Introduction to Android
Hello World! Application

Çağatay Sönmez

04.02.2021

Arcelik Electronics Plant, R&D Team - Software Design

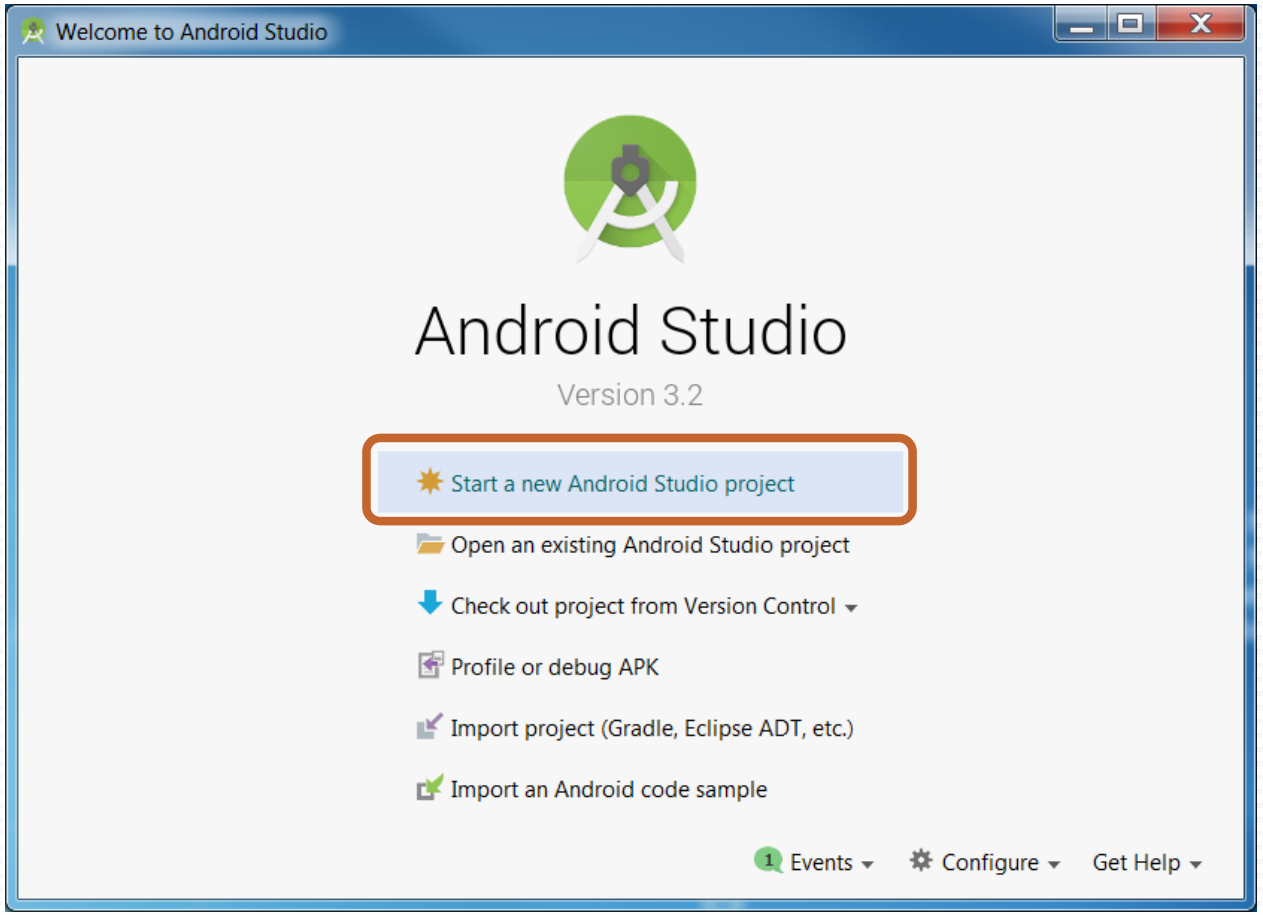
Agenda

- 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

