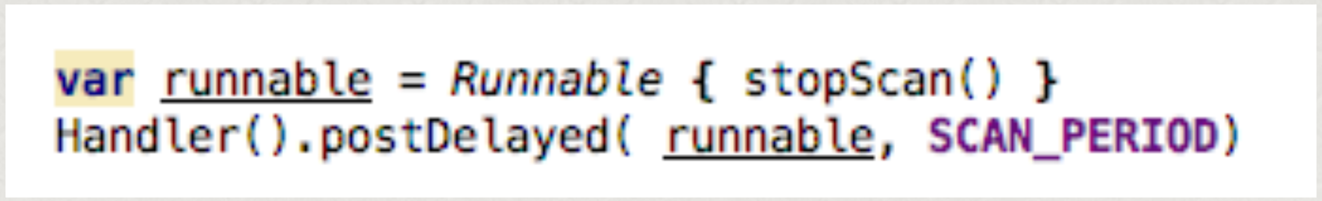
- *Na koniec metody startScan dodajemy*



```
mScanCallback = BtleScanCallback()
mBluetoothLeScanner = mBluetoothAdapter?.bluetoothLeScanner
mBluetoothLeScanner?.startScan(filters, settings, mScanCallback)
mScanning = true
```

Klient

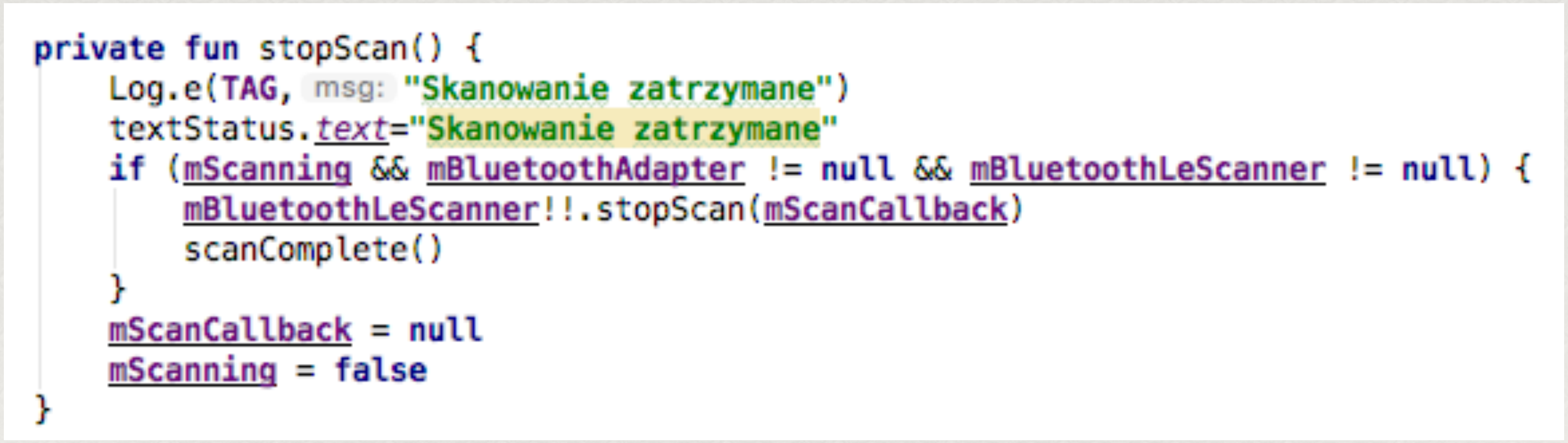
- *Gdybyśmy teraz uruchomili aplikację, skanowanie działałoby bez końca*
- *Dlatego dodamy zatrzymanie procesu po określonym czasie na koniec metody startScan*



```
var runnable = Runnable { stopScan() }
Handler().postDelayed( runnable, SCAN_PERIOD)
```

