Introduction to Android
Hello World! Application

Çağatay Sönmez

07.12.2018

Arcelik Electronics Plant Division, R&D Team - Software Design

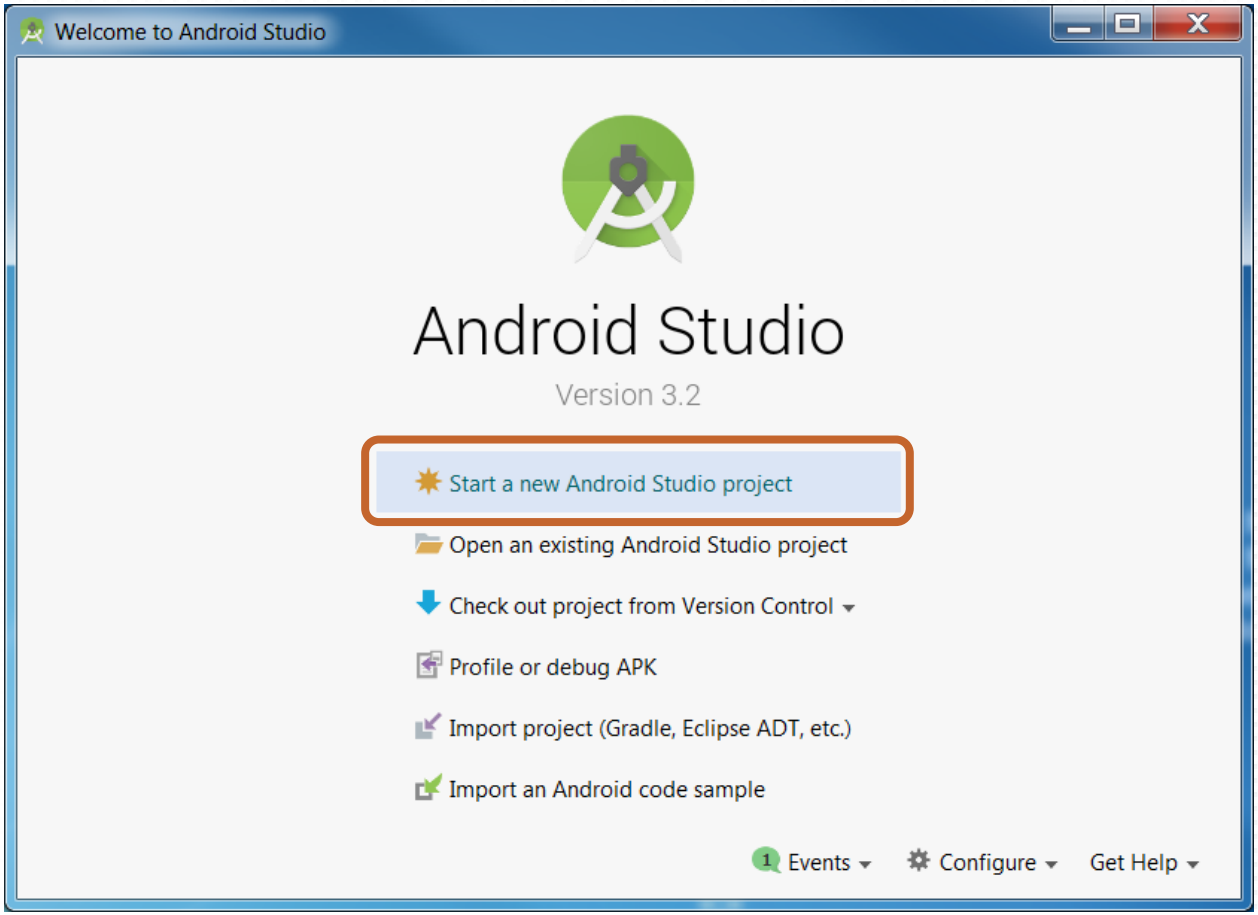
Agenda

- Creating a sample Android TV application
- Adding Activities
- Adding Services
- Adding BroadcastReceivers
- Running the app on the emulator
- Running the app on the device over USB
- Running the app on the device over network
- Profile the app performance
- Debugging the app

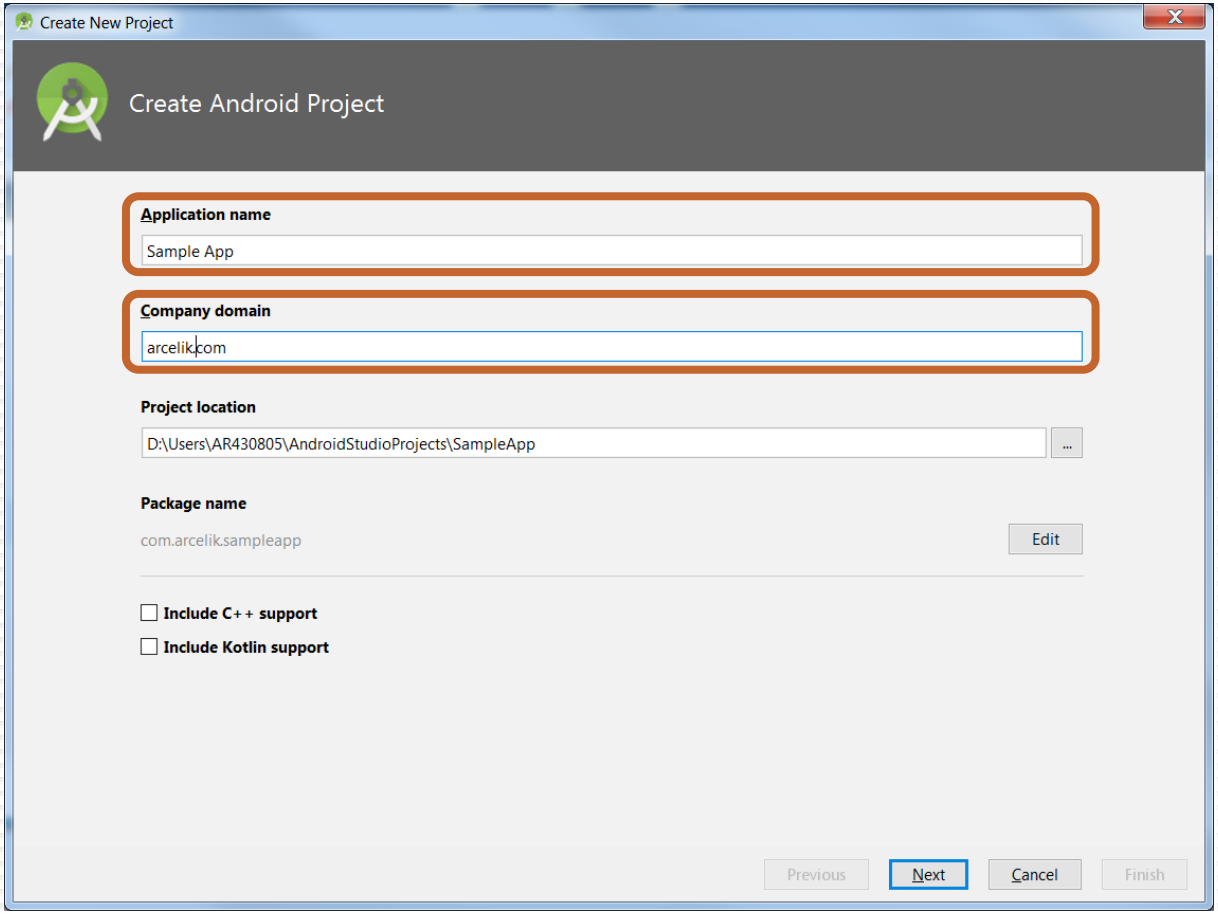
Meterials

- Application source code can be found on GitHub
  - https://github.com/CagataySonmez/Android-for-Beginners/tree/master/3-IntroductionToAndroid-HelloWorldApplication
- Android Studio version 3.2.1 is used on this training
- Android Studio can be downloaded from the official website
  - https://developer.android.com/studio/
- Free courses can be found on Google Developers Training web page
  - https://developers.google.com/training/android/