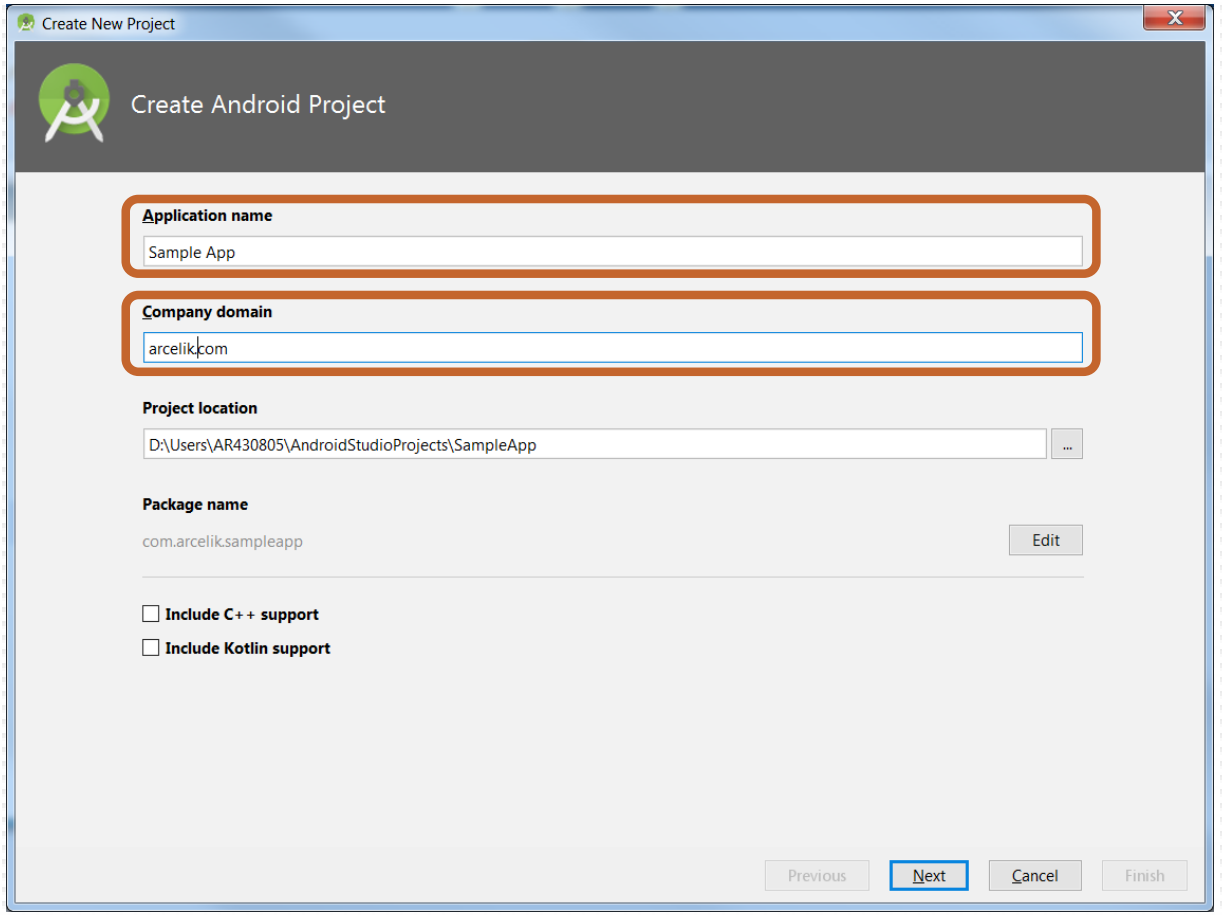
Meterials

- 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 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 website
  - https://developers.google.com/training/android/