Klient

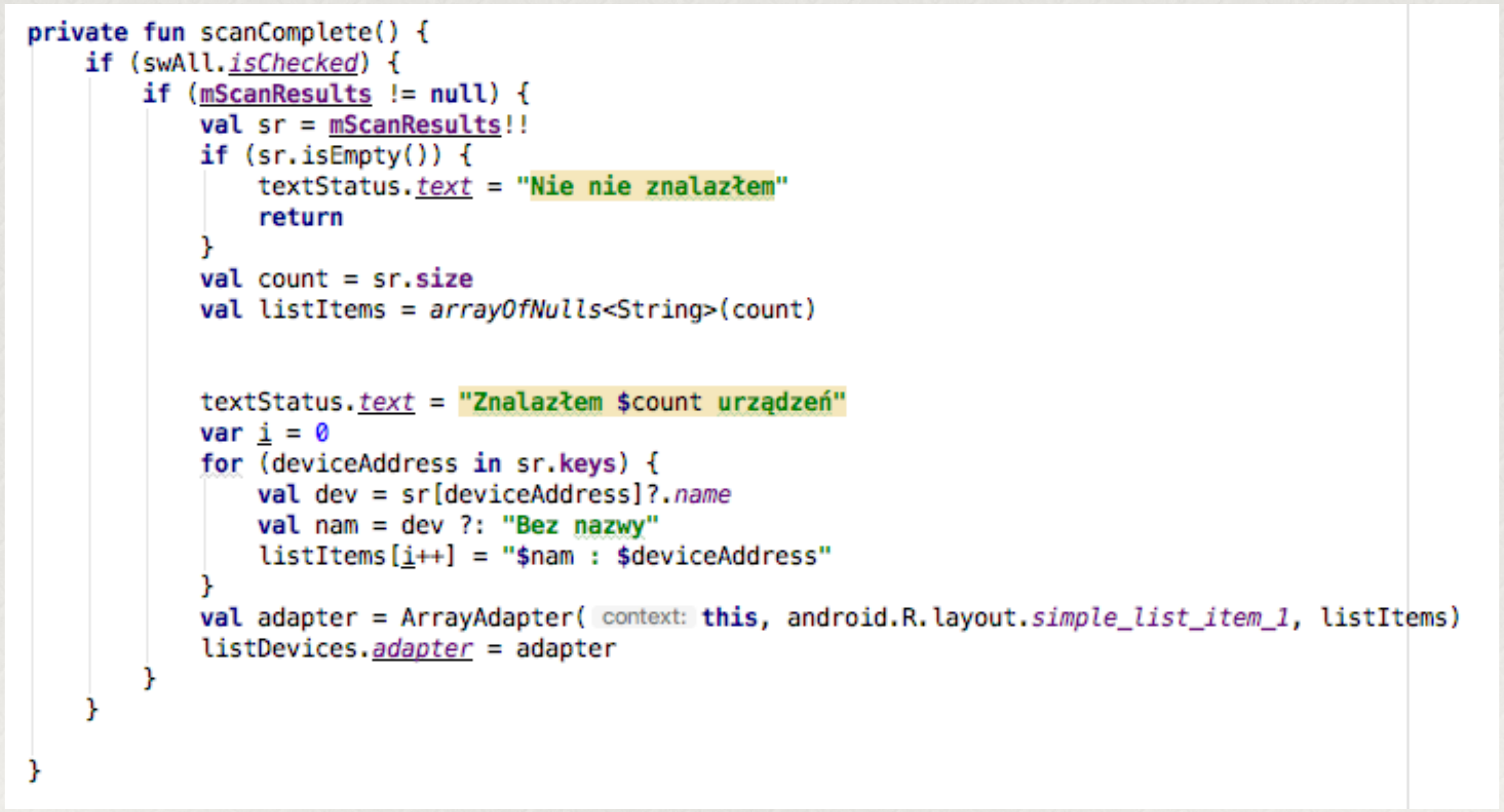
- *Musimy jeszcze dodać metodę stopScan*



```
private fun stopScan() {
    Log.e(TAG, msg: "Skanowanie zatrzymane")
    textStatus.text="Skanowanie zatrzymane"
    if (mScanning && mBluetoothAdapter != null && mBluetoothLeScanner != null) {
        mBluetoothLeScanner!!.stopScan(mScanCallback)
        scanComplete()
    }
    mScanCallback = null
    mScanning = false
}
```