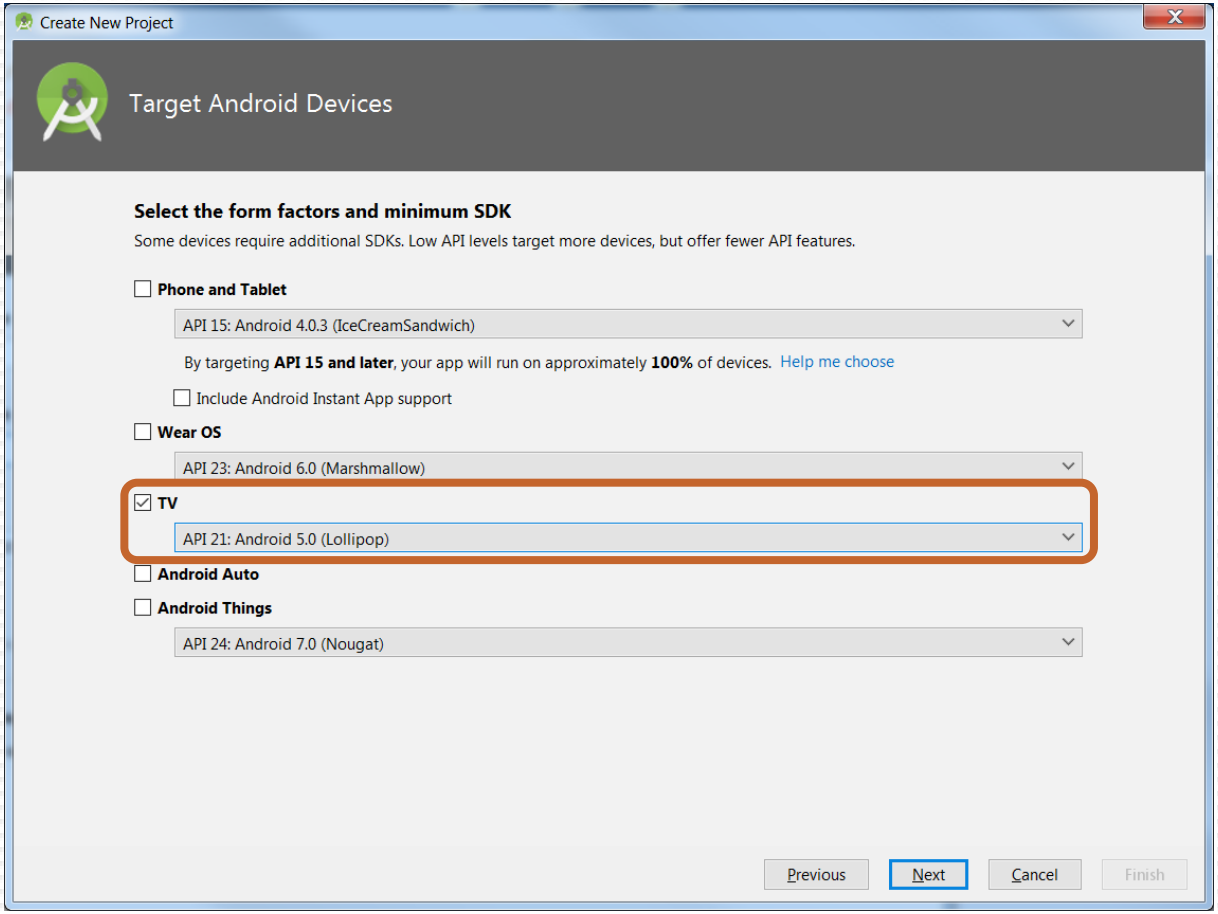
Creating Android TV App I



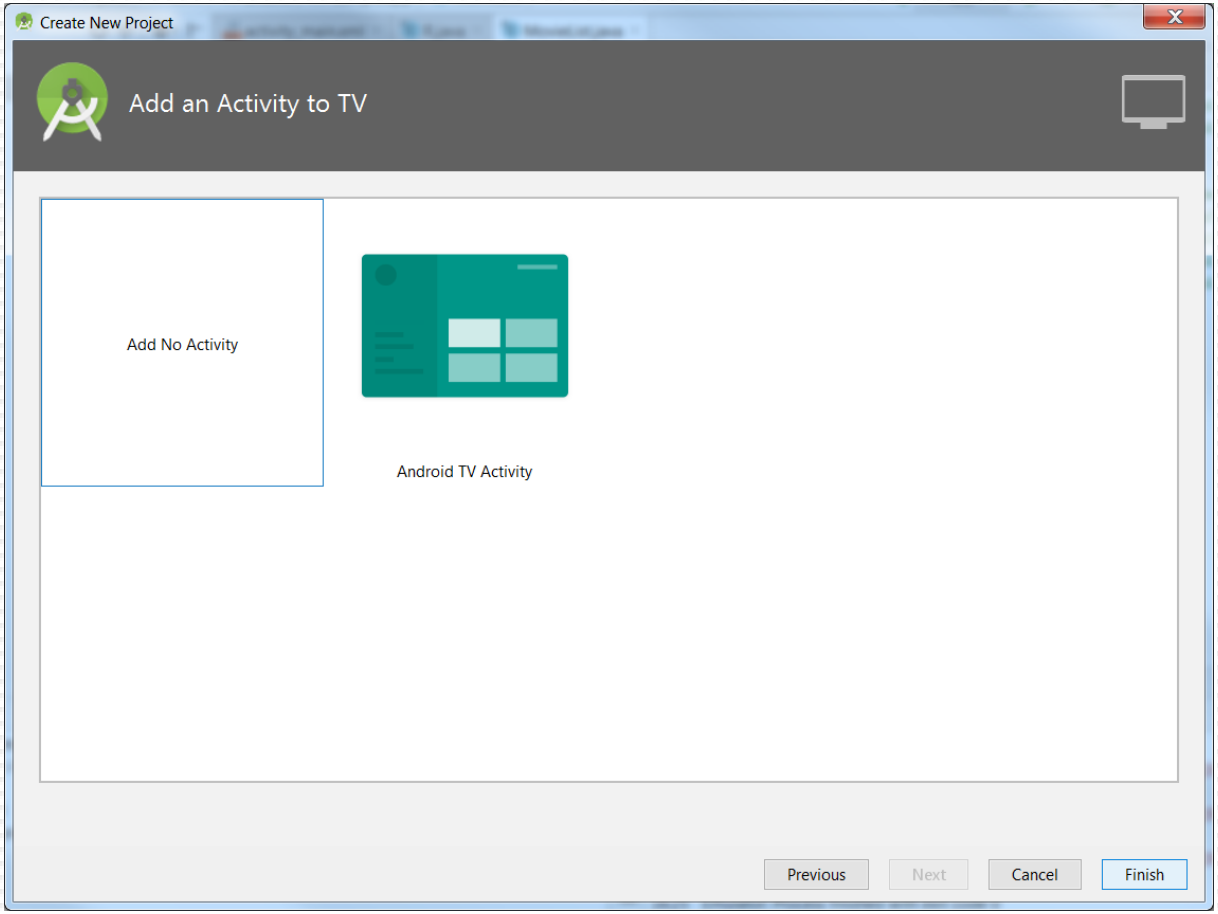
Creating Android TV App II



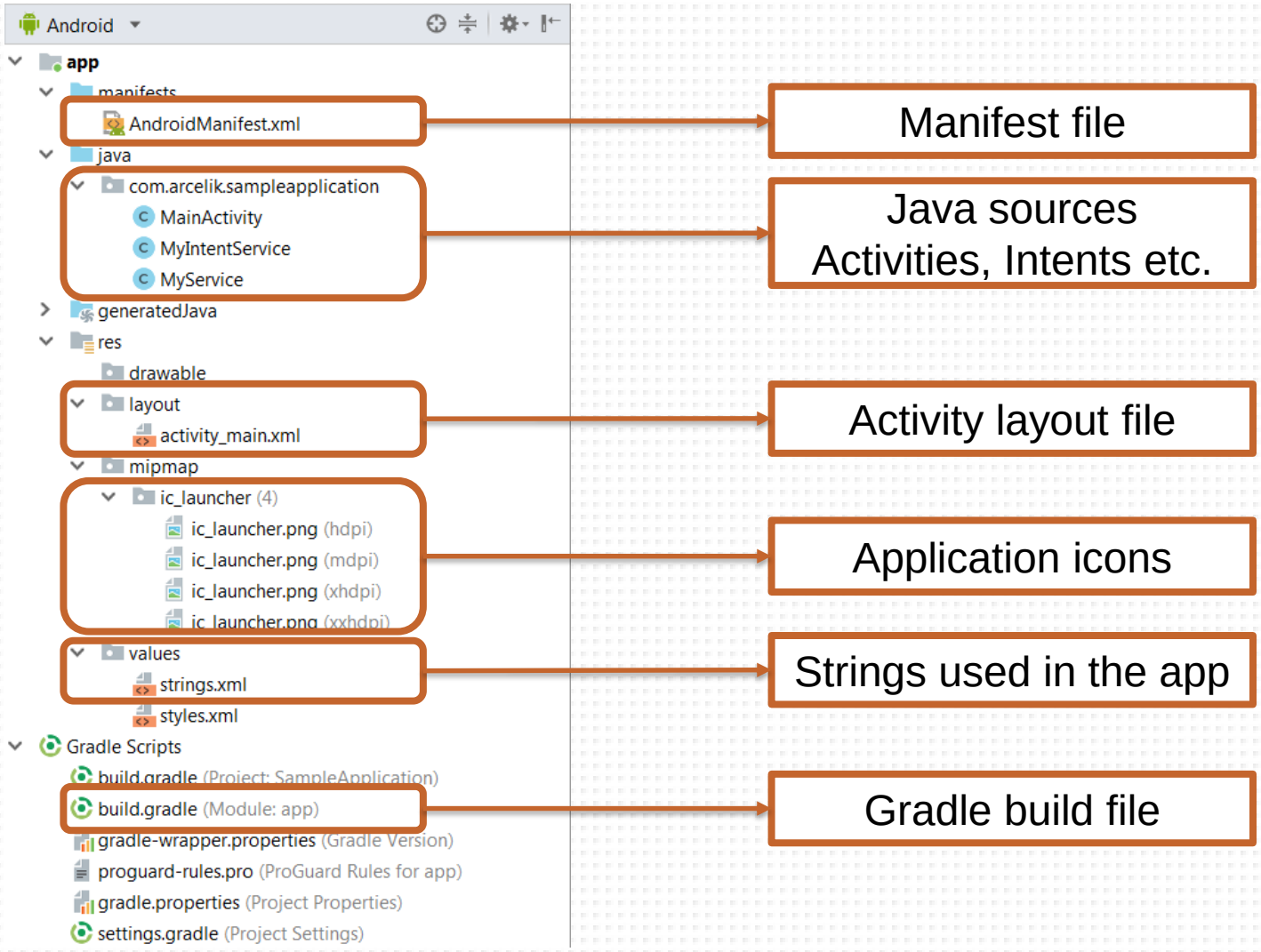
Creating Android TV App III



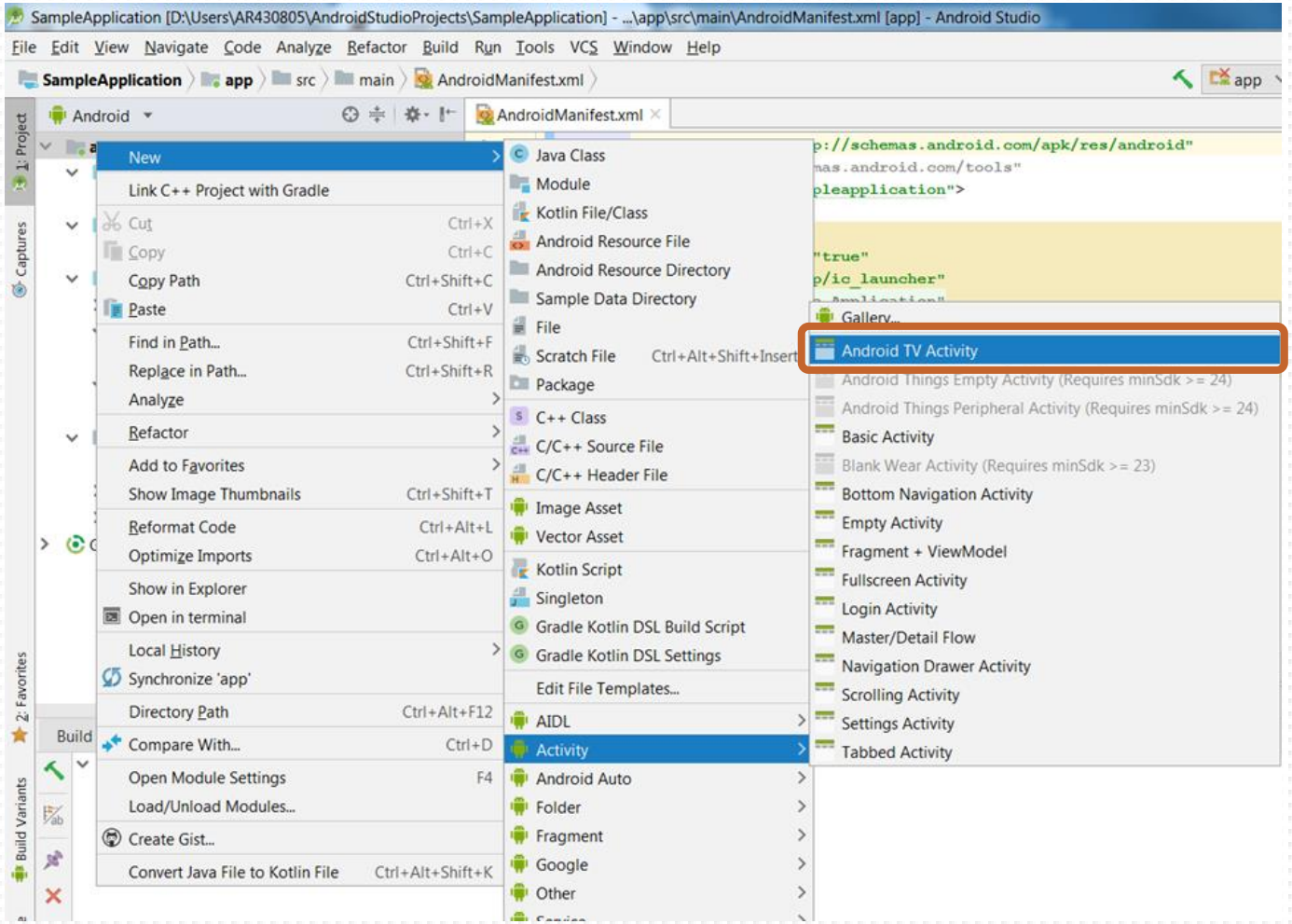
Creating Android TV App IV



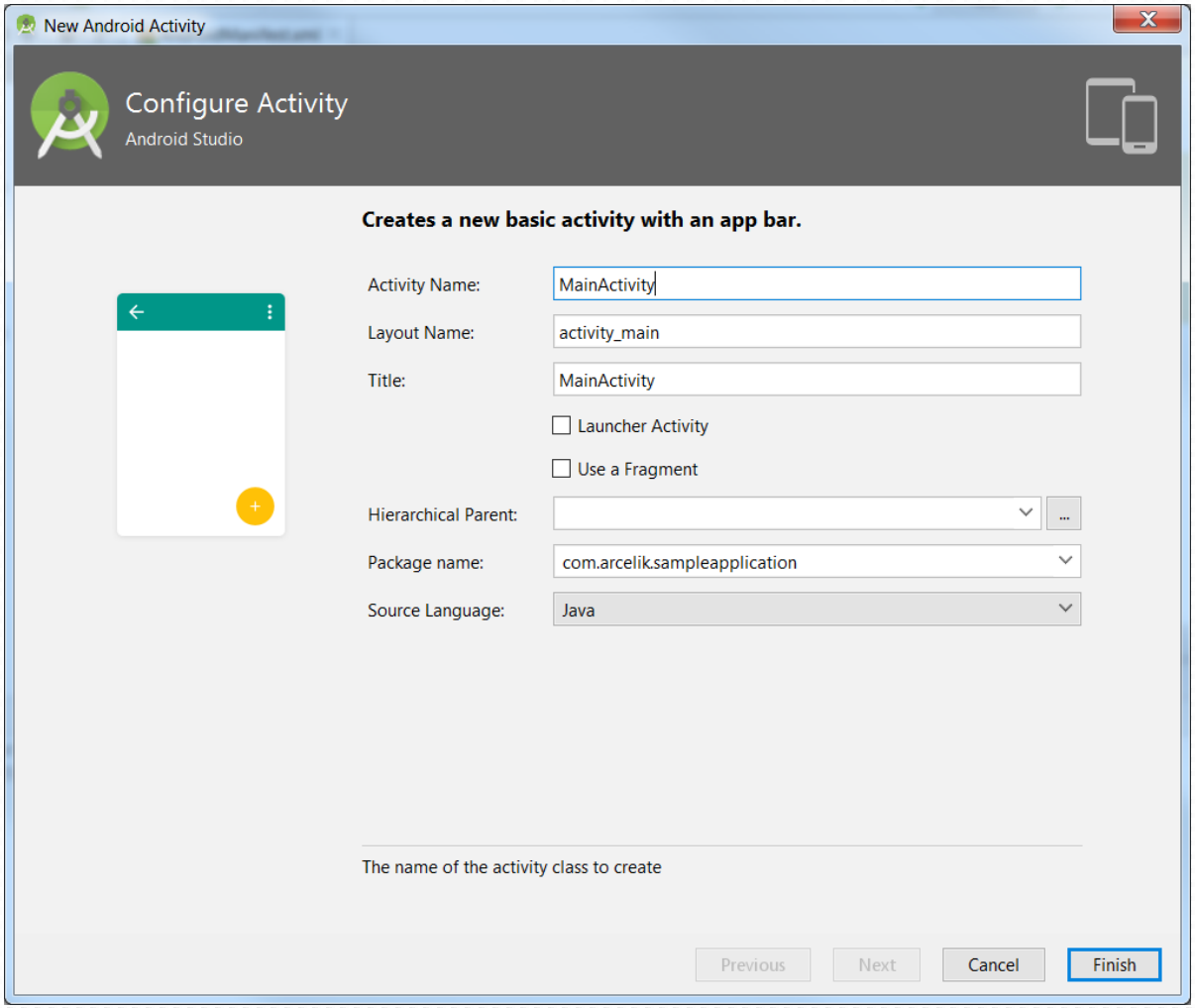
Creating Android TV App V



Adding Activity I

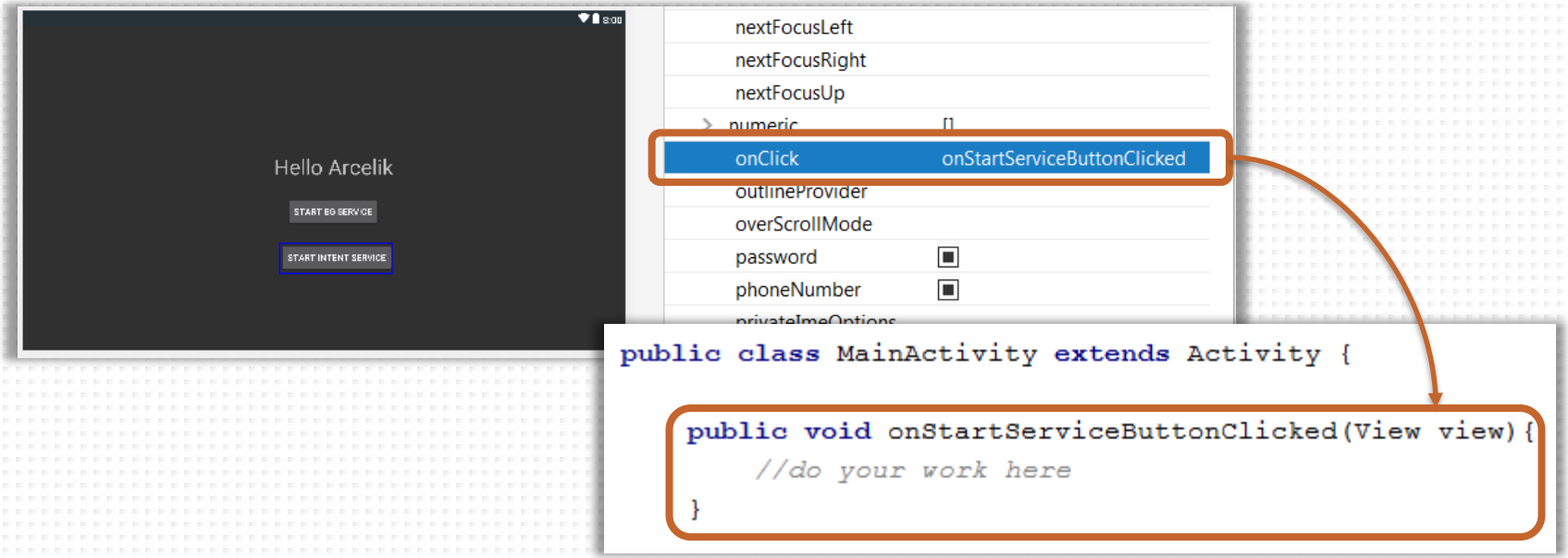


Adding Activity II



Handling Click Events of Button I

- There are many options to handle click events of button
- Option 1: Using onClick property in xml layout file
- The callback function's signature cannot be changed!



Handling Click Events of Button II

- Option 2: Using an View.OnClickListener via an **anonymous inner class**
- The system executes the code in onClick on the main thread!

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // set onclick listener of start_service button
    Button startService = (Button)findViewById(R.id.start_service);
    startService.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            //Do your work here
        }
    });
```

Handling Click Events of Button III

- Option 3: Using an View.OnClickListener via an anonymous inner class which can be reusable

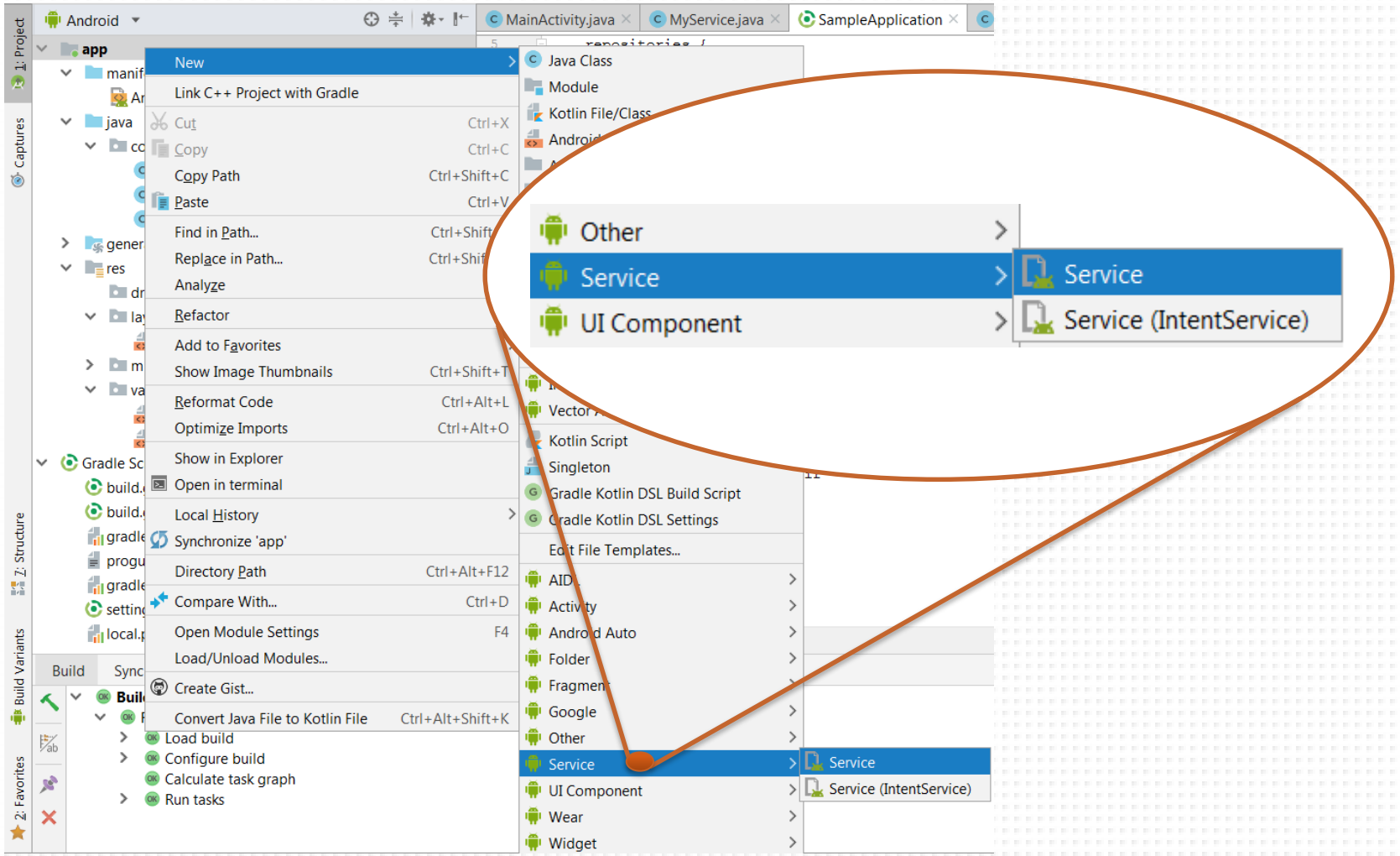
```
private View.OnClickListener startServiceOnClickListener = new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //Do your work here
    }
};

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // set onclick listener of start_service button
    Button startService = (Button)findViewById(R.id.start_service);
    startService.setOnClickListener(startServiceOnClickListener);
```

Handling Click Events of Button IV

- Option 4: Using your own class by implementing View.OnClickListener Interface

```
class StartServiceButtonClick implements View.OnClickListener {
    @Override
    public void onClick(View v) {
        //Do your work here
    }
}

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // set onclick listener of start_service button
    Button startService = (Button)findViewById(R.id.start_service);
    startService.setOnClickListener(new StartServiceButtonClick());
```

Adding Service

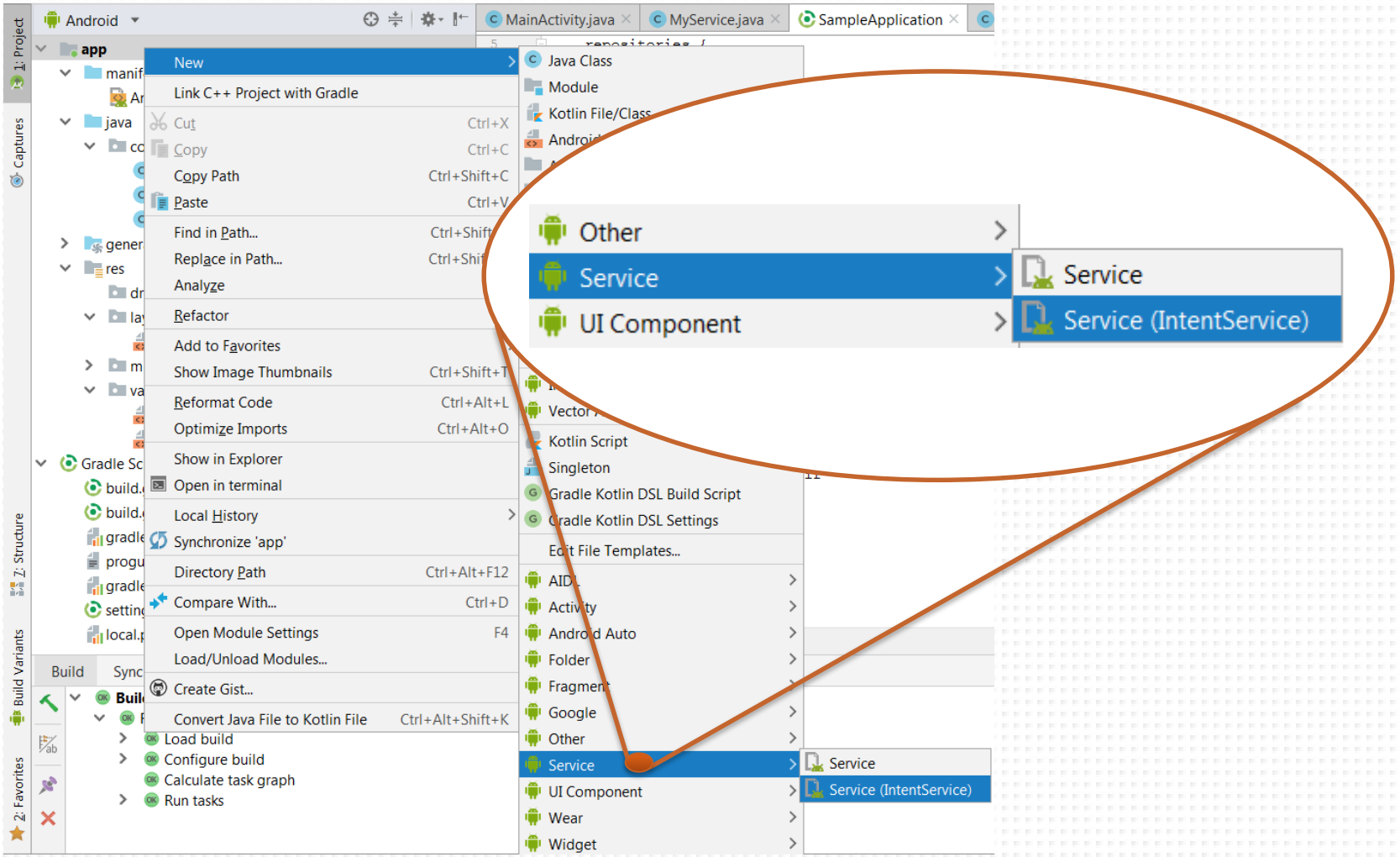


Toogle the Service

1. Add a button click listener
2. Create an explicit Intent when the button is clicked
3. Start or stop the Service via created Intent