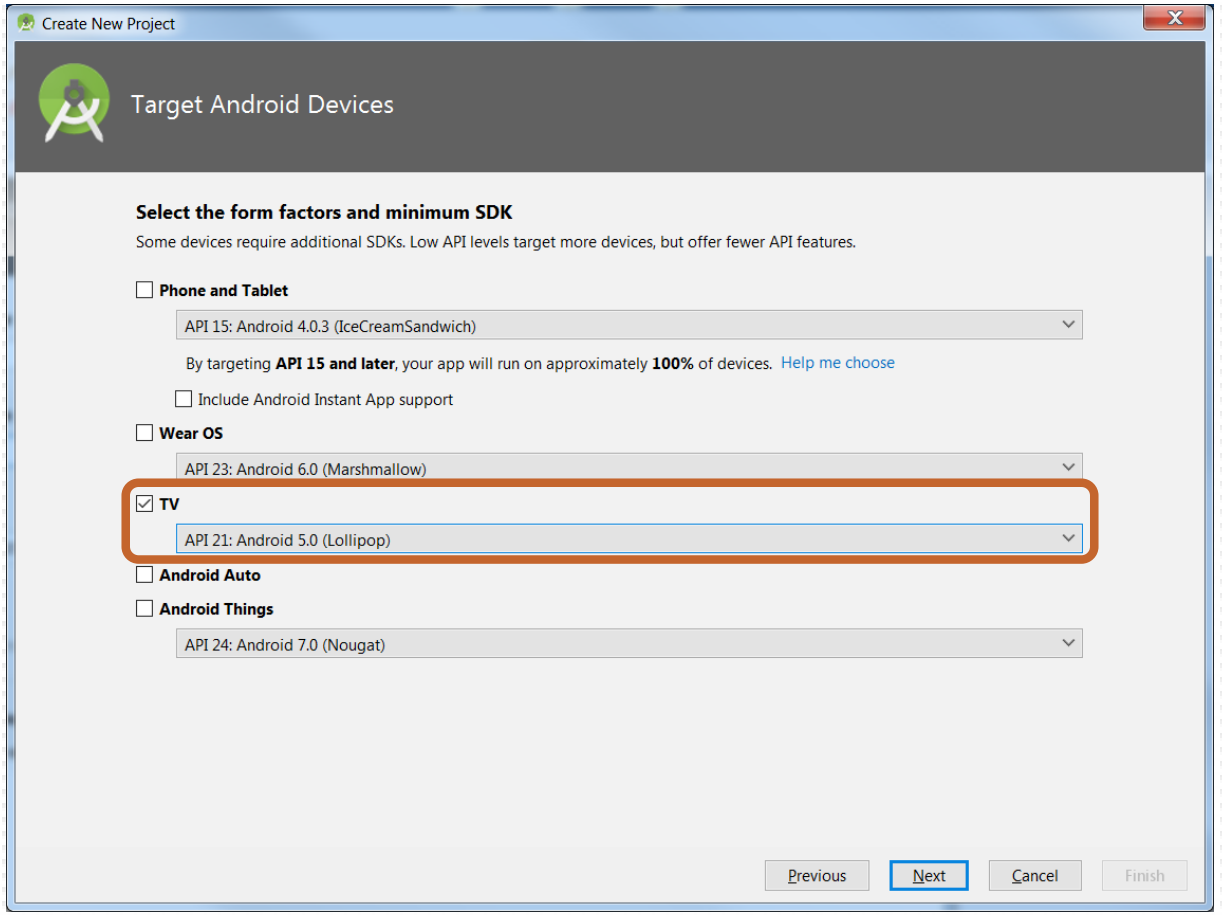
Creating Android TV App I



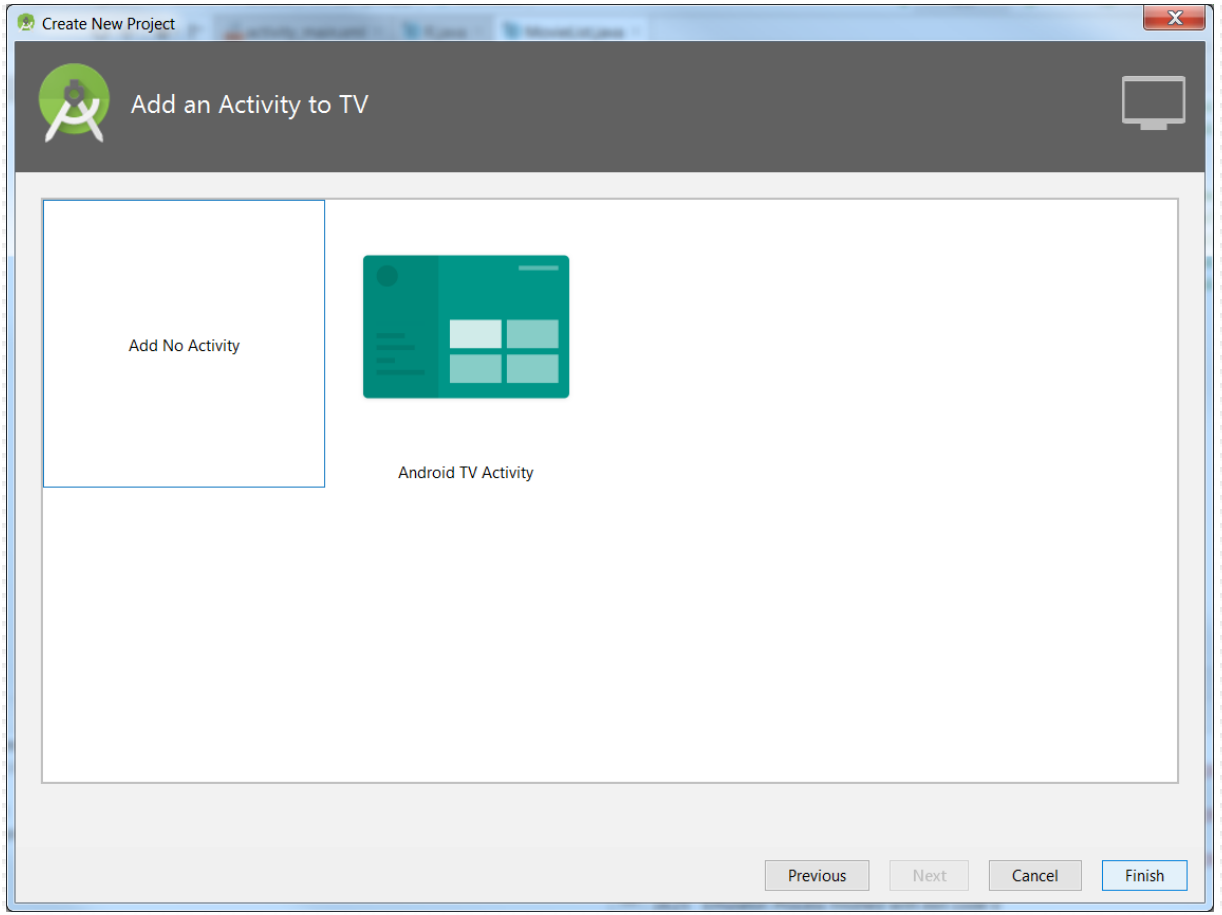