Klient

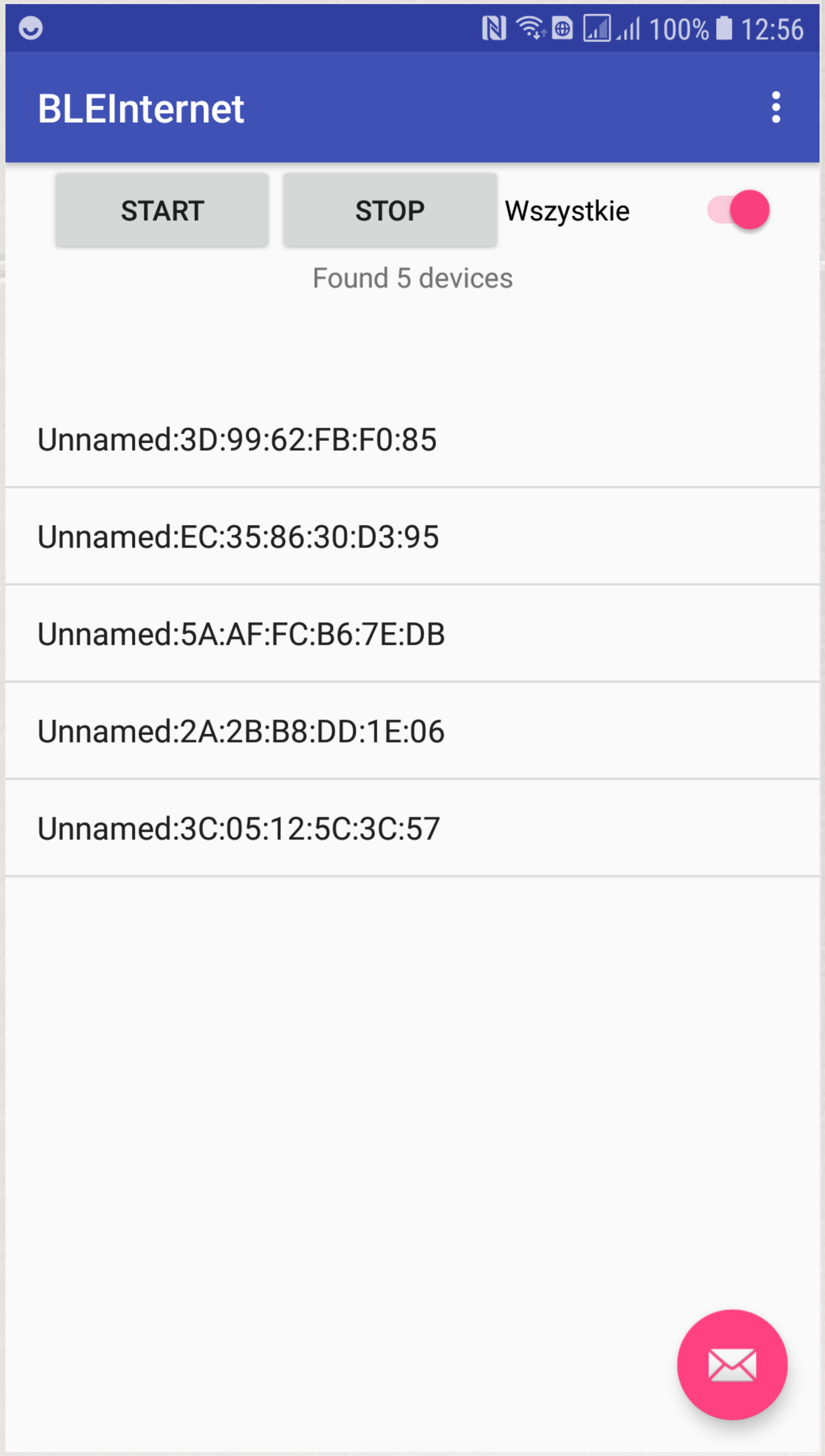
- *I metodę scanComplete*



```kotlin
private fun scanComplete() {
    if (swAll.isChecked) {
        if (mScanResults != null) {
            val sr = mScanResults!!
            if (sr.isEmpty()) {
                textStatus.text = "Nie nie znalazłem"
                return
            }
            val count = sr.size
            val listItems = arrayOfNulls<String>(count)


            textStatus.text = "Znalazłem $count urządzeń"
            var i = 0
            for (deviceAddress in sr.keys) {
                val dev = sr[deviceAddress]?.name
                val nam = dev ?: "Bez nazwy"
                listItems[i++] = "$nam : $deviceAddress"
            }
            val adapter = ArrayAdapter( context: this, android.R.layout.simple_list_item_1, listItems)
            listDevices.adapter = adapter
        }
    }

}
```

Klient

- *Możemy teraz uruchomić aplikację*
- *Po wciśnięciu przycisku Start i upływie 5s powinna wyświetlić się lista znalezionych urządzeń BLE w otoczeniu (checkbox wszystkie musi być zaznaczony)*



Found 5 devices

```
Unnamed:3D:99:62:FB:F0:85
Unnamed:EC:35:86:30:D3:95
Unnamed:5A:AF:FC:B6:7E:DB
Unnamed:2A:2B:B8:DD:1E:06
Unnamed:3C:05:12:5C:3C:57
```