```
// Click this button to start service.
Button startService = (Button)findViewById(R.id.start_service);
startService.setOnClickListener((v) → {
        Intent intent = new Intent( packageContext: MainActivity.this, MyService.class);
        if(isMyServiceRunning(MyService.class)){
            stopService(intent);
            ((Button)findViewById(R.id.start_service)).setText("Start BG Service");
        }
        else {
            startService(intent);
            ((Button)findViewById(R.id.start_service)).setText("Stop BG Service");
        }
});
```

Adding Intent Service

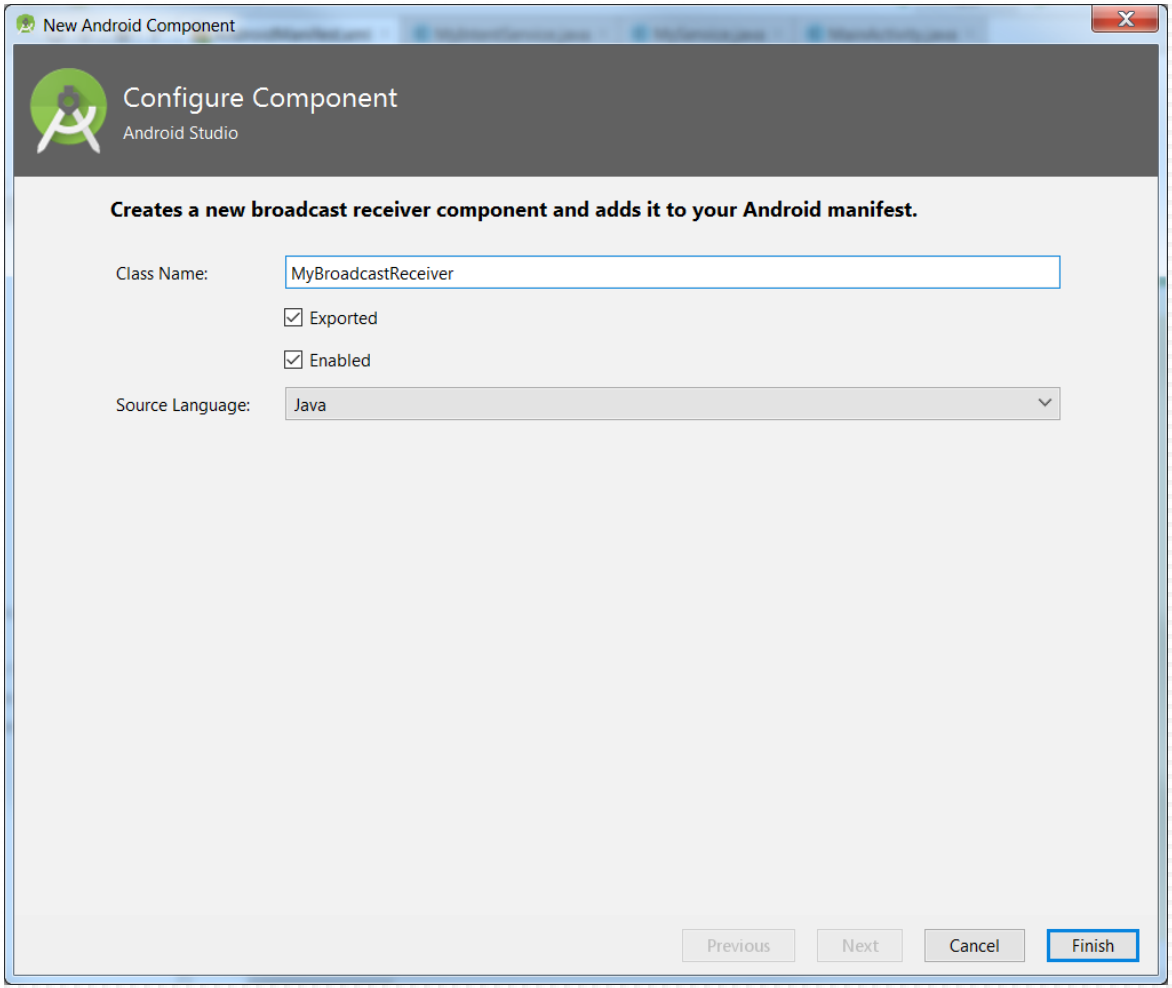


Starting Intent Service

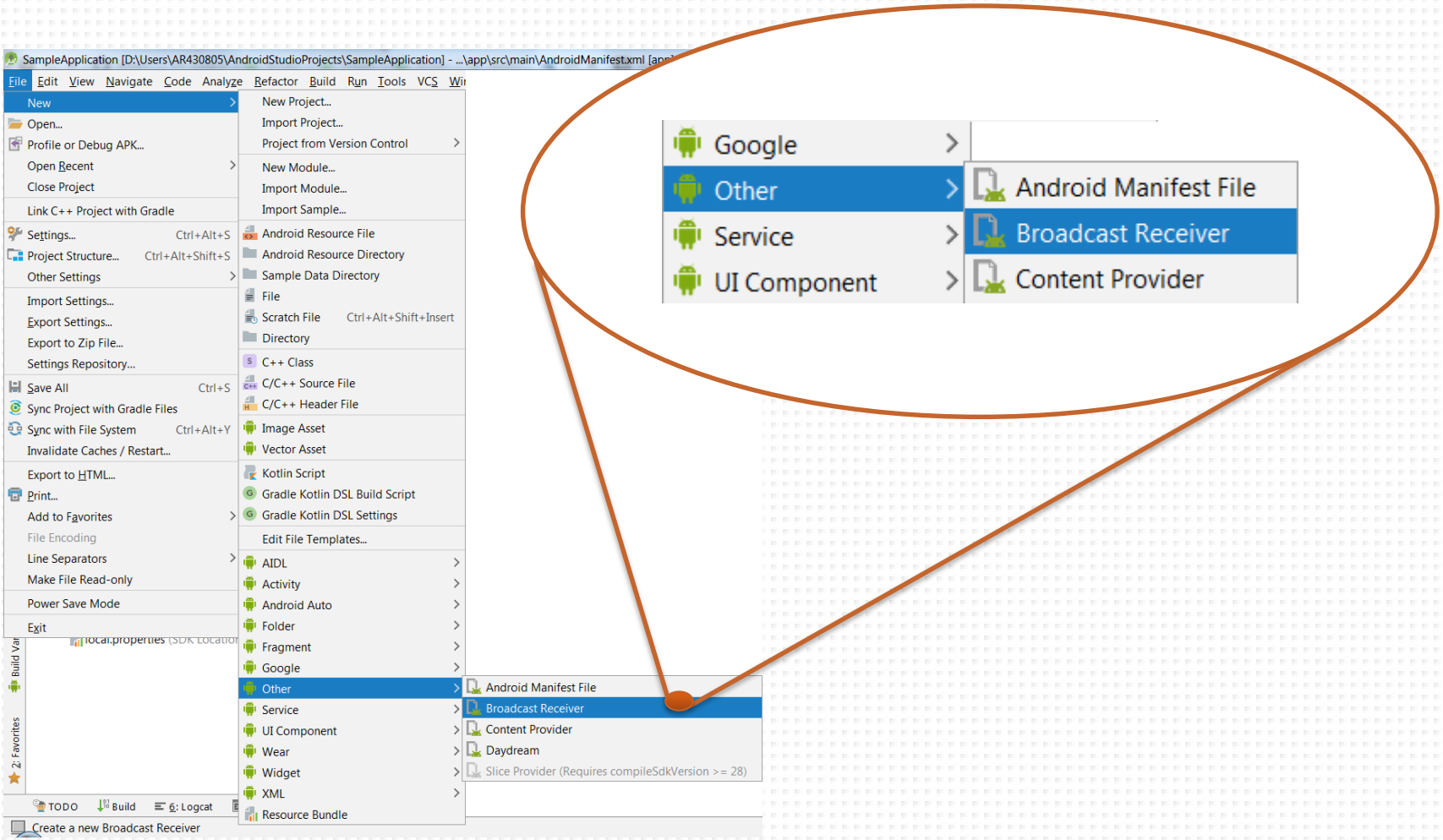
1. Add a button click listener
2. Create an explicit Intent when the button is clicked
3. Set Intent action and extra data
4. Start the Intent Service via created Intent

```
// Click this button to start intent service.
Button startIntentService = (Button)findViewById(R.id.start_intent_service);
startIntentService.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent( packageContext: MainActivity.this, MyIntentService.class);
        intent.setAction(MyIntentService.ACTION_SEND_NOTIFICATION);
        intent.putExtra(MyIntentService.NOTIFICATION_METHOD, value: "TOAST");
        startService(intent);
    }
});
```

Adding Broadcast Receiver I



Adding Broadcast Receiver II



Sending Broadcast

1. Declare related permission in manifest file

```
<permission
    android:name="com.arcelik.sampleapplication.permission.NOTIFICATION"
    android:protectionLevel="dangerous">
</permission>

<uses-permission android:name="com.arcelik.sampleapplication.permission.NOTIFICATION" />
```

2. Send broadcast message in your application

```
//Send broadcast
Log.d(TAG_SERVICE, msg: "Sending message to activity: " + notification);
final Intent intent = new Intent(BROADCAST_INTENT);
intent.putExtra( name: "msg", value: dateAsStr + ":" + notification);
sendBroadcast(intent, BROADCAST_PERMISSION);
```

Receiving Broadcast

1. Create an IntentFilter to catch proper broadcast event
2. Create BroadcastReceiver and overide onReceive method
3. Do your job with Intent provided by the receiver

```
public class MainActivity extends Activity {

    //Use broadcast receiver to get broadcast messages
    final IntentFilter myFilter = new IntentFilter(MyService.BROADCAST_INTENT);
    private MyBroadcastReceiver mReceiver = new MyBroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            final TextView responseFromService = (TextView)findViewById(R.id.msgArea);
            responseFromService.setText(intent.getCharSequenceExtra( name: "msg"));
        }
    };
```

Manifest File

AndroidManifest.xml | MyBroadcastReceiver.java | MyIntentService.java | MyService.java | MainActivity.java

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.arcelik.sampleapplication">
    <permission
        android:name="com.arcelik.sampleapplication.permission.NOTIFICATION"
        android:protectionLevel="dangerous">
    </permission>
    <uses-permission android:name="com.arcelik.sampleapplication.permission.NOTIFICATION" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <activity
            android:name=".MainActivity"
            android:label="@string/title_activity_main">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <service
            android:name=".MyIntentService"
            android:exported="false">
        </service>
        <service
            android:name=".MyService"
            android:exported="true">
        </service>
        <receiver
            android:name=".MyBroadcastReceiver"
            android:exported="true">
        </receiver>
    </application>
</manifest>
```

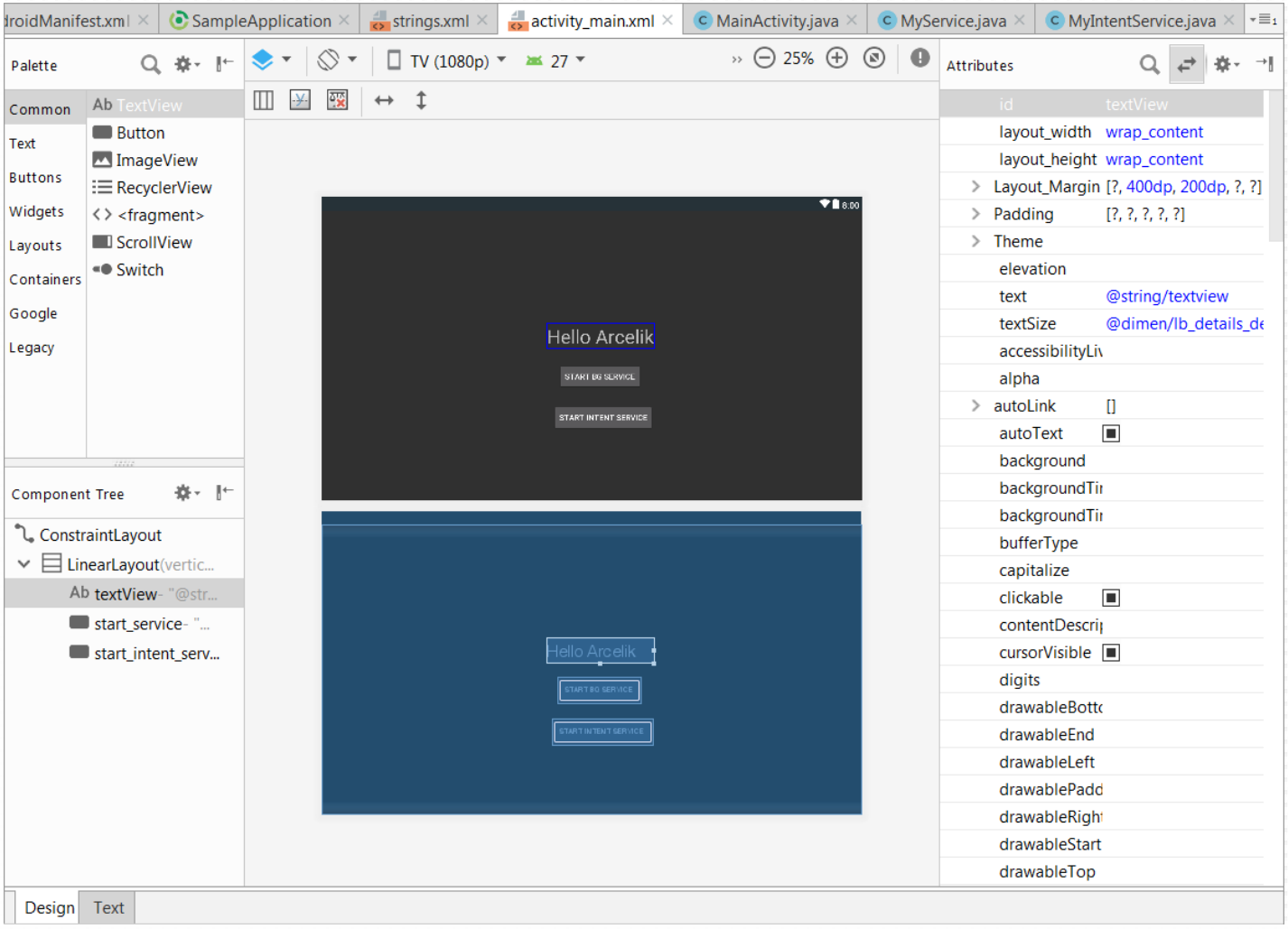
Permissions

Activities

Services

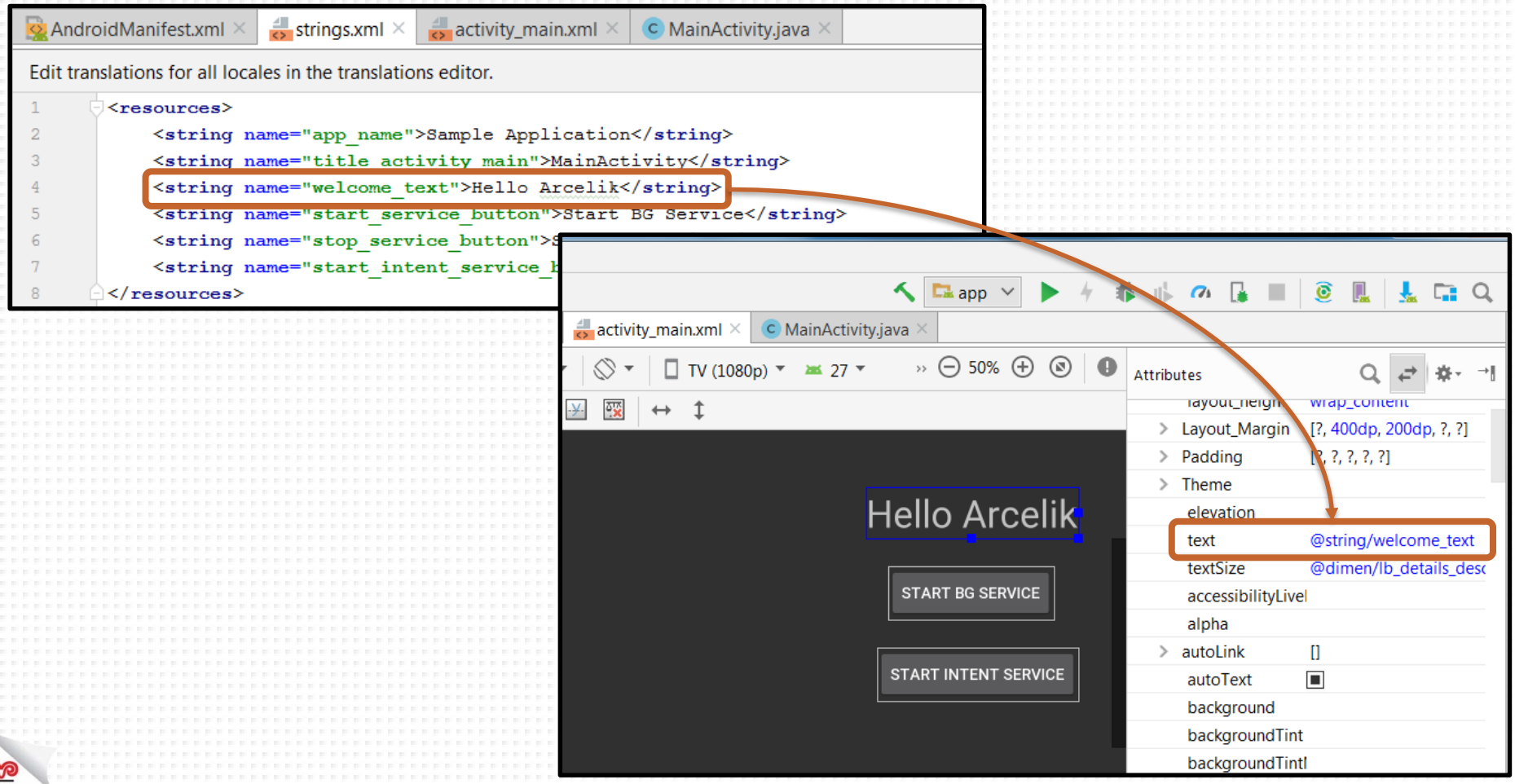
Receivers

Activity Layout File



Strings.txt file I

- Do not embed texts to layout file!



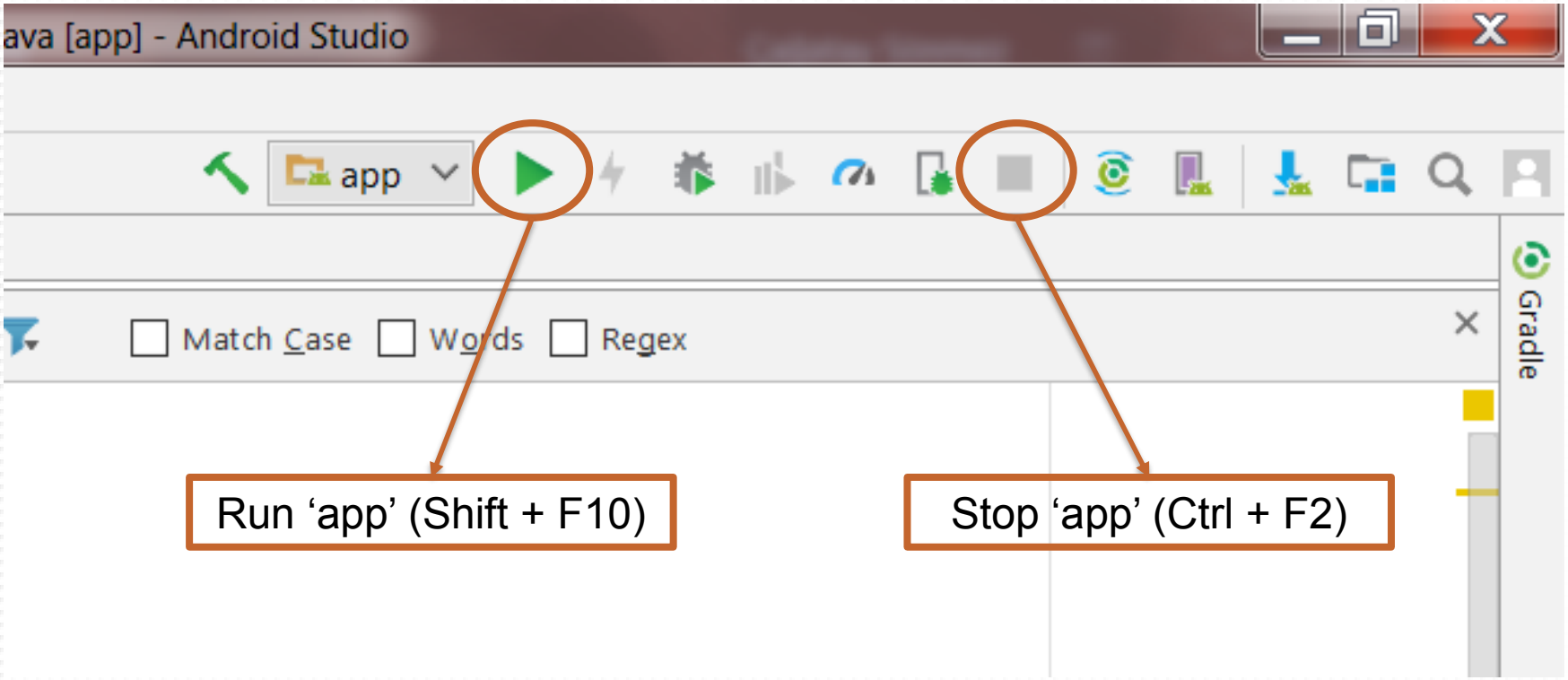
Strings.txt file II

- Do not use static texts in your source code!