Creating Android TV App II



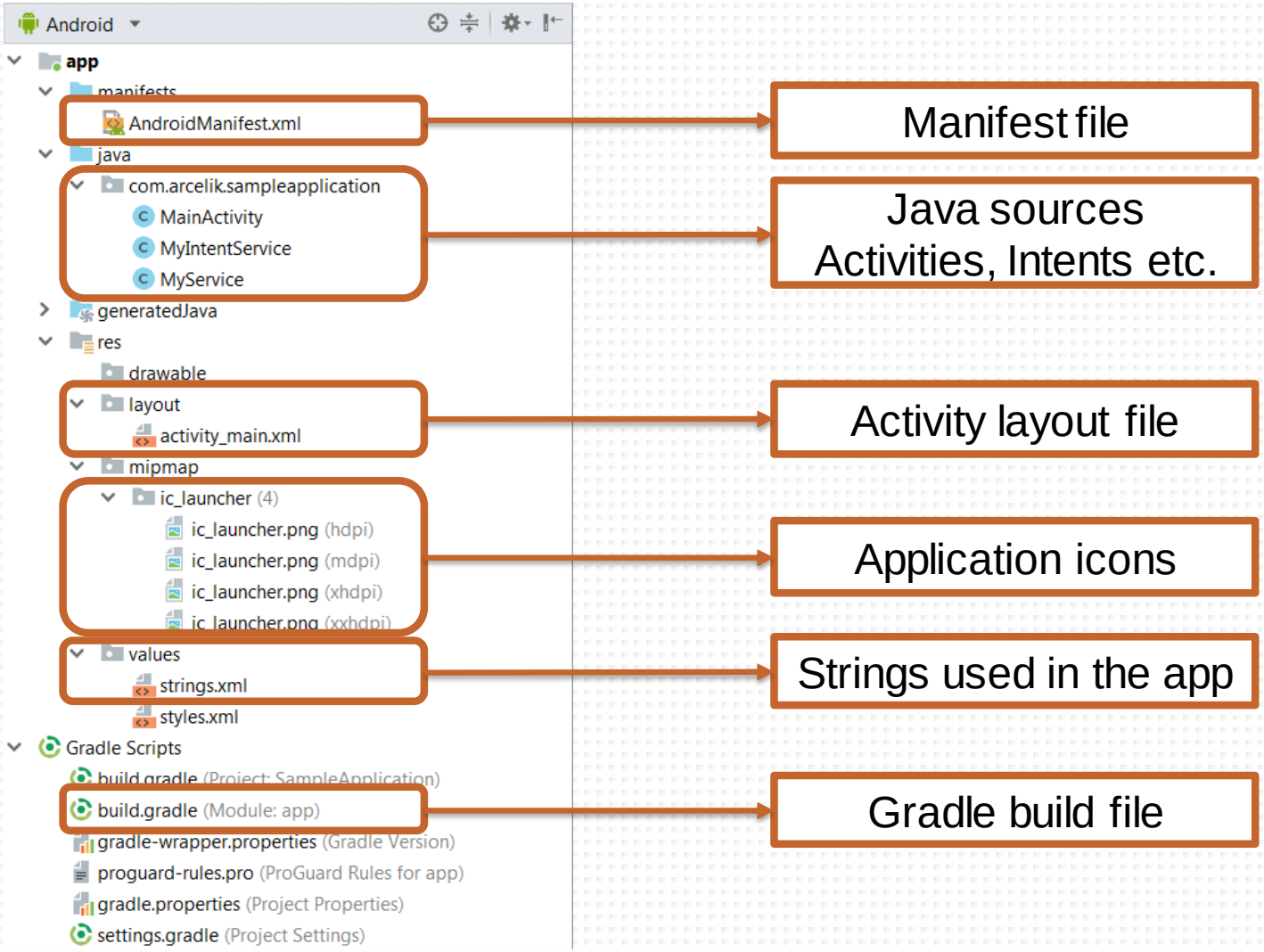
Creating Android TV App III



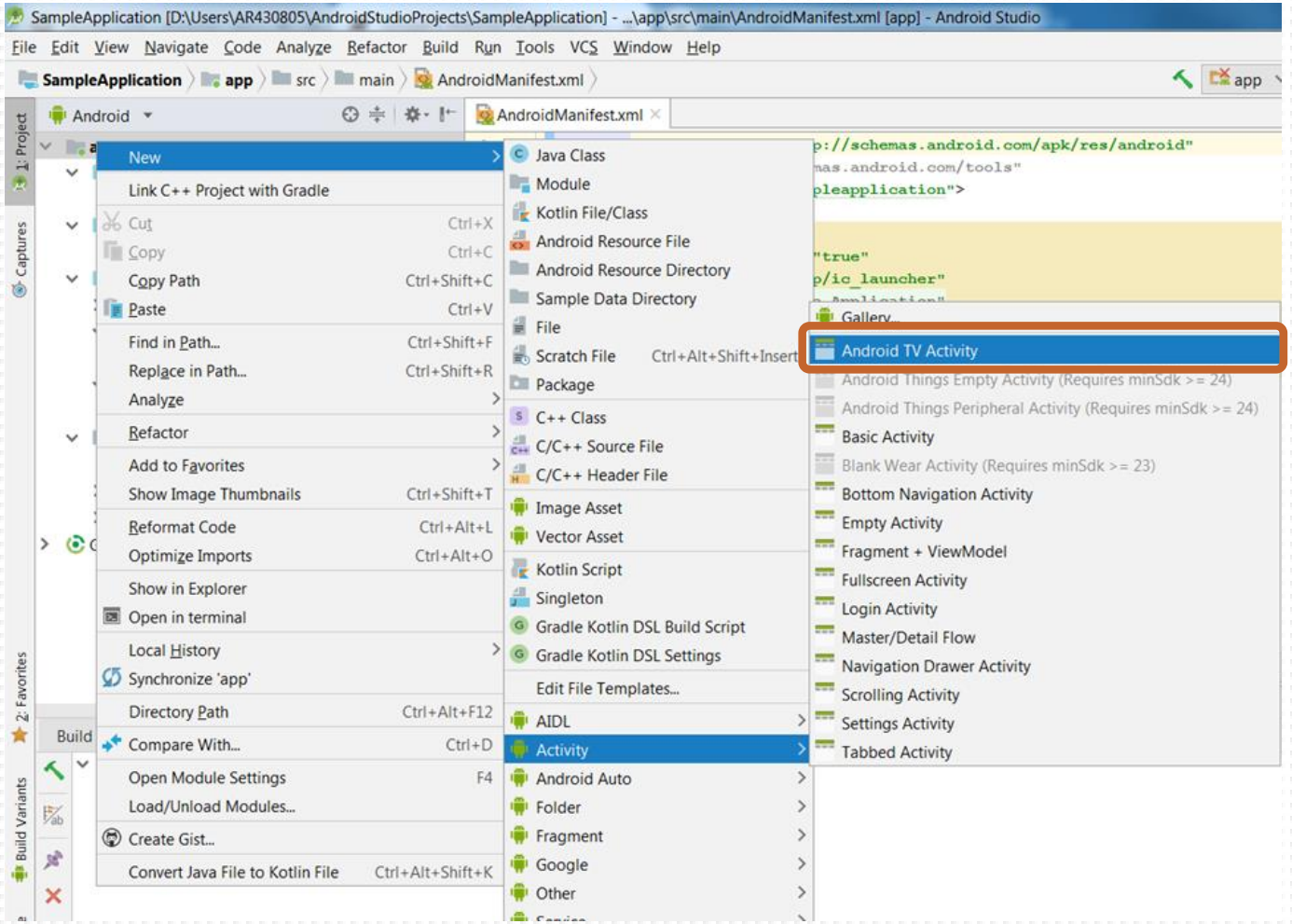
Creating Android TV App IV