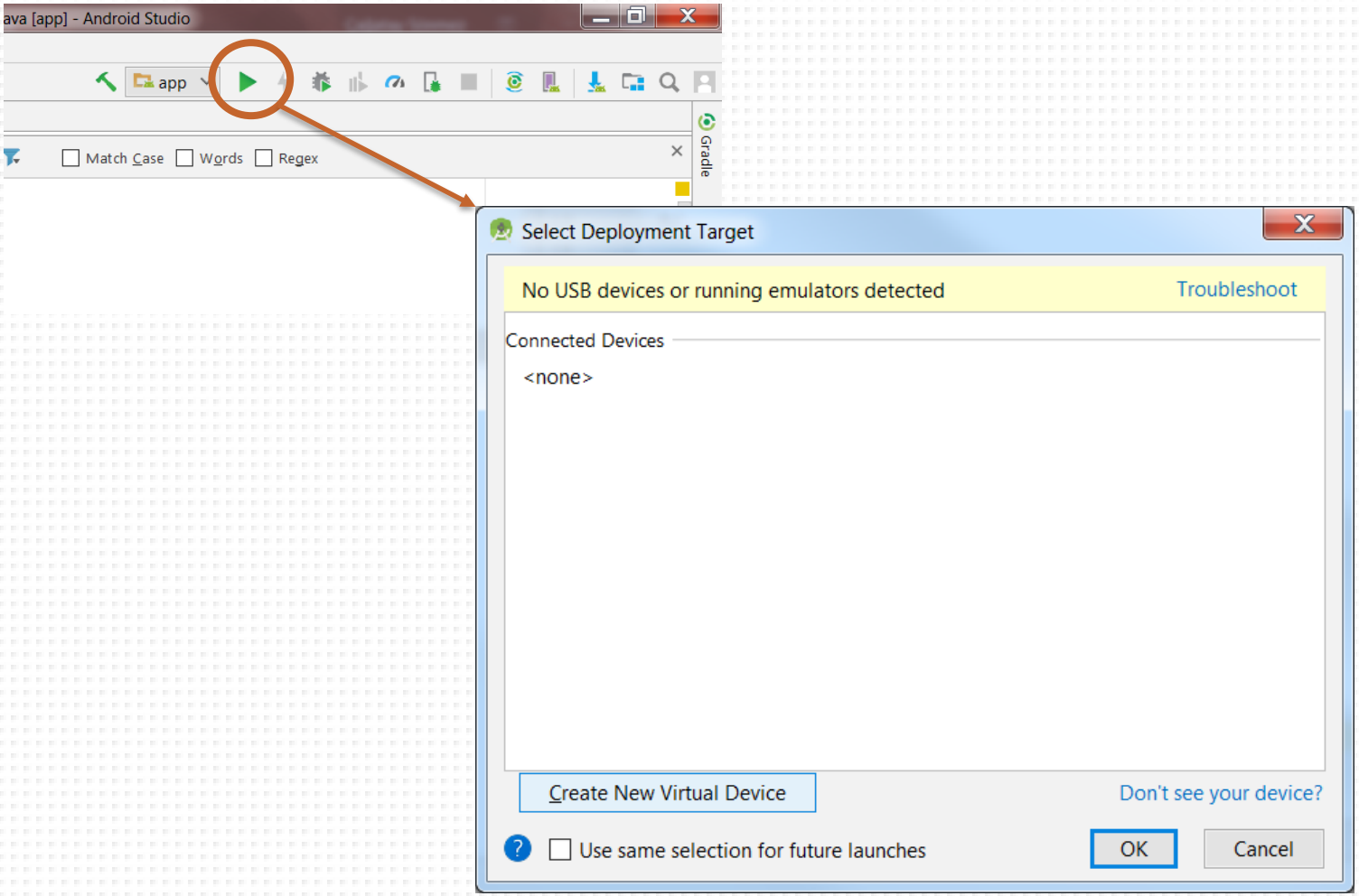
```
if(isMyServiceRunning(MyService.class))
    ((Button)findViewById(R.id.start_service)).setText("Stop BG Service");
```

```
if(isMyServiceRunning(MyService.class))
    ((Button)findViewById(R.id.start_service)).setText(R.string.stop_service_button);
```

Running an Android Application I

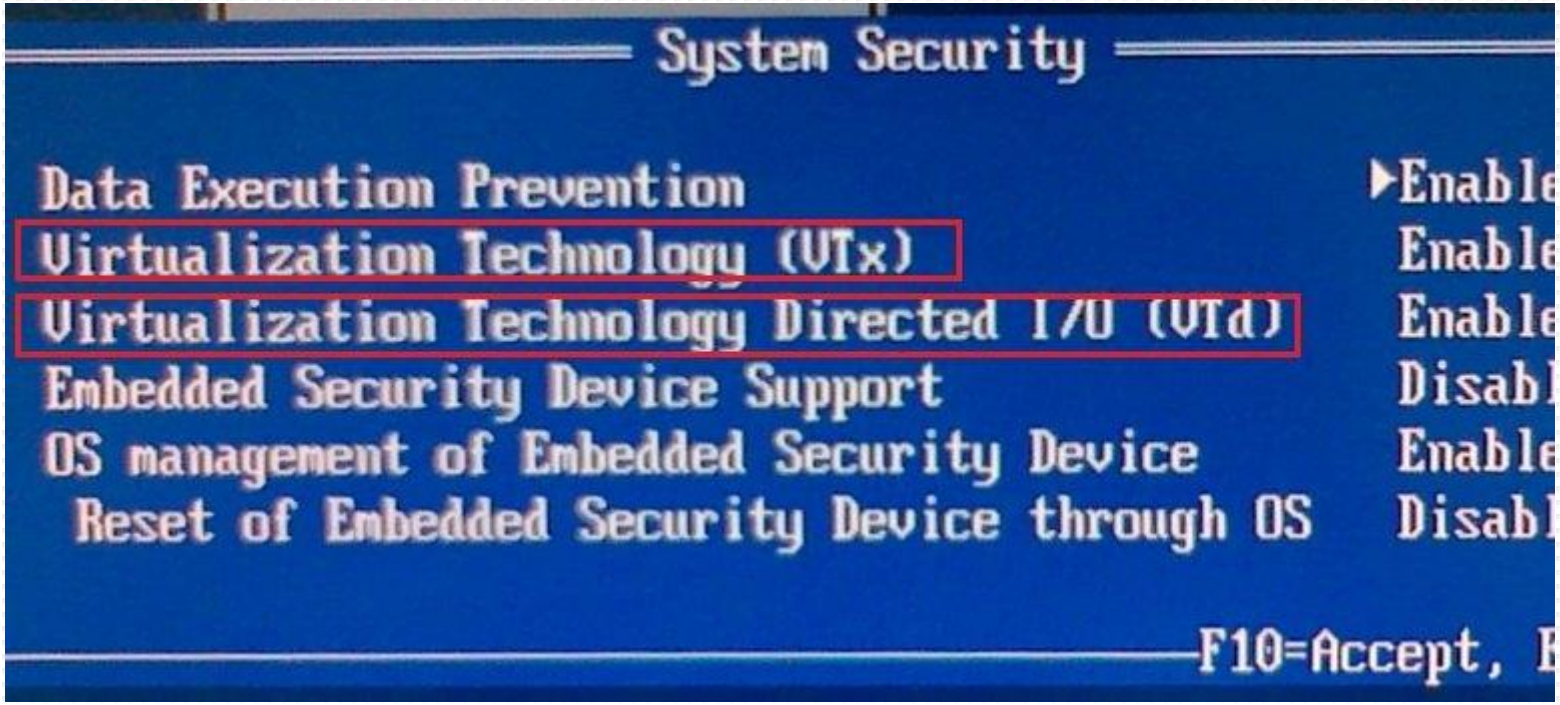


Running an Android Application II

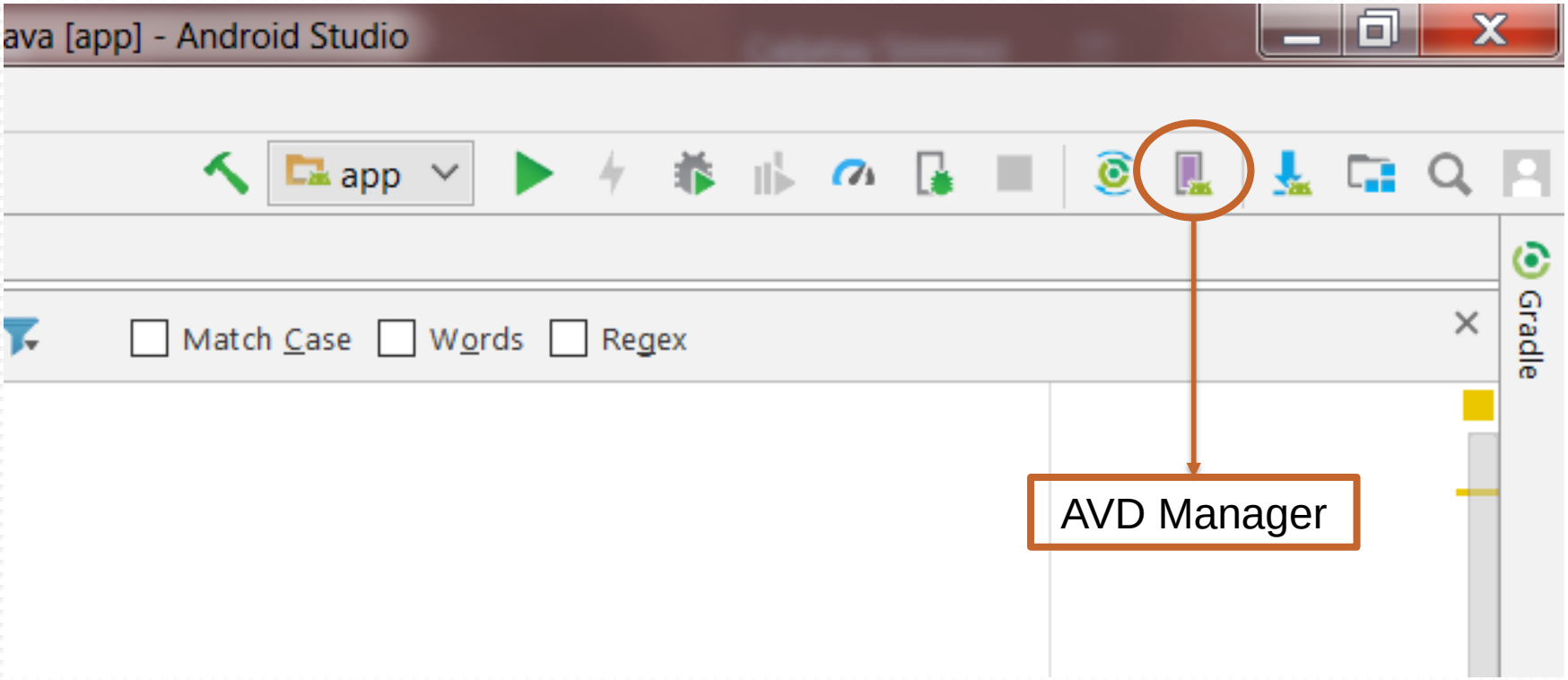


Running App on Emulator

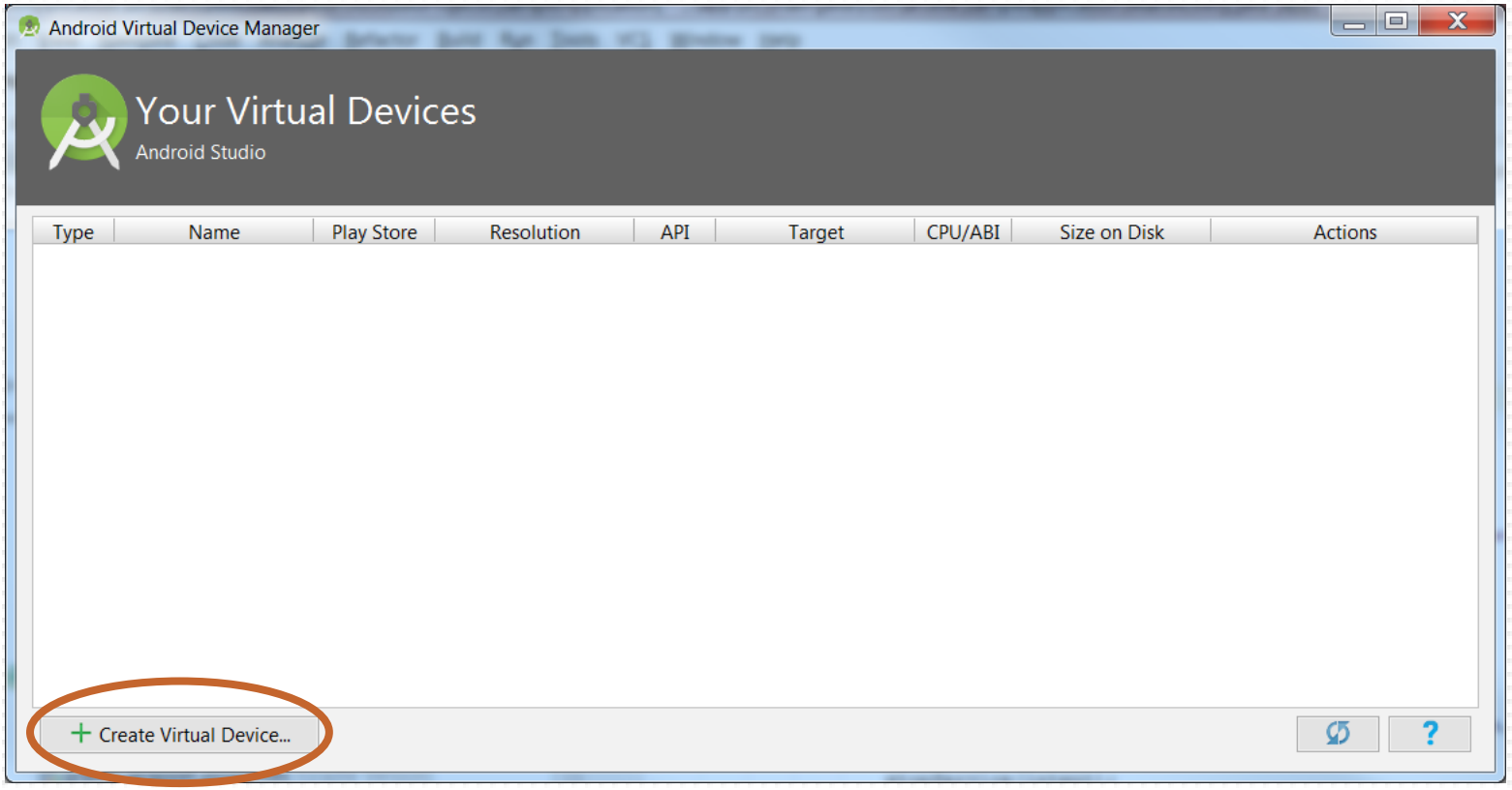
- To use emulator, enable Intel Virtualization Technology or AMD-V depending on the brand of the processor