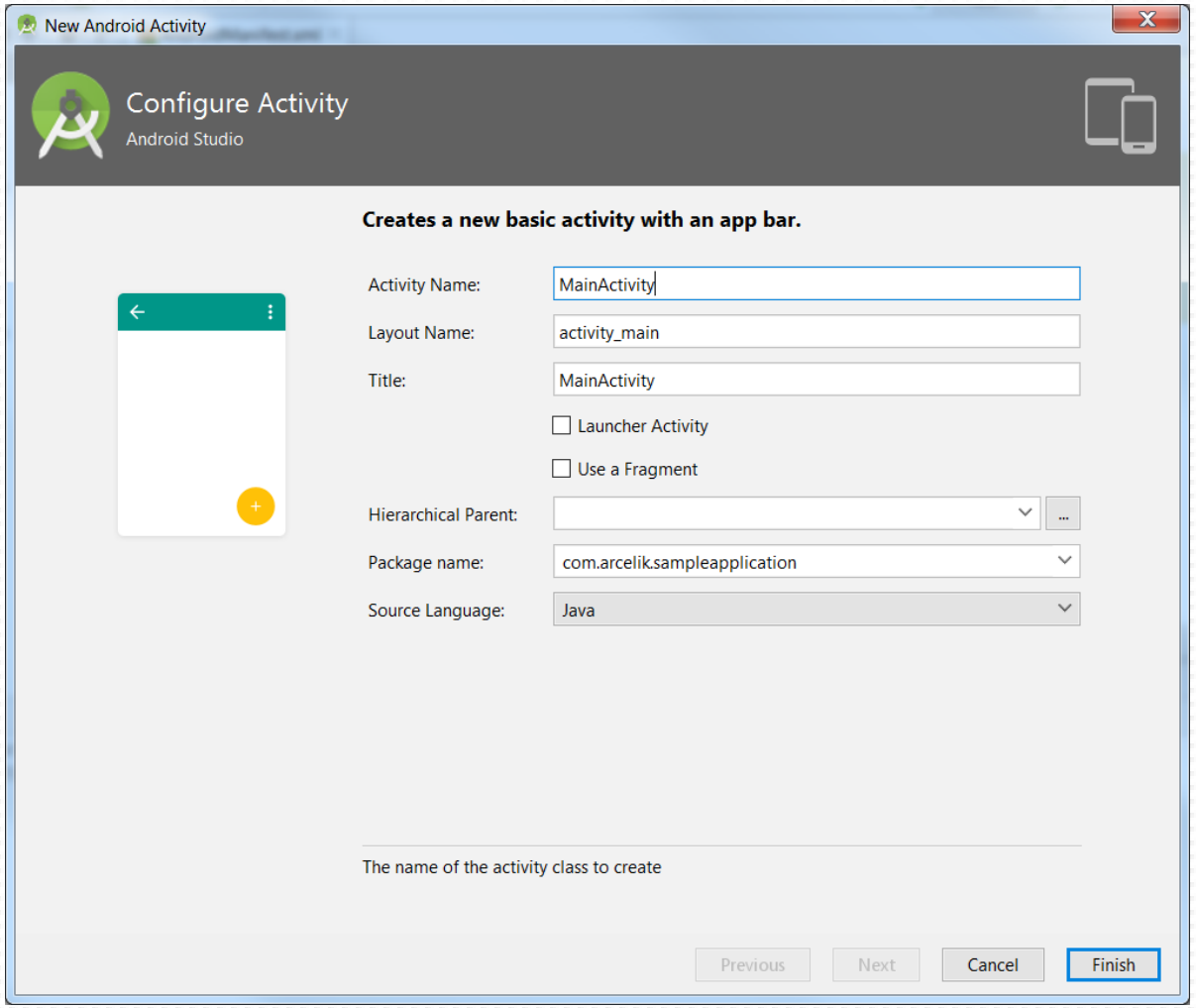
Creating Android TV App V



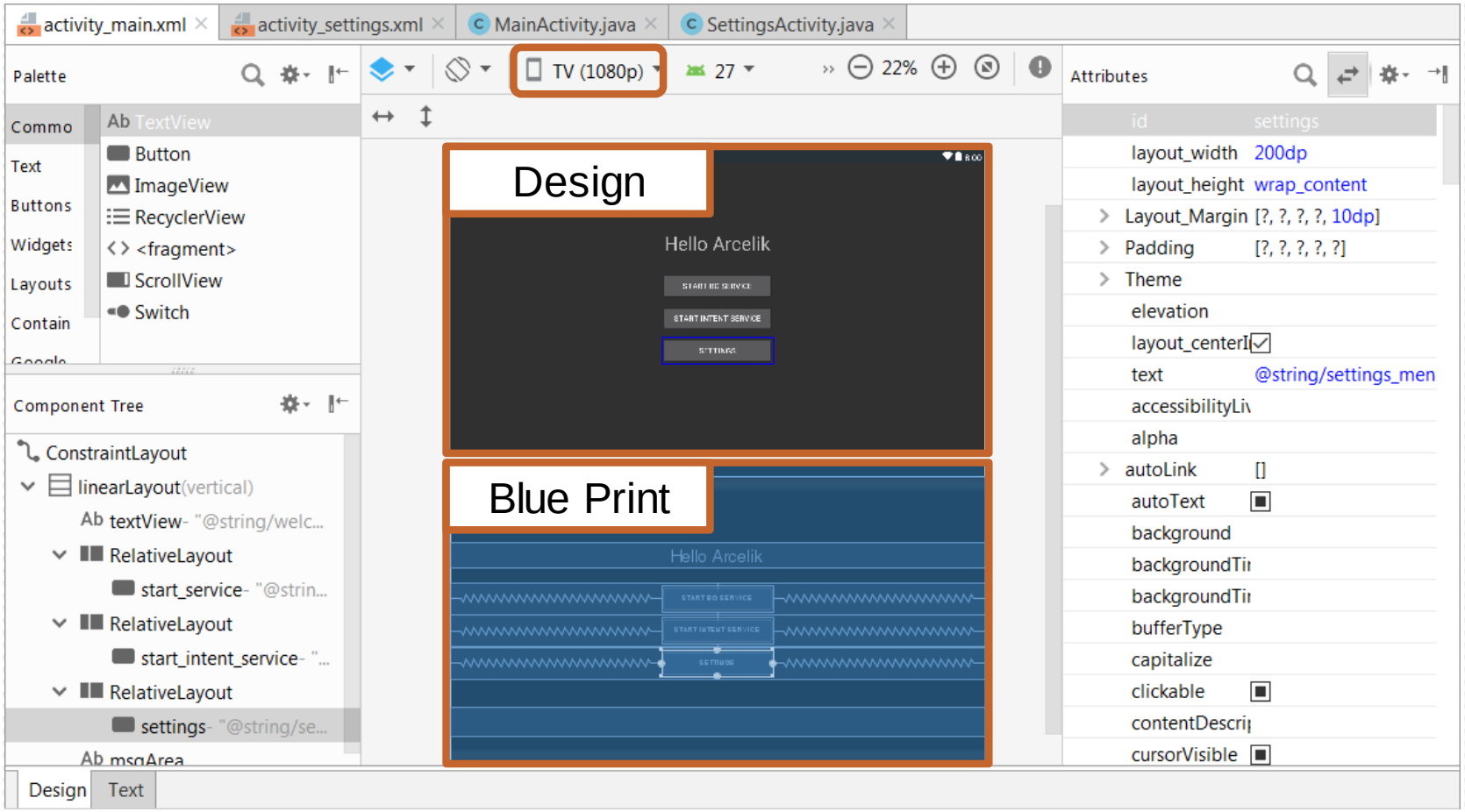