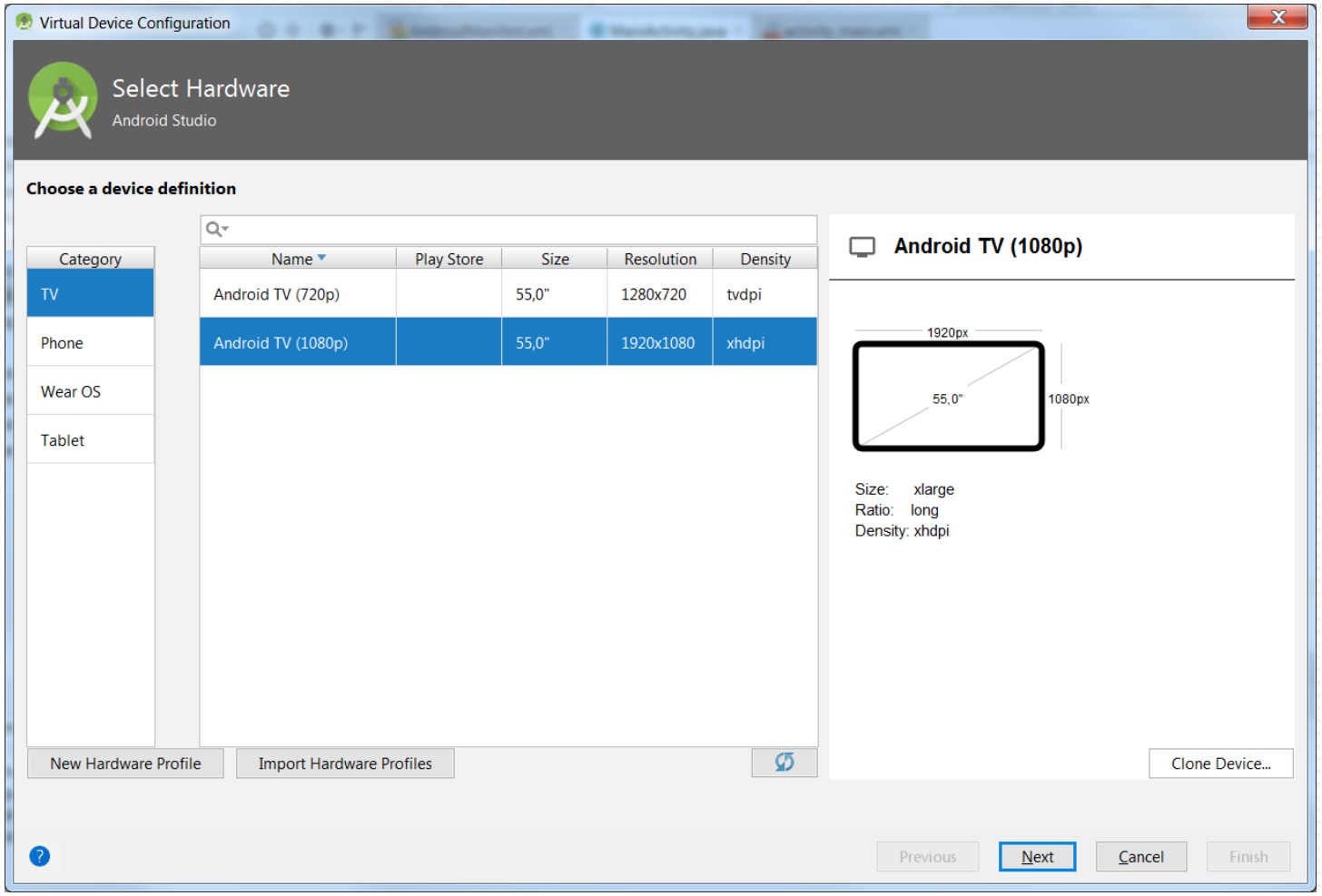
Creating Virtual Device I



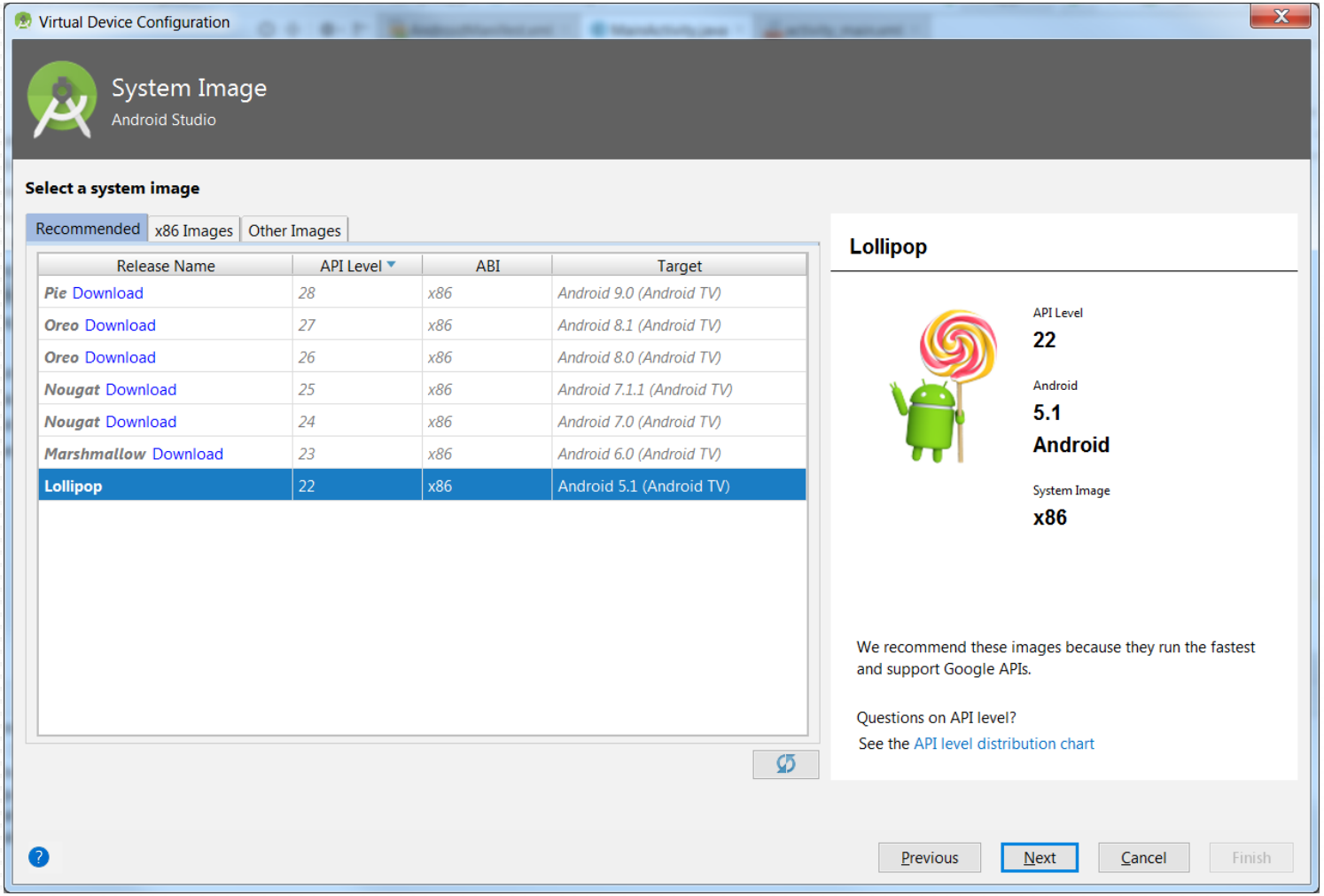
Creating Virtual Device II



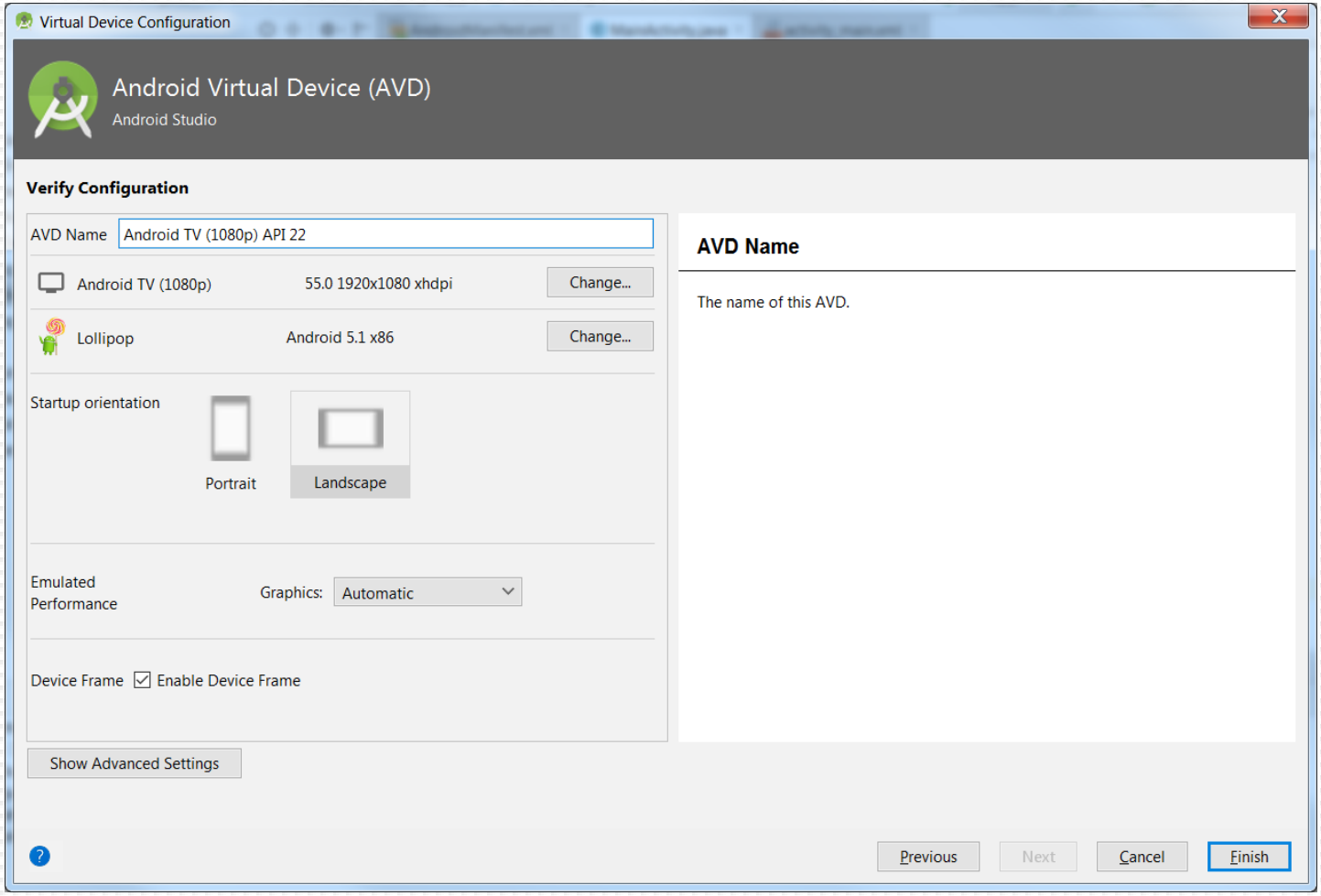
Creating Virtual Device III



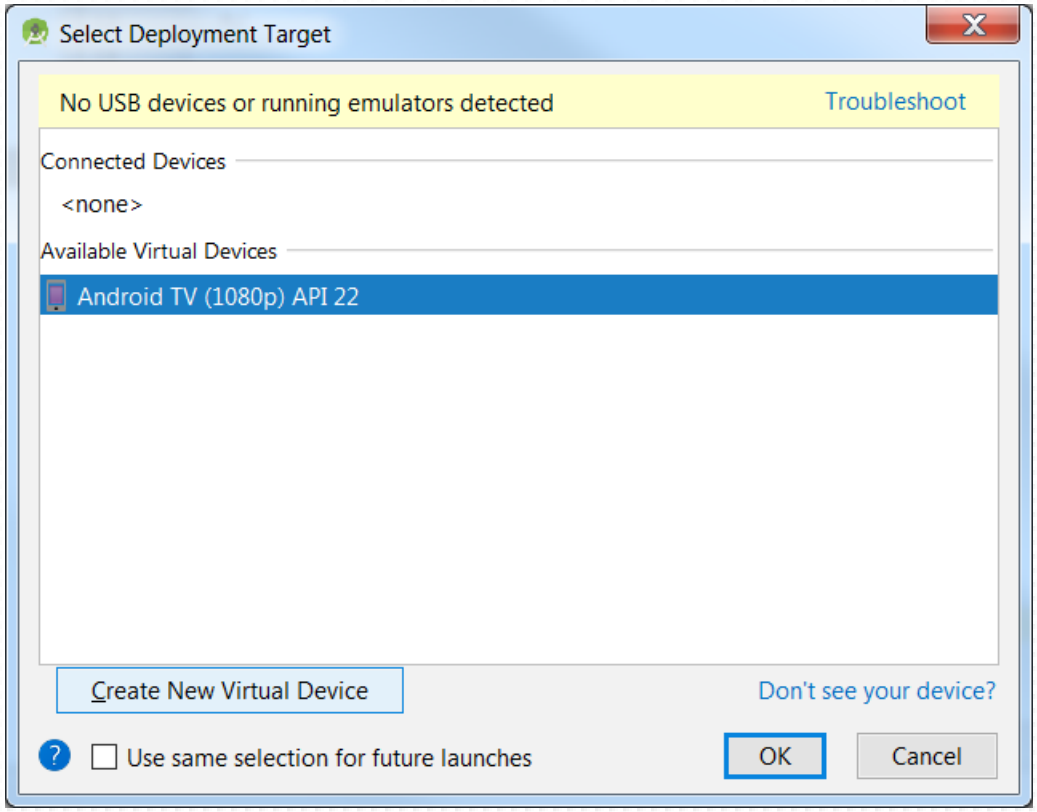
Creating Virtual Device IV



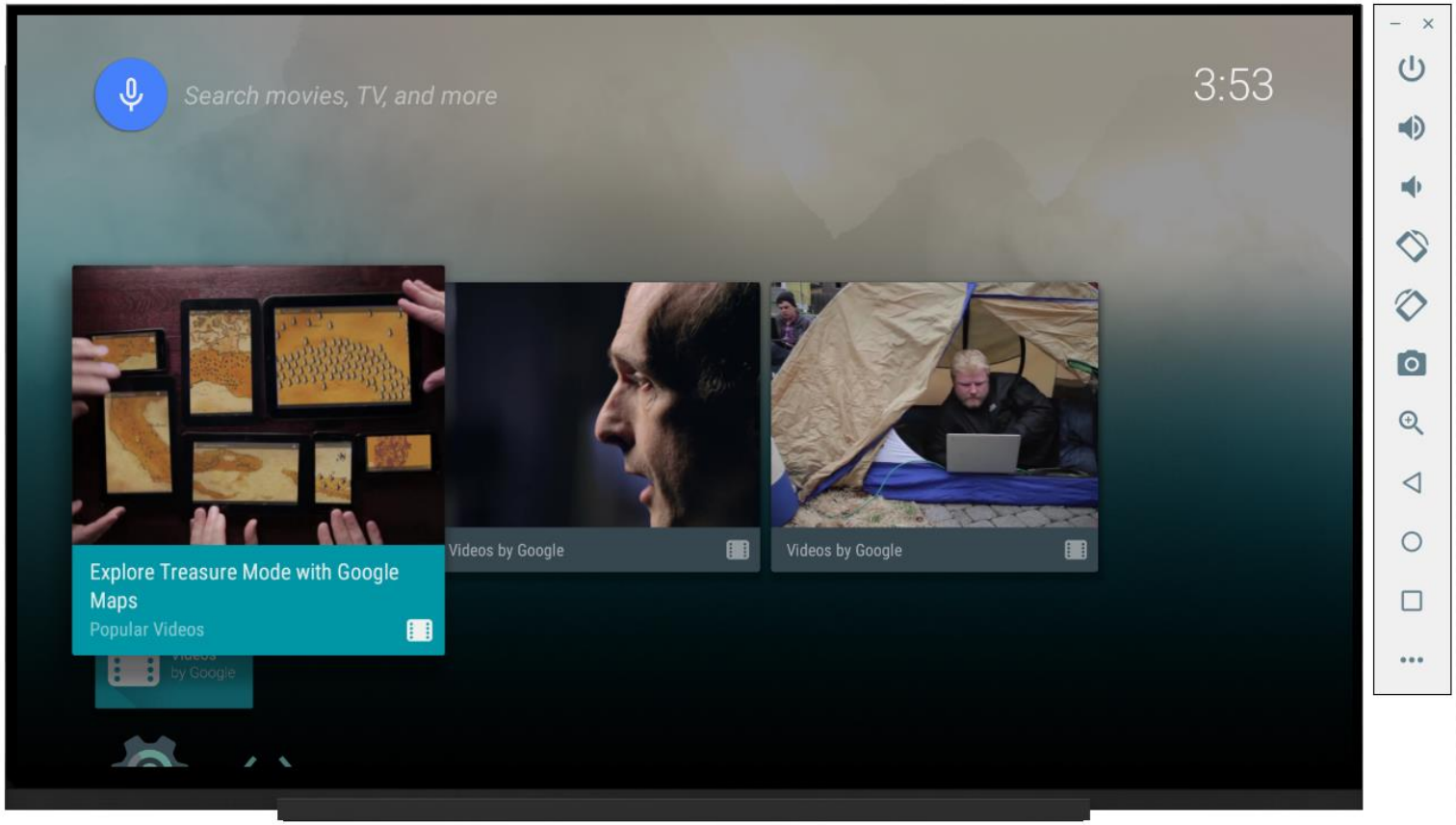
Creating Virtual Device V



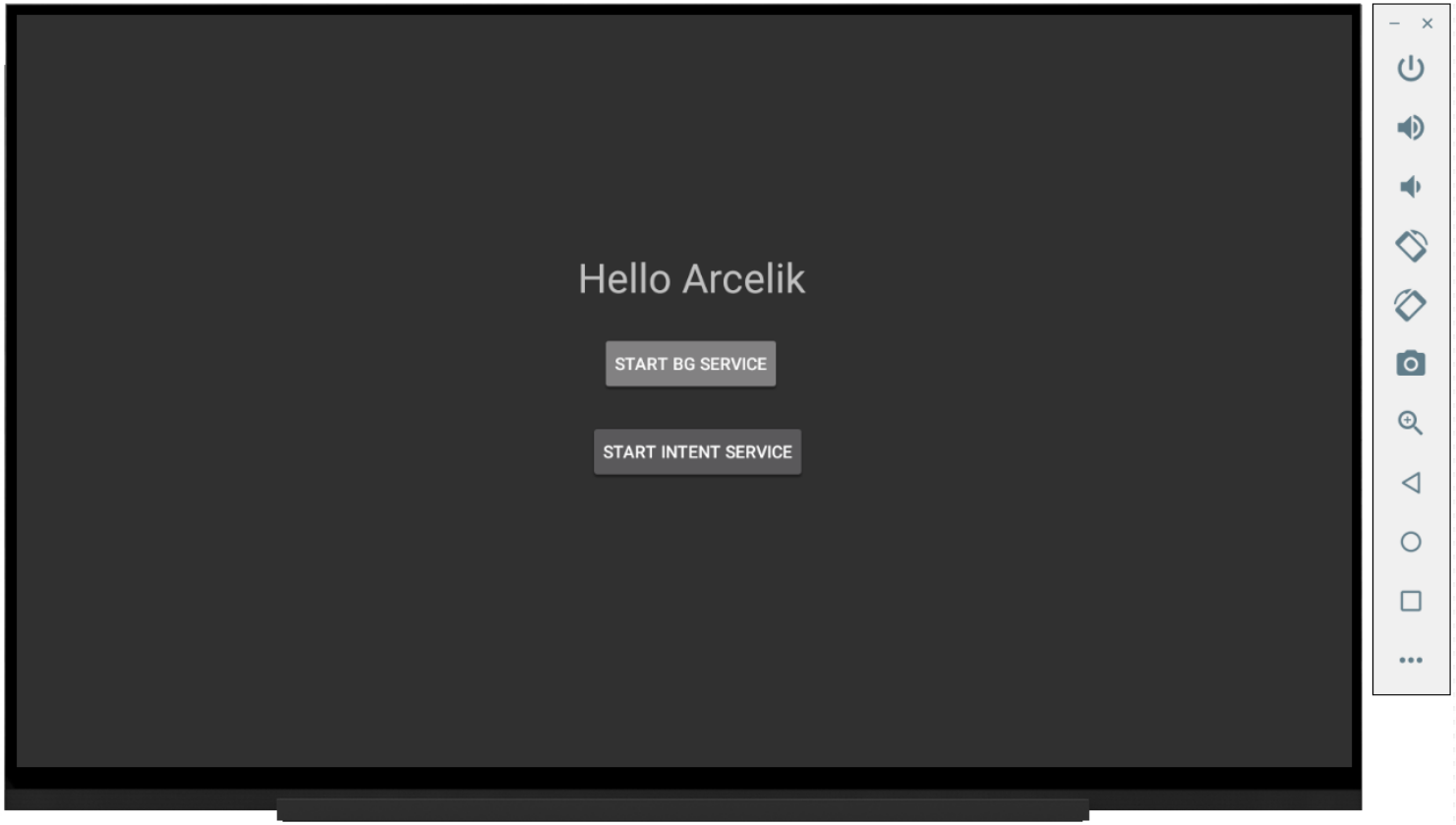
Running App on Emulator I



Running App on Emulator II



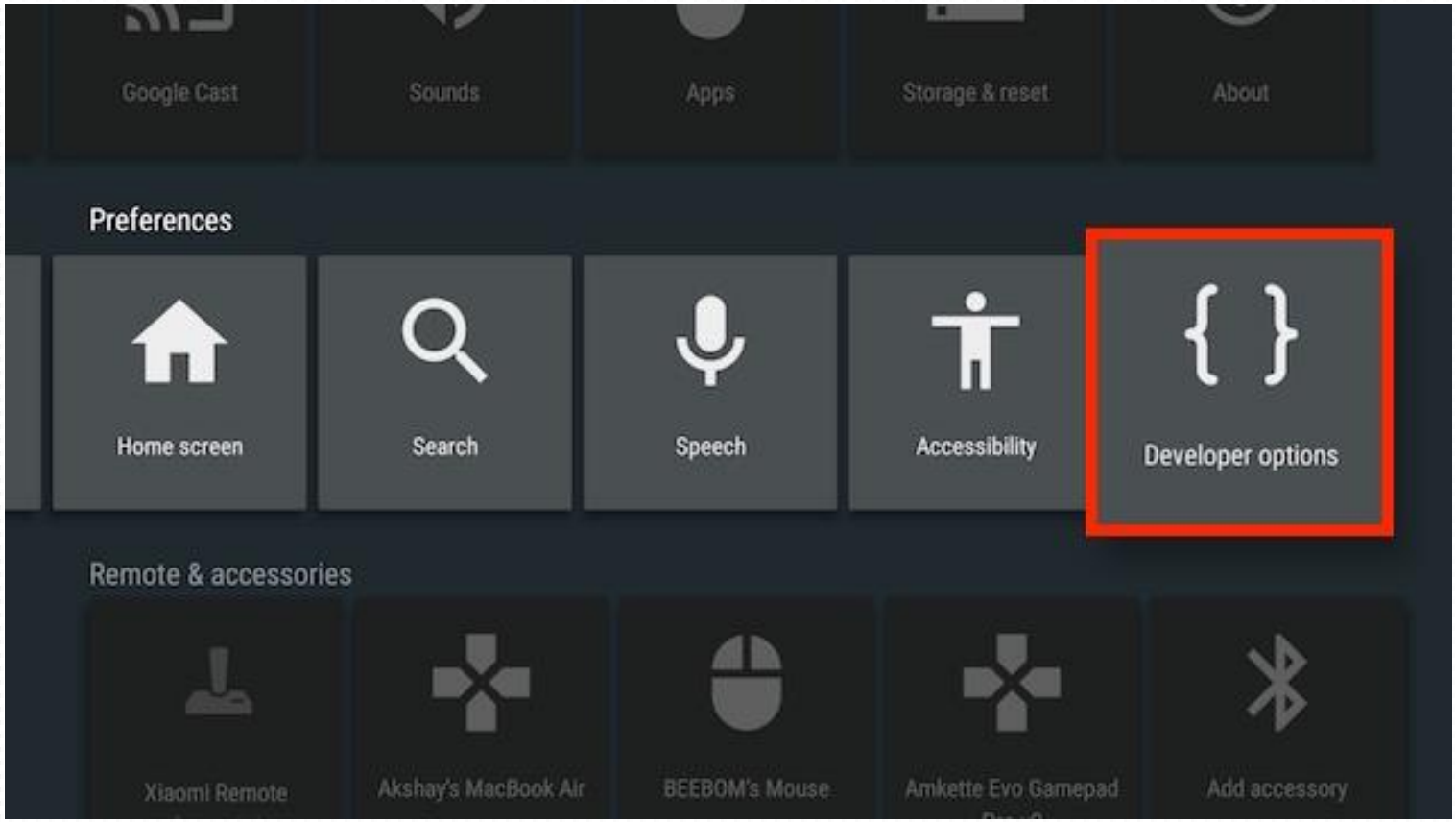
Running App on Emulator III



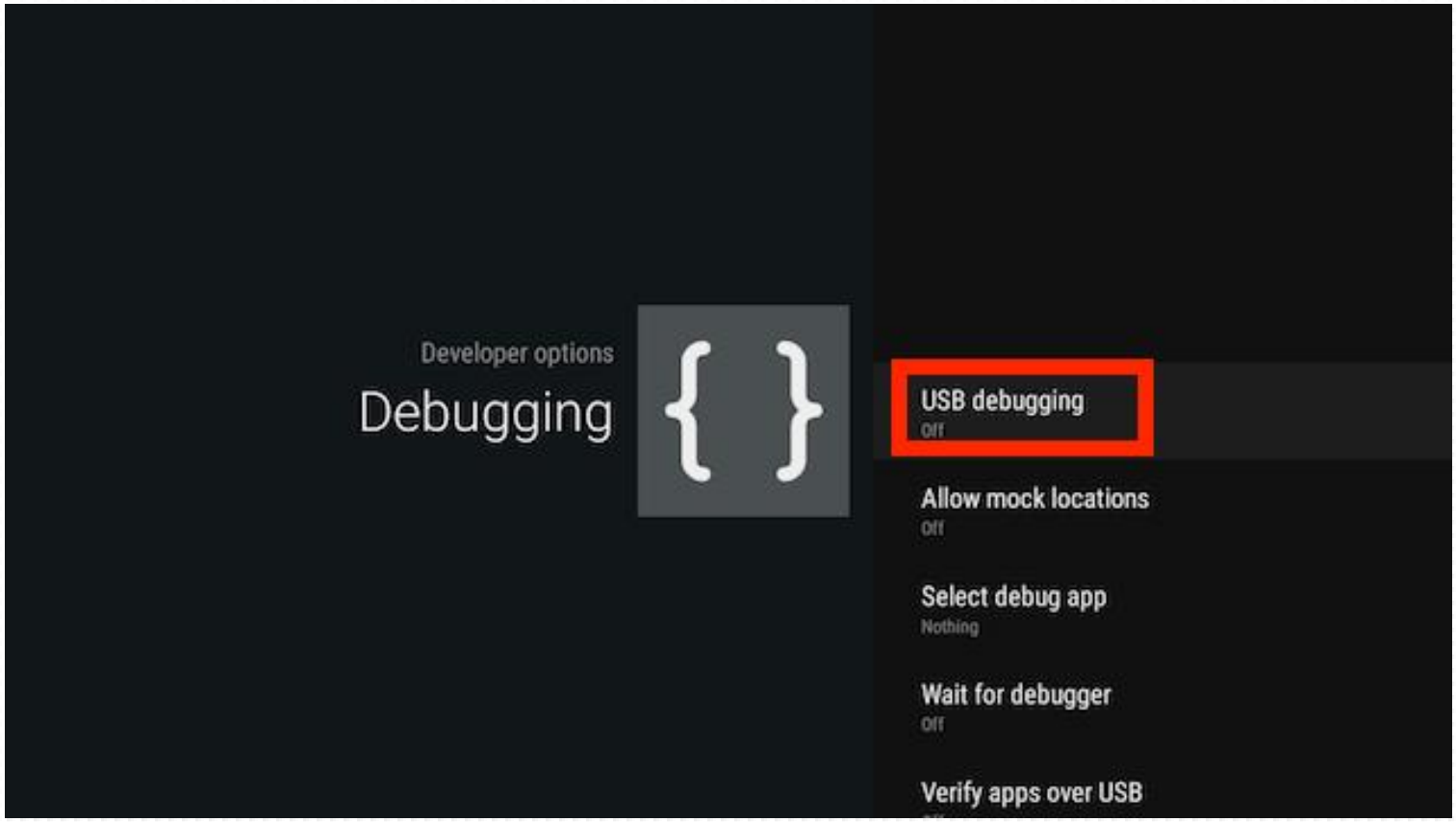
Running App on Device

- Android Debug Bridge (adb) tool is used to communicate with a device
- adb can be used with
  - Network
  - USB
- To use adb with a device connected over USB, you must enable **USB debugging** in the device system settings
- To use adb with a device connected over network, you must enable **ADB debugging** in the device system settings

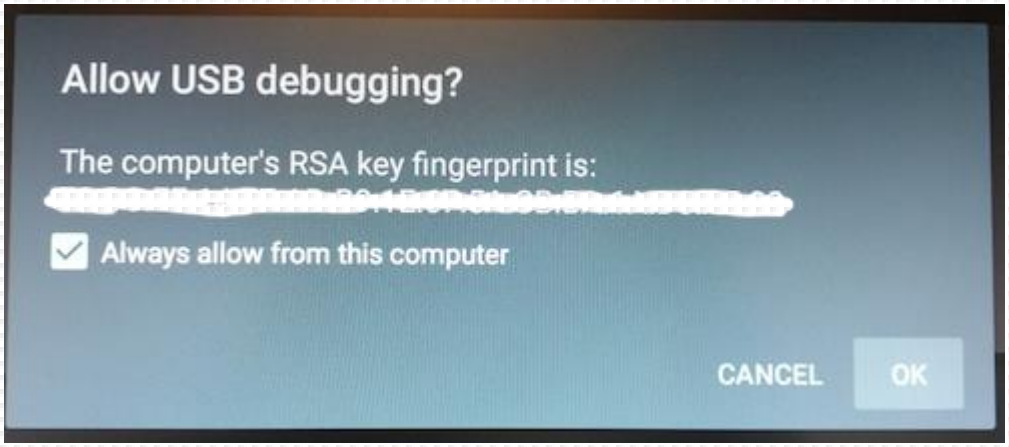
Connect Device (Android TV) over USB I



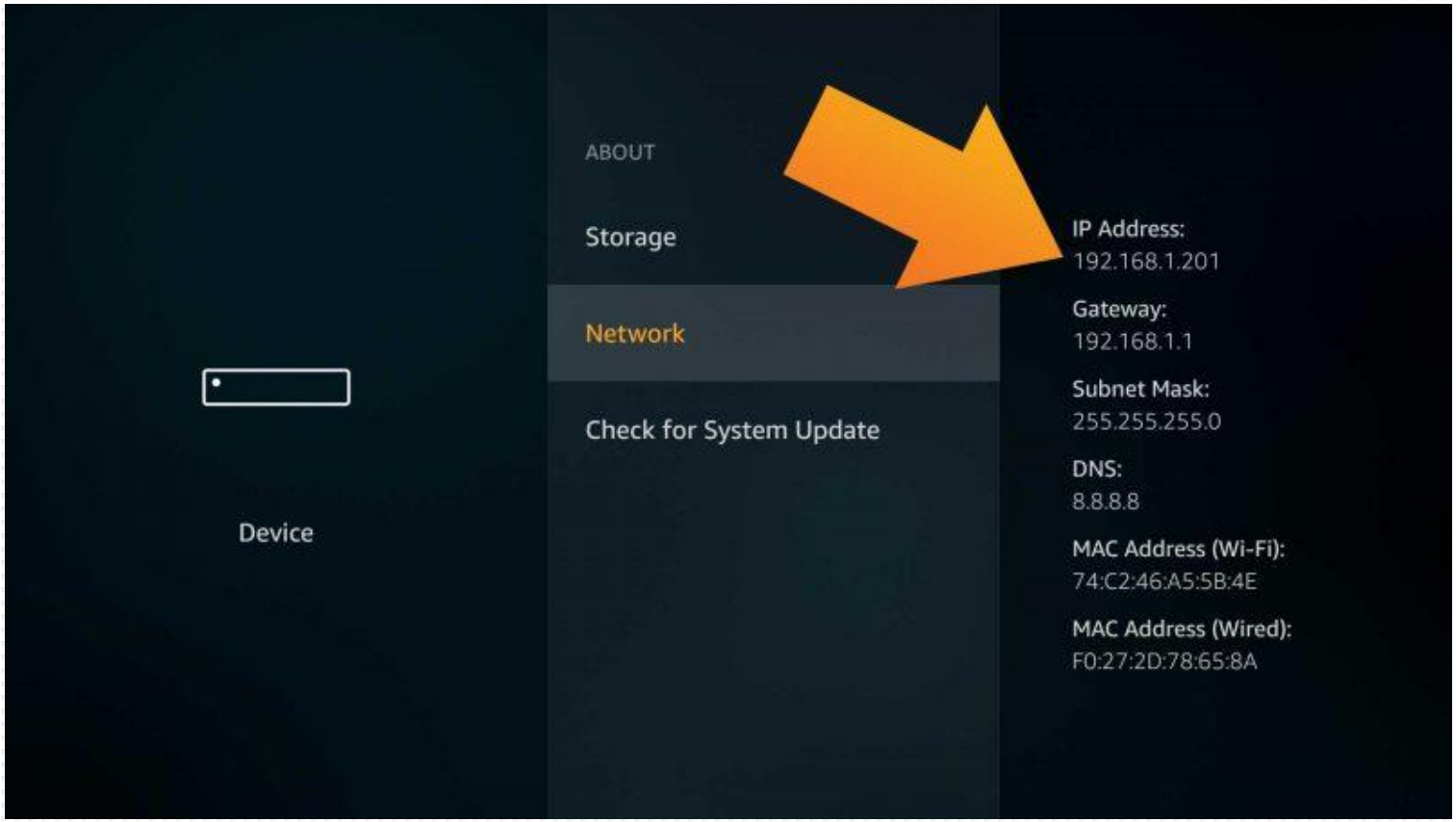
Connect Device (Android TV) over USB II



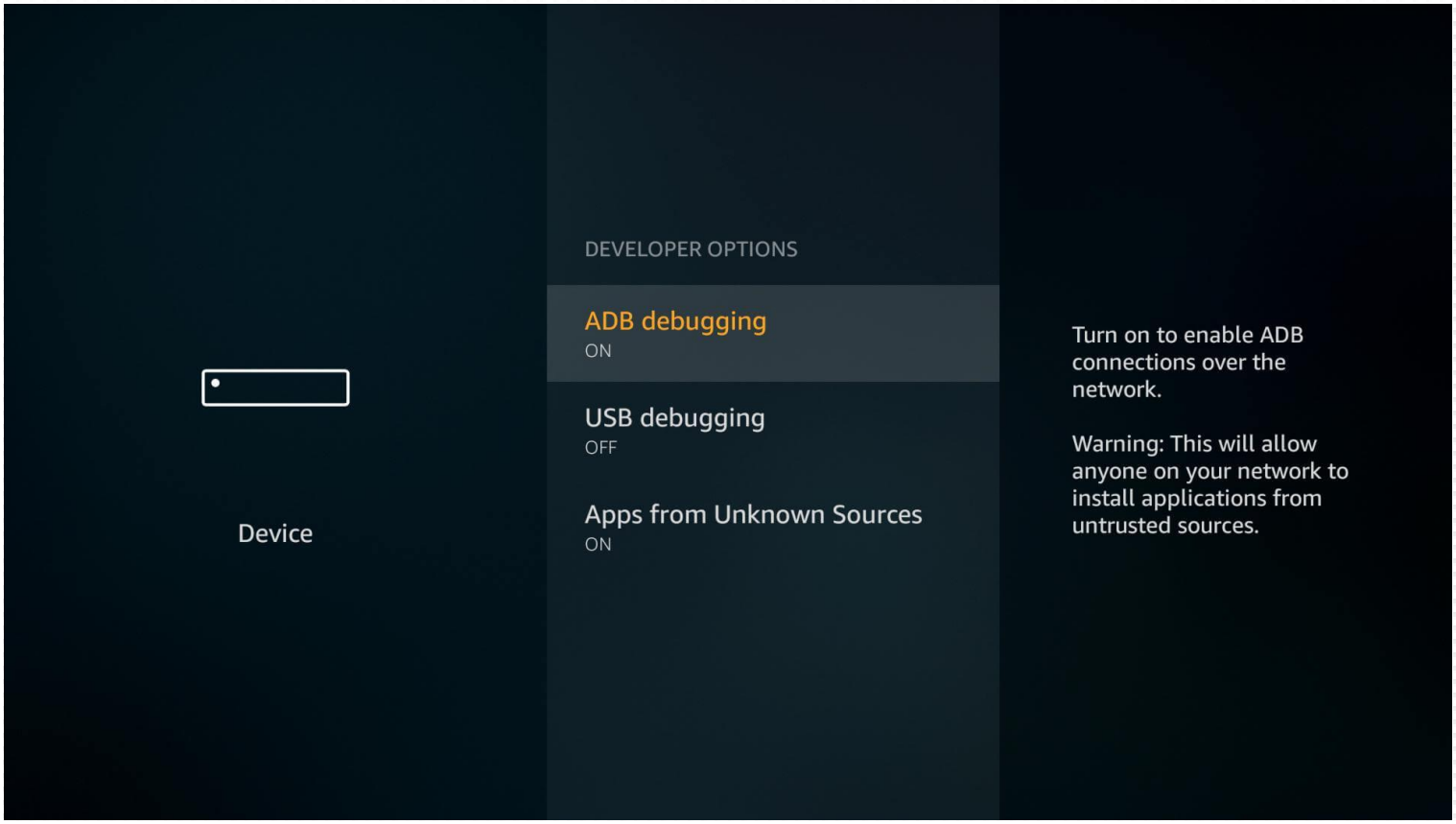
Connect Device (Android TV) over USB III



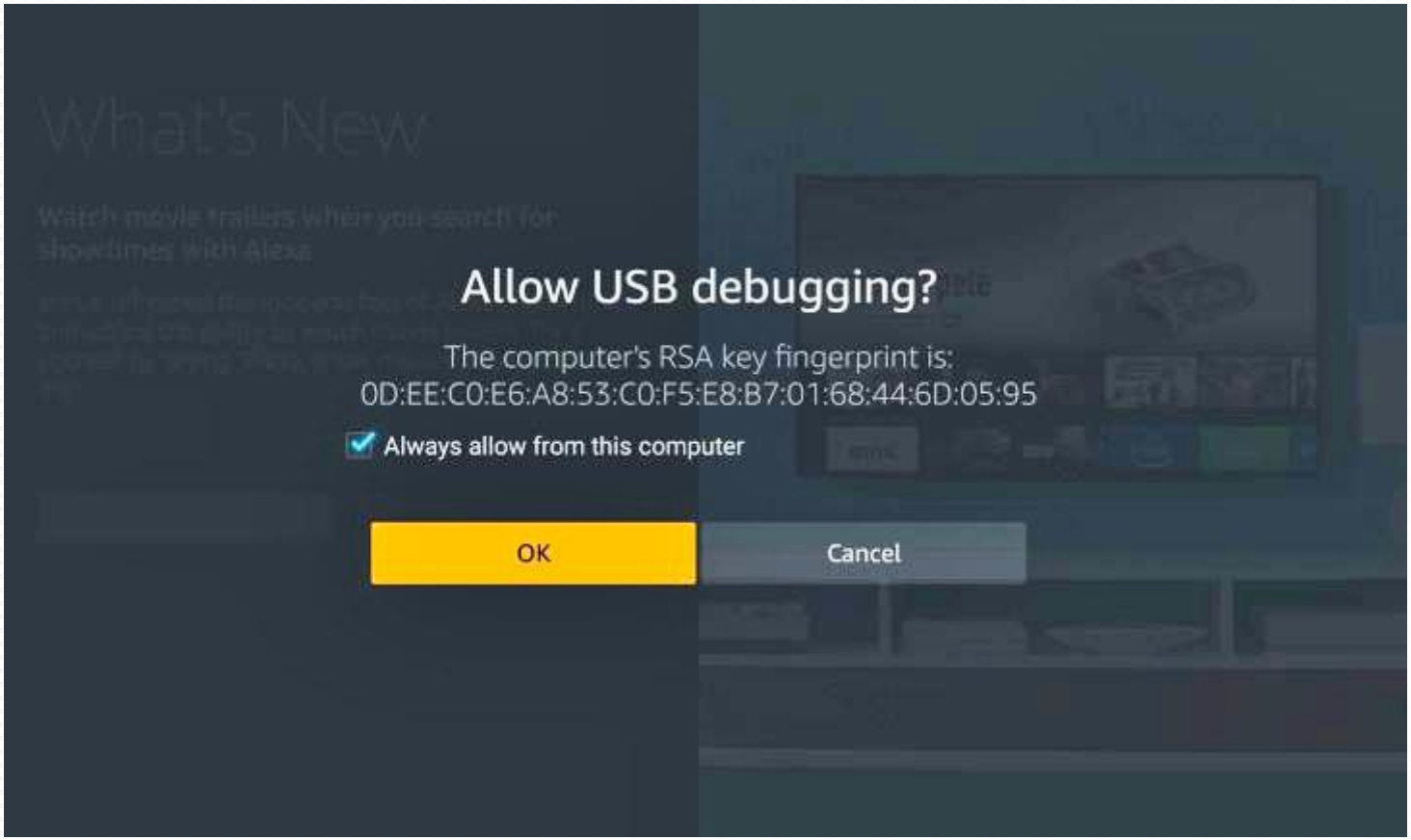
Connect Device (Fire TV) over Network I



Connect Device (Fire TV) over Network II



Connect Device (Fire TV) over Network

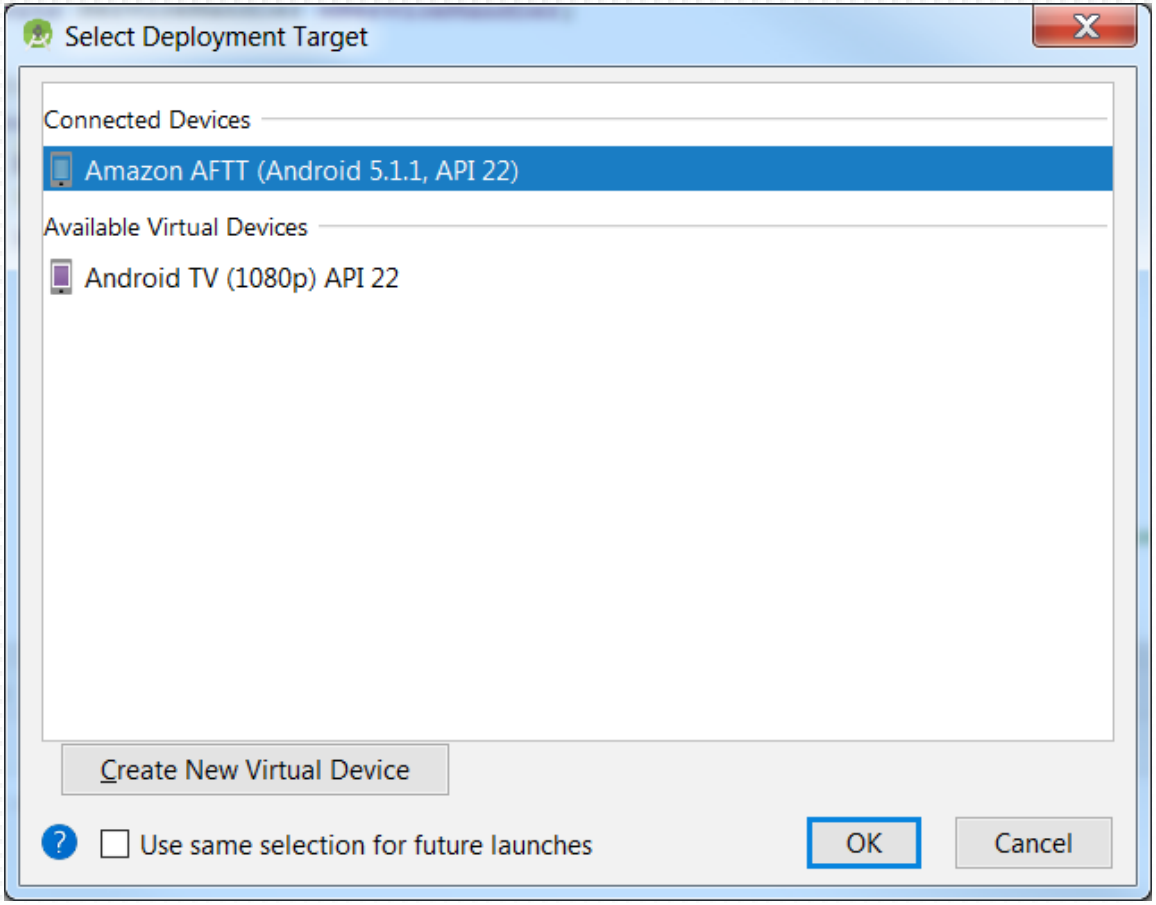


Connect Device (Fire TV) over Network III

Terminal

```
D:\Users\AR430805\AndroidStudioProjects\SampleApplicatio>cd D:\Users\AR430805\AppData\Local\Android\sdk\platform-tools

D:\Users\AR430805\AppData\Local\Android\sdk\platform-tools>adb start-server
* daemon not running; starting now at tcp:5037
* daemon started successfully

D:\Users\AR430805\AppData\Local\Android\sdk\platform-tools>adb connect 192.168.1.119
^C
D:\Users\AR430805\AppData\Local\Android\sdk\platform-tools>adb connect 192.168.0.119
failed to authenticate to 192.168.0.119:5555

D:\Users\AR430805\AppData\Local\Android\sdk\platform-tools>adb connect 192.168.0.119
already connected to 192.168.0.119:5555

D:\Users\AR430805\AppData\Local\Android\sdk\platform-tools>adb devices
List of devices attached
192.168.0.119:5555      device
192.168.1.119:5555      offline


D:\Users\AR430805\AppData\Local\Android\sdk\platform-tools>
```

Running App on Device (Fire TV) I



Installing App on Device (Fire TV) II

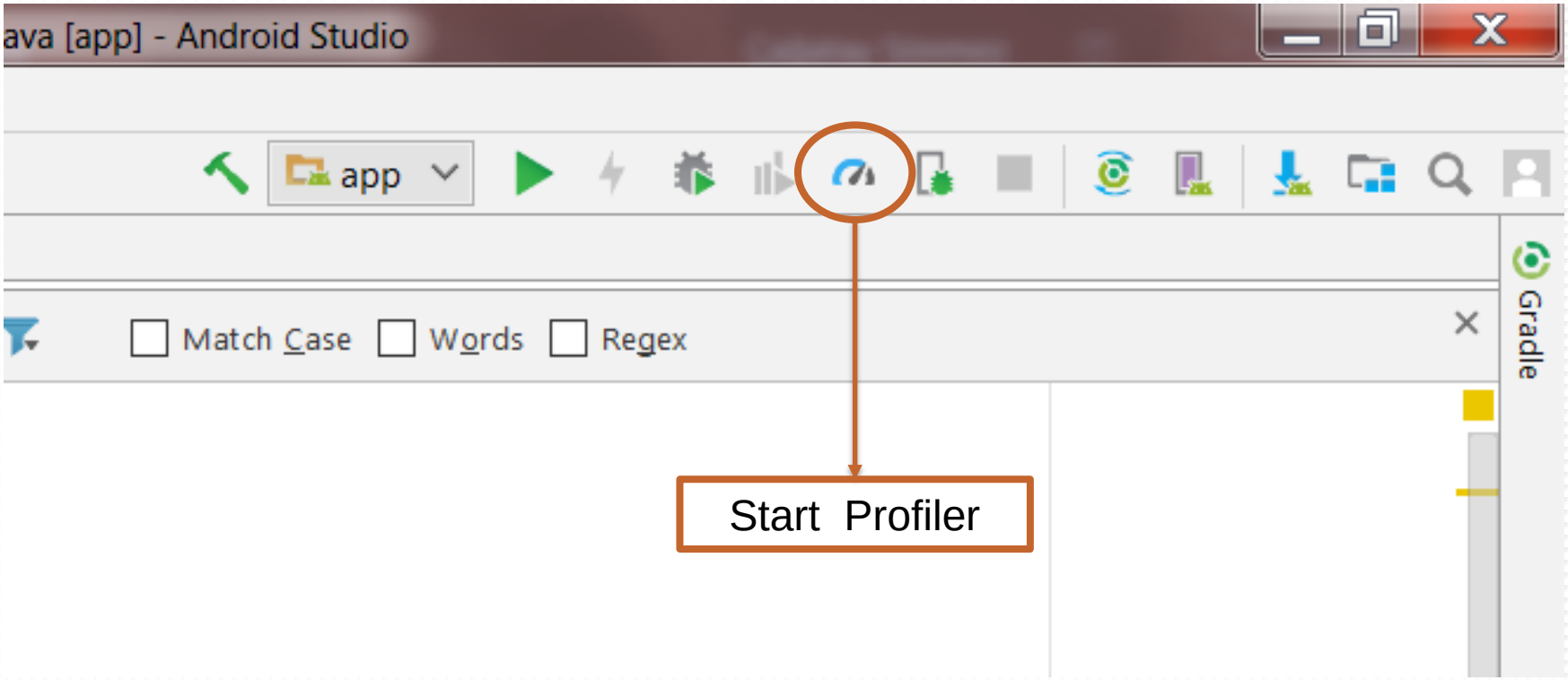
- You can also install external apk files to Android devices
- "Apps from Unknown Sourced" option should be ‘ON’ in developer options menu

adb install options

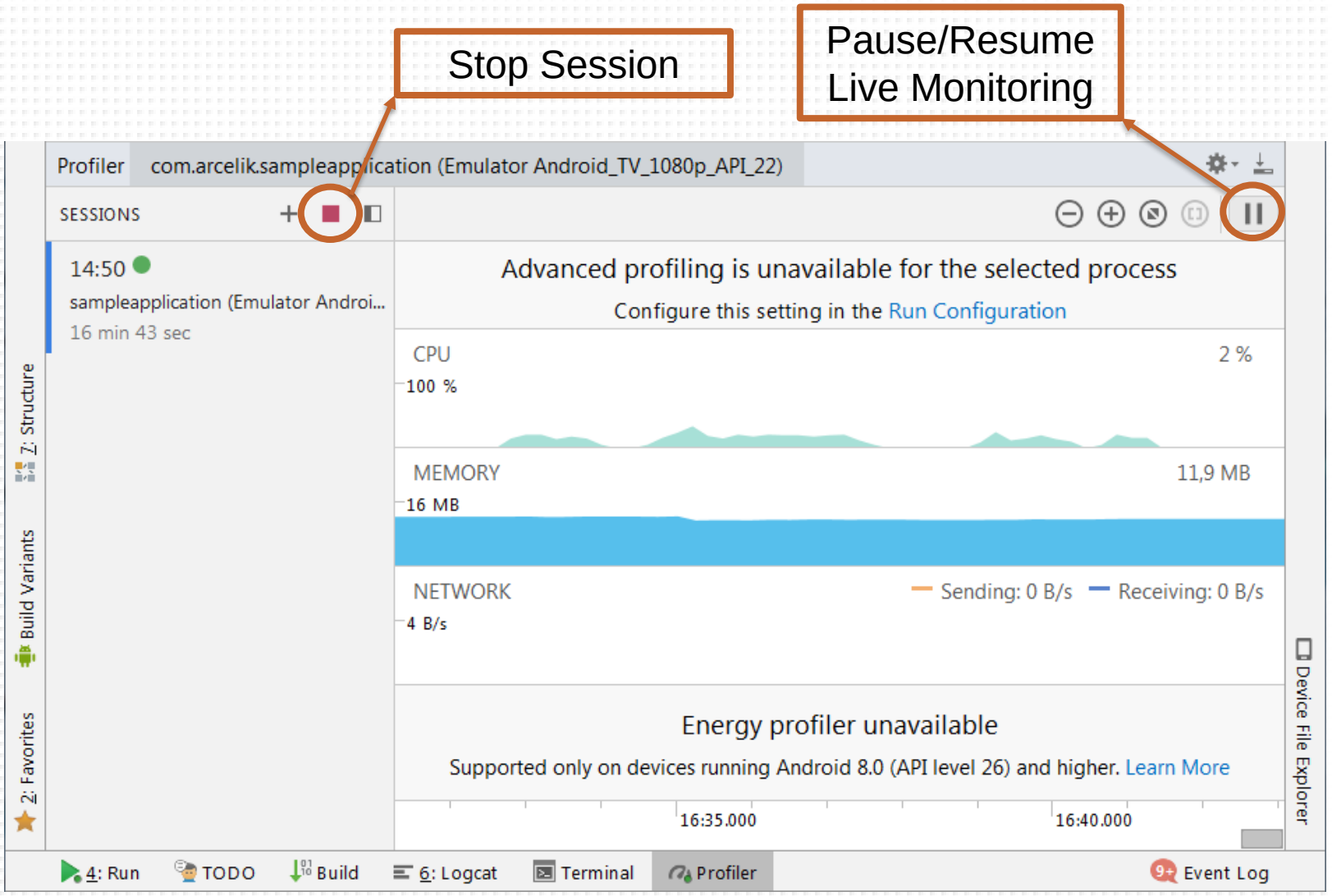
Command	Action
adb install test.apk	
adb install **-l** test.apk	forward lock application
adb install **-r** test.apk	replace existing application
adb install **-t** test.apk	allow test packages
adb install **-s** test.apk	install application on sdcard
adb install **-d** test.apk	allow version code downgrade
adb install **-p** test.apk	partial application install

Monitoring App Performance

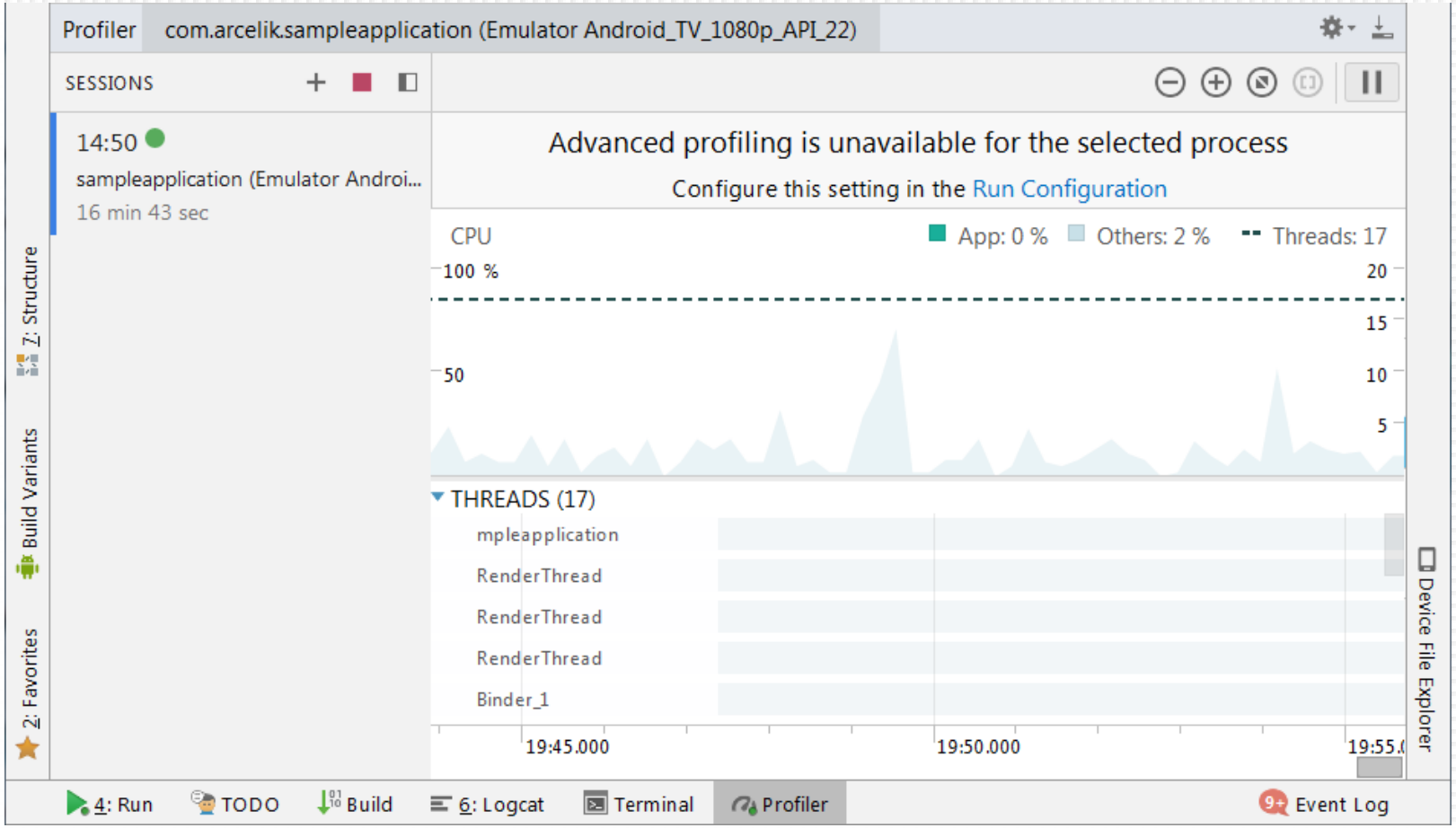
- Use Profiler to monitor the resource usage of your application



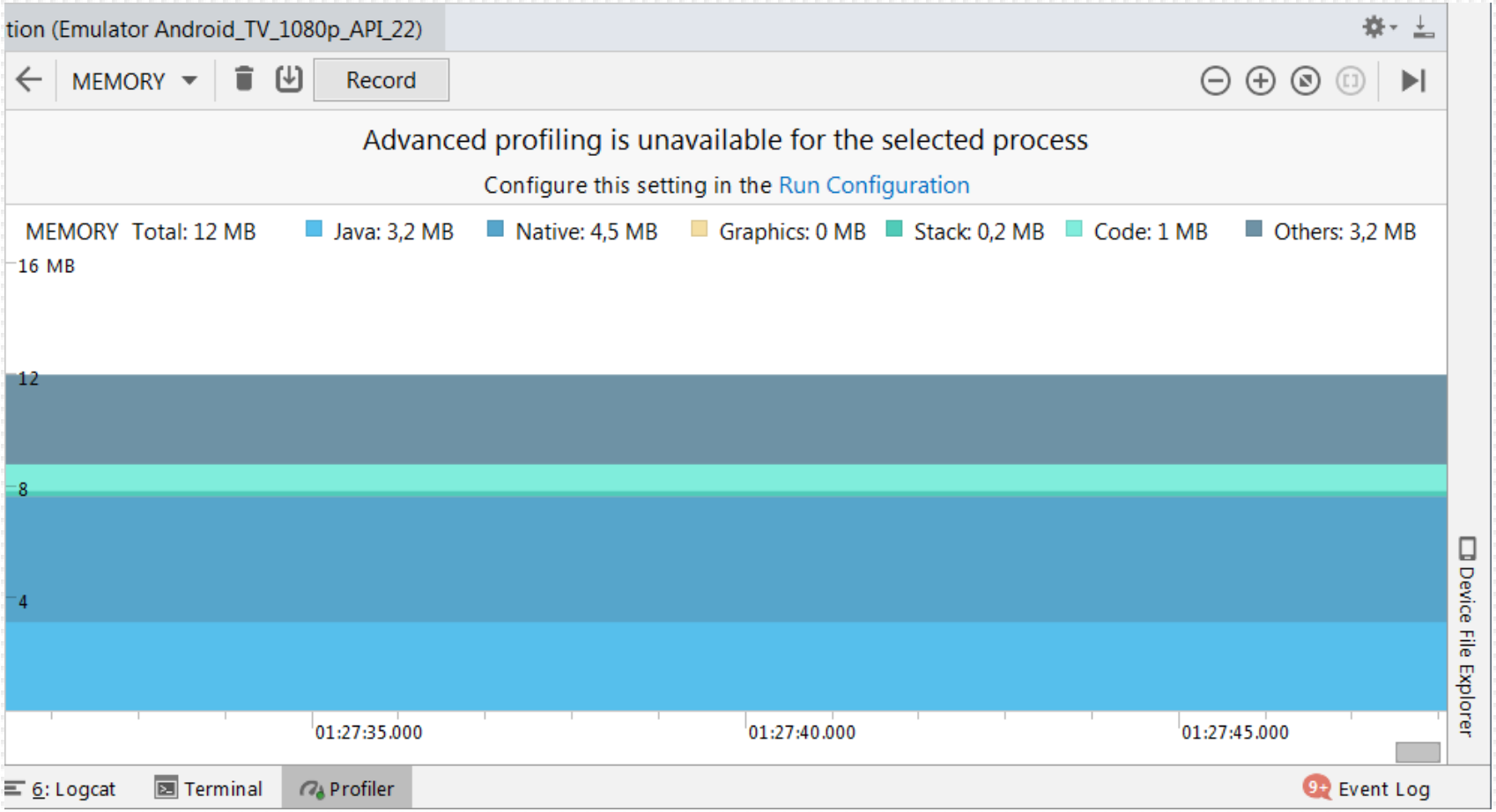
Profiler


Stop Session
Pause/Resume
Live Monitoring
Profiler
com.arcelik.sampleapplication (Emulator Android_TV_1080p_API_22)
SESSIONS
14:50
sampleapplication (Emulator Androi...
16 min 43 sec
Advanced profiling is unavailable for the selected process
Configure this setting in the Run Configuration
CPU
2 %
100 %
MEMORY
11,9 MB
16 MB
NETWORK
Sending: 0 B/s
Receiving: 0 B/s
4 B/s
Energy profiler unavailable
Supported only on devices running Android 8.0 (API level 26) and higher. Learn More
16:35.000
16:40.000
7: Structure
Build Variants
2: Favorites
Device File Explorer
4: Run
TODO
Build
6: Logcat
Terminal
Profiler
Event Log

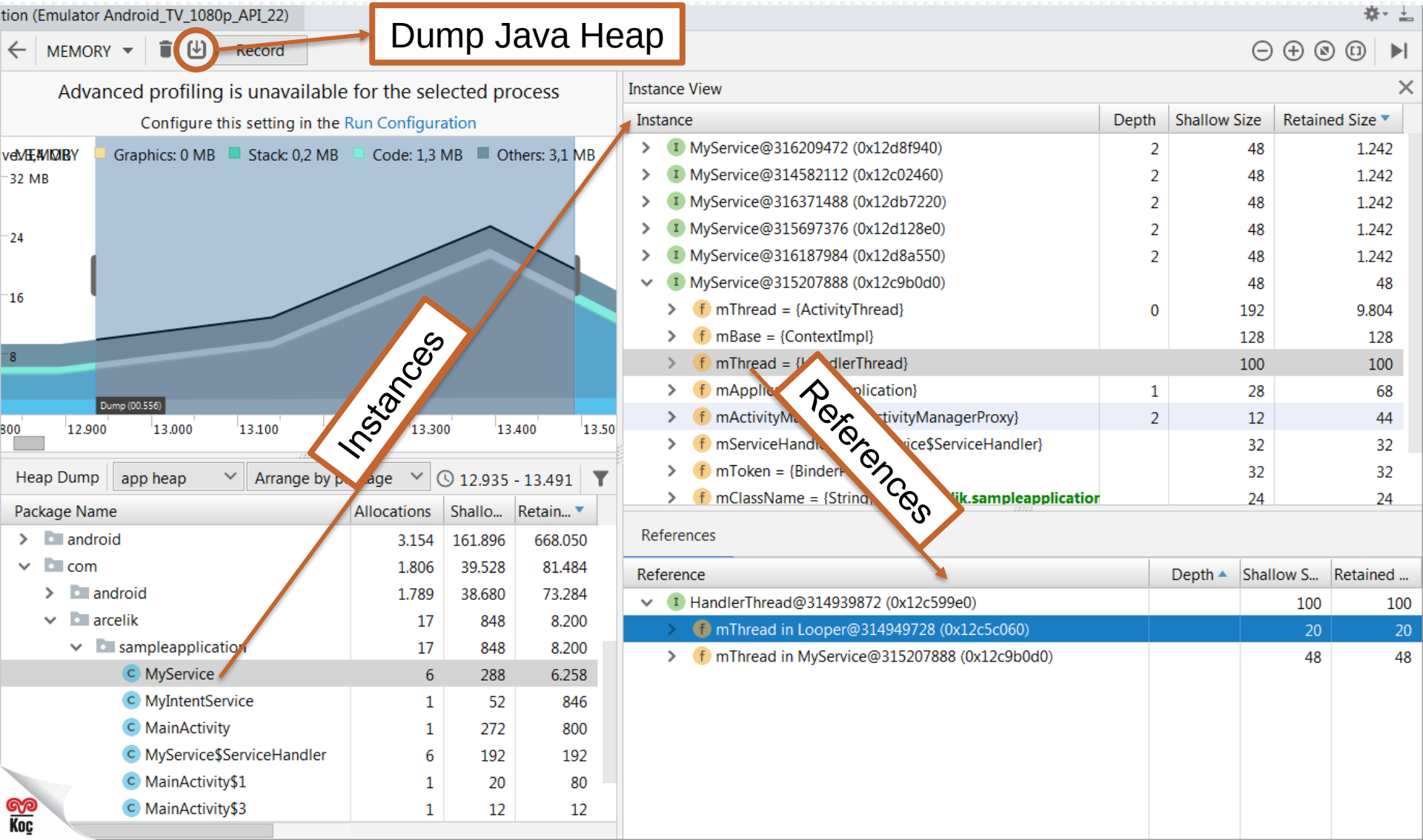
Profiler - CPU Usage



Profiler -Memeory Usage



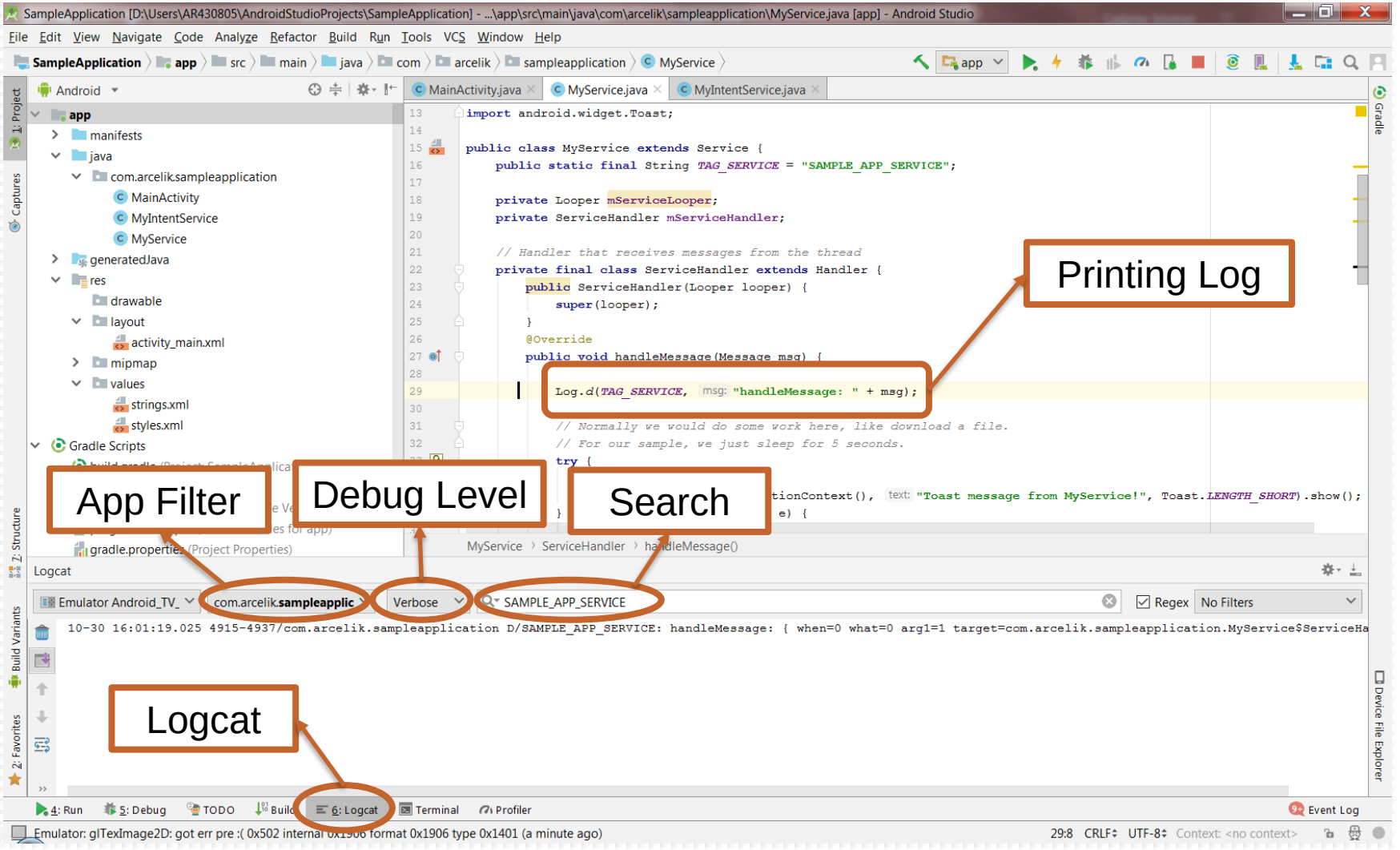
Profiler - Java Heap



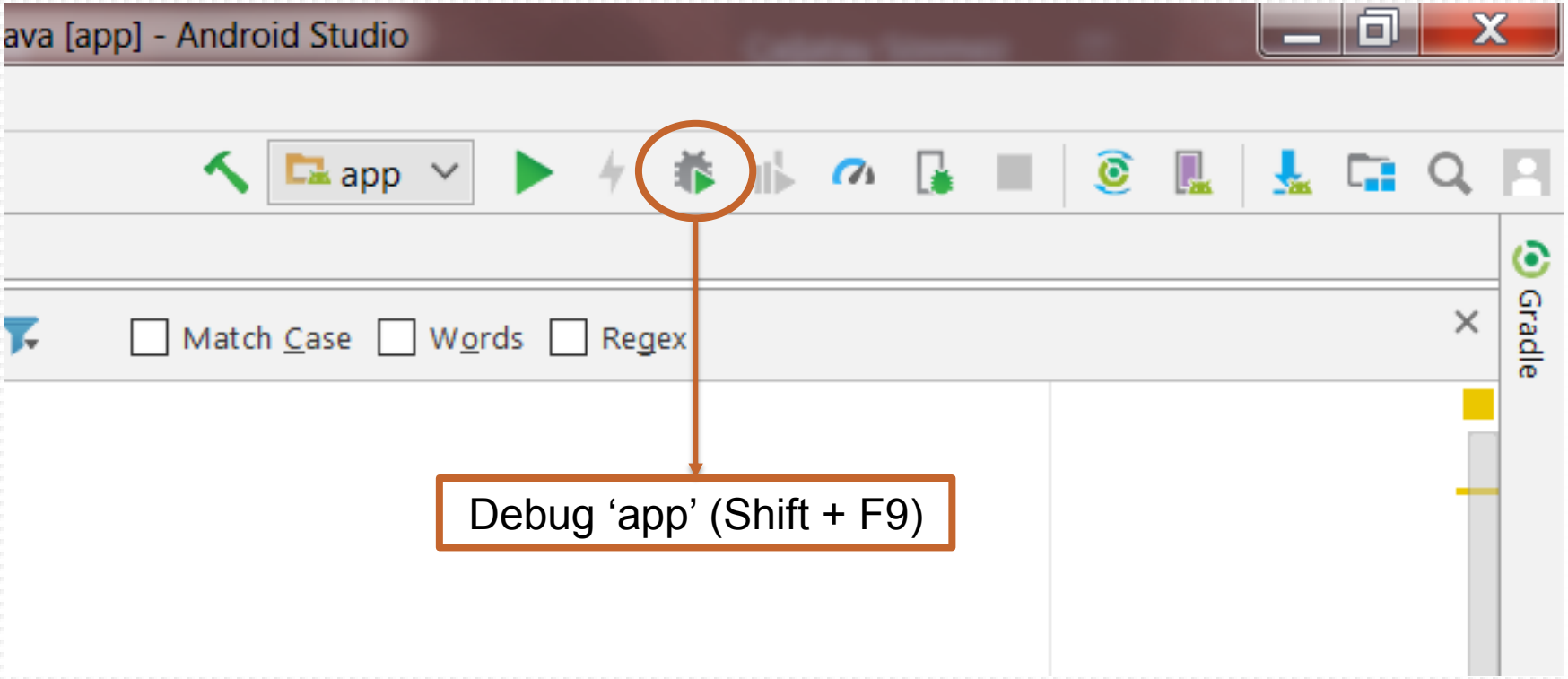
Debugging

- Android Studio provides a debugger to debug application
  - View logs
  - Attach the debugger to a running app
  - Use breakpoints
  - Analyze stack trace
  - View on-device files
  - Take screenshot
  - Record a video

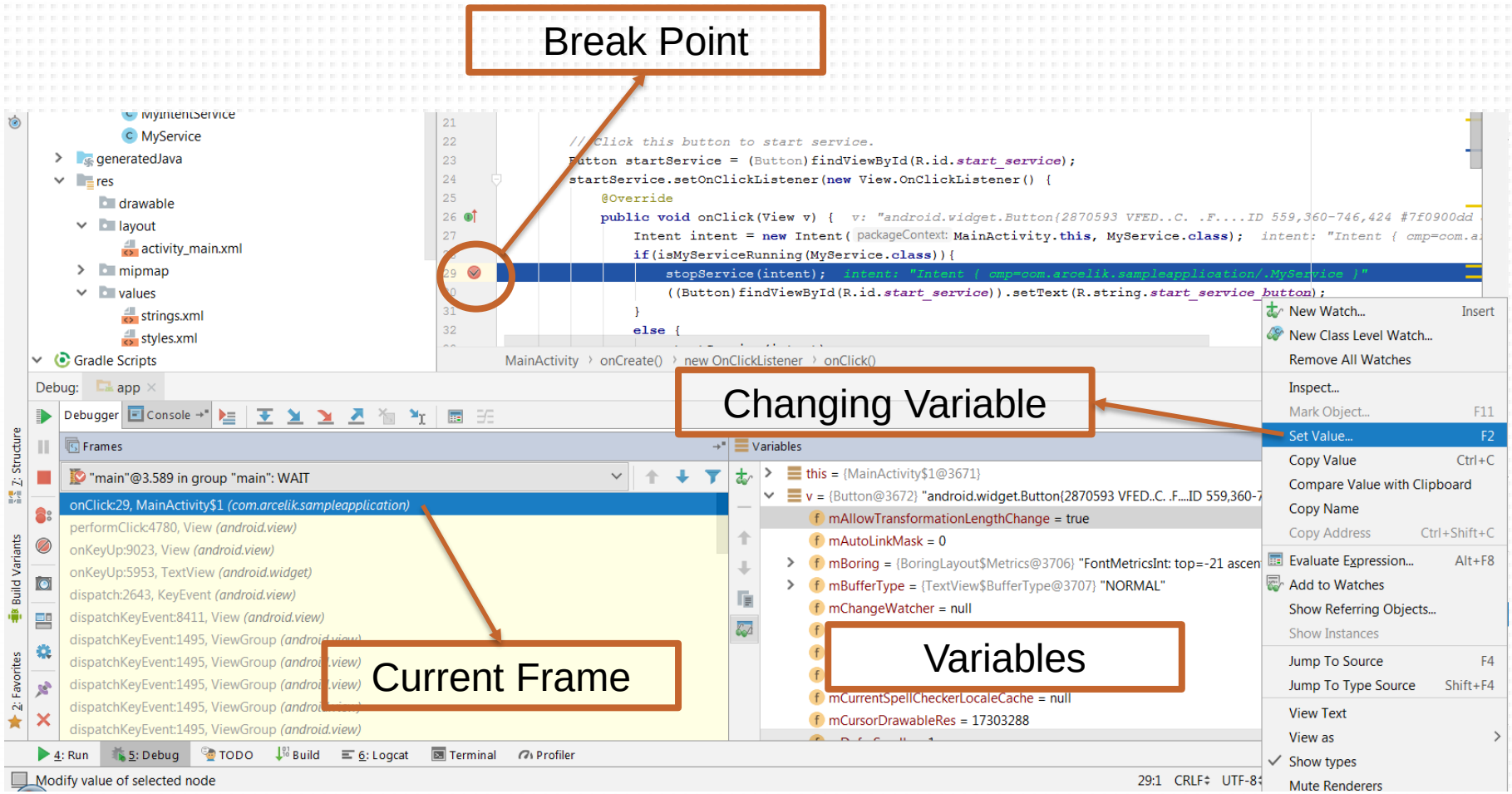
Viewing System Logs



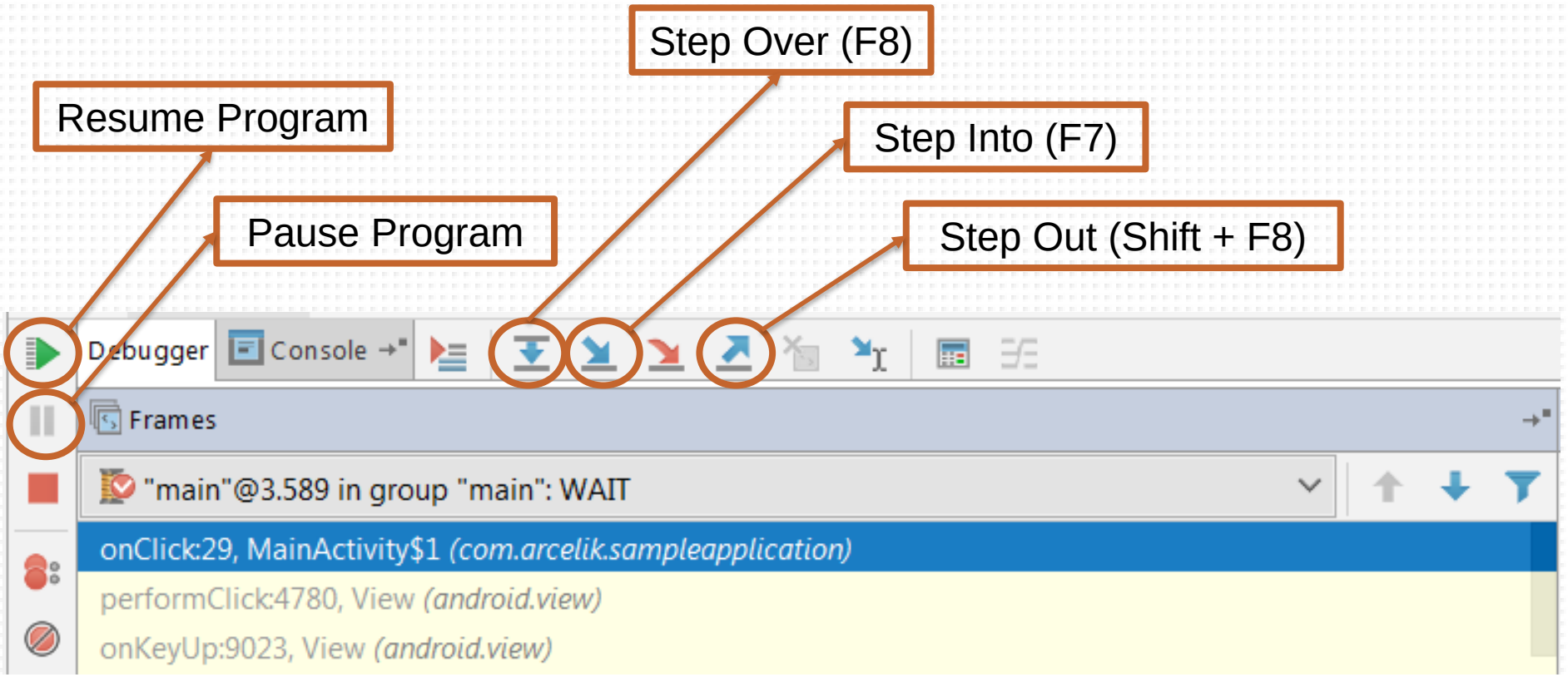
Starting Debugger



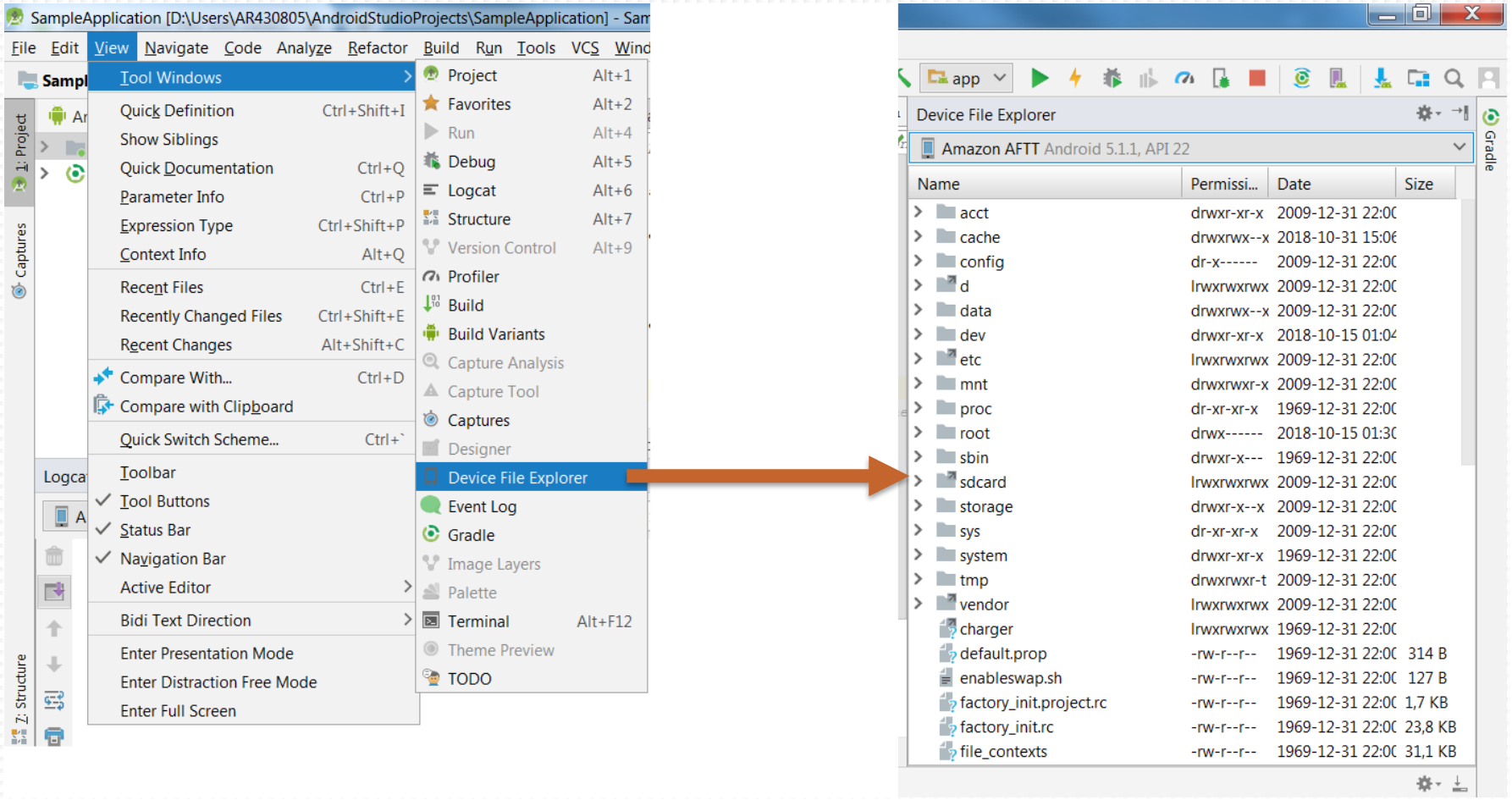
Using Breakpoint I



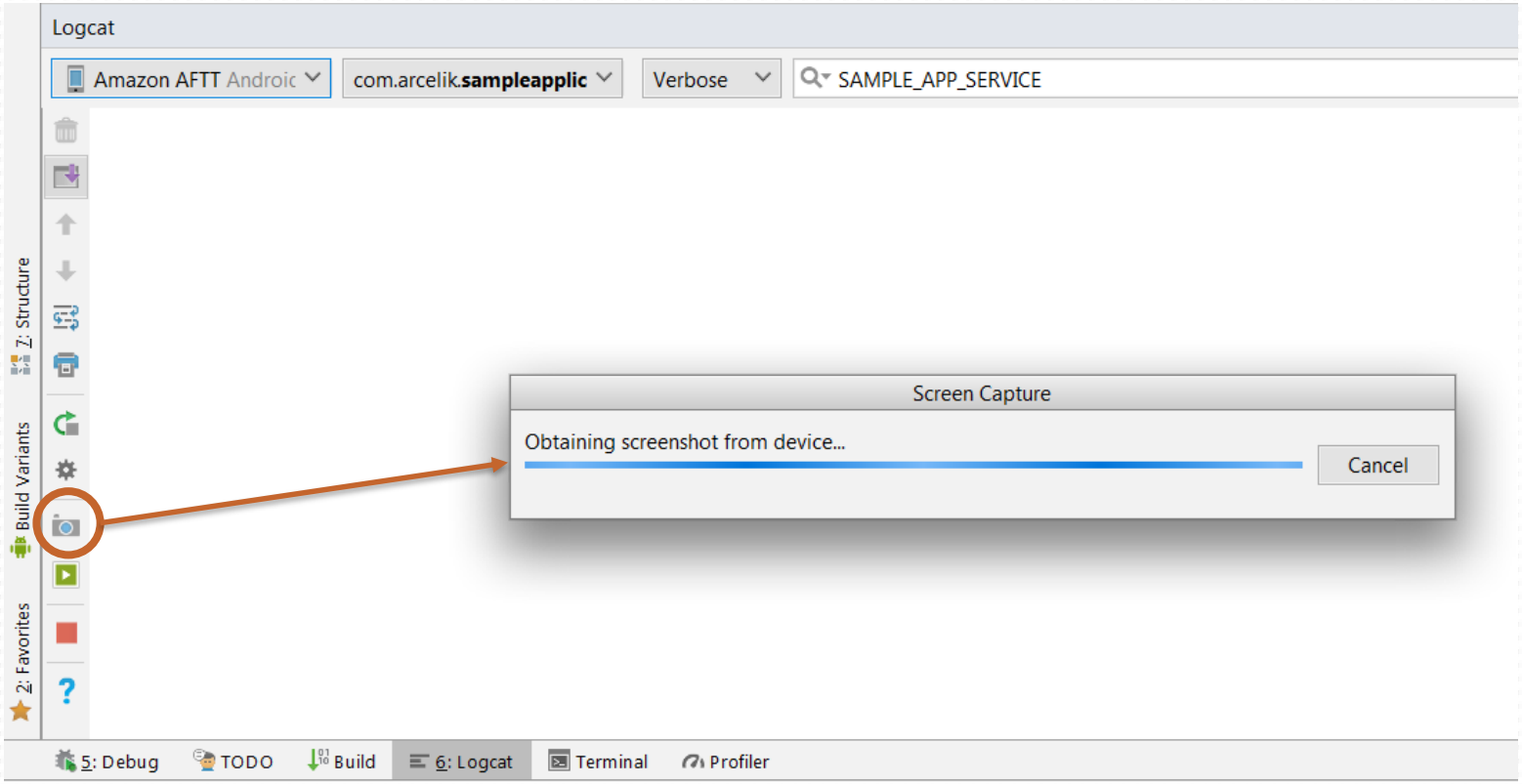
Using Breakpoint II



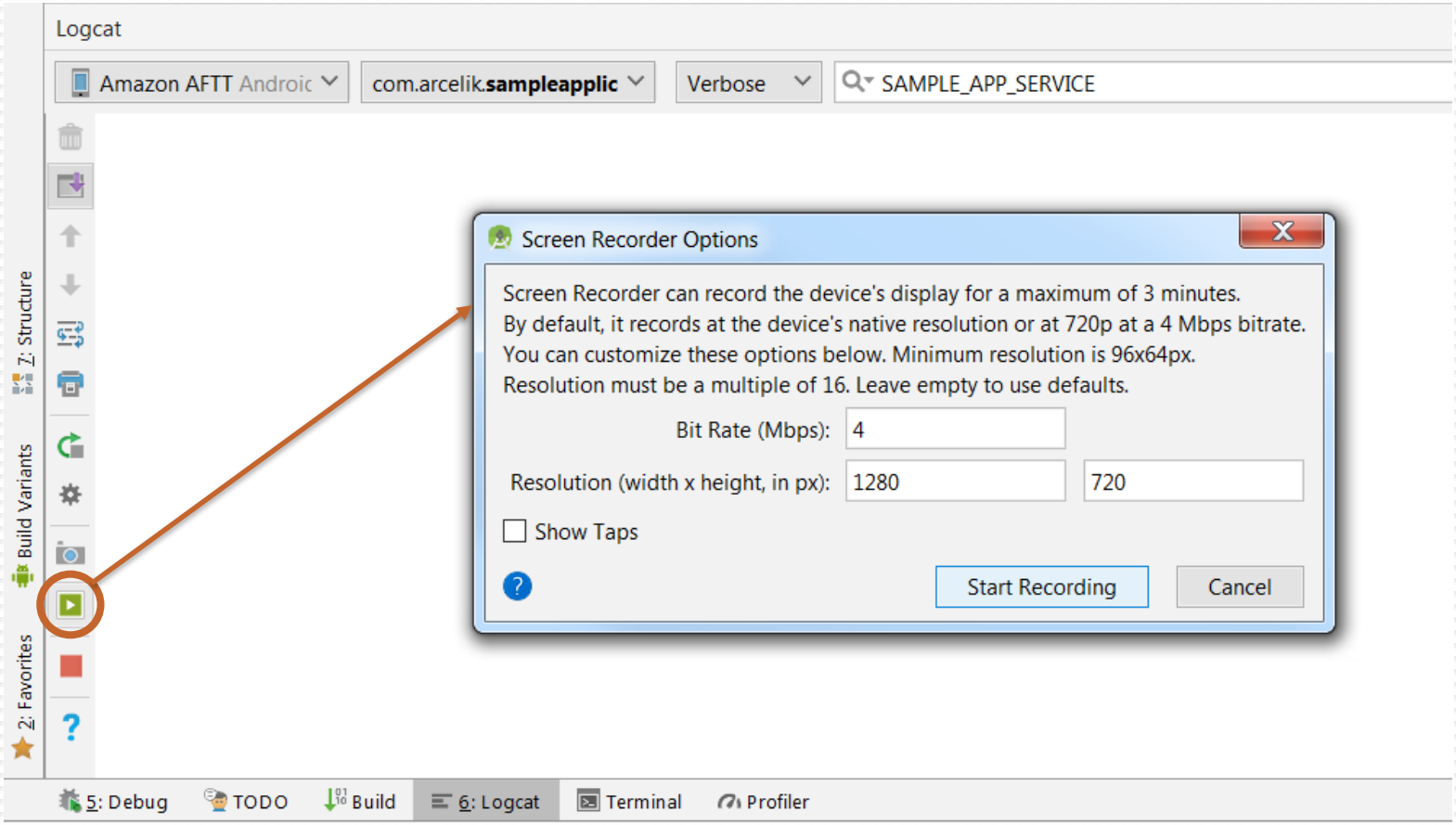
View On-Device Files



Take screenshot



Record Video



QUESTIONS?