Adding Activity I



Adding Activity II

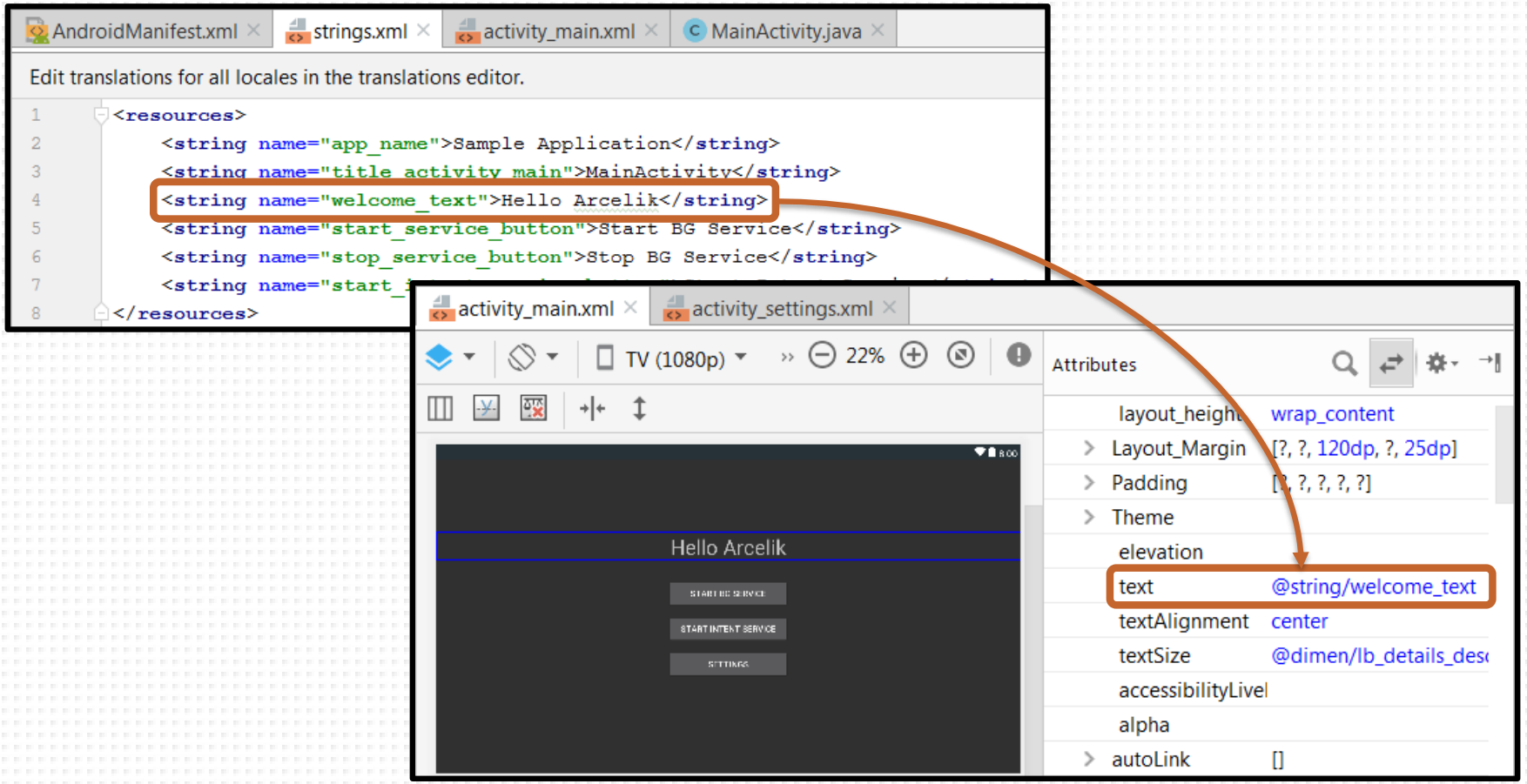


Activity Layout File



Strings.txt file I

- Do not embed texts to layout file!



Strings.txt file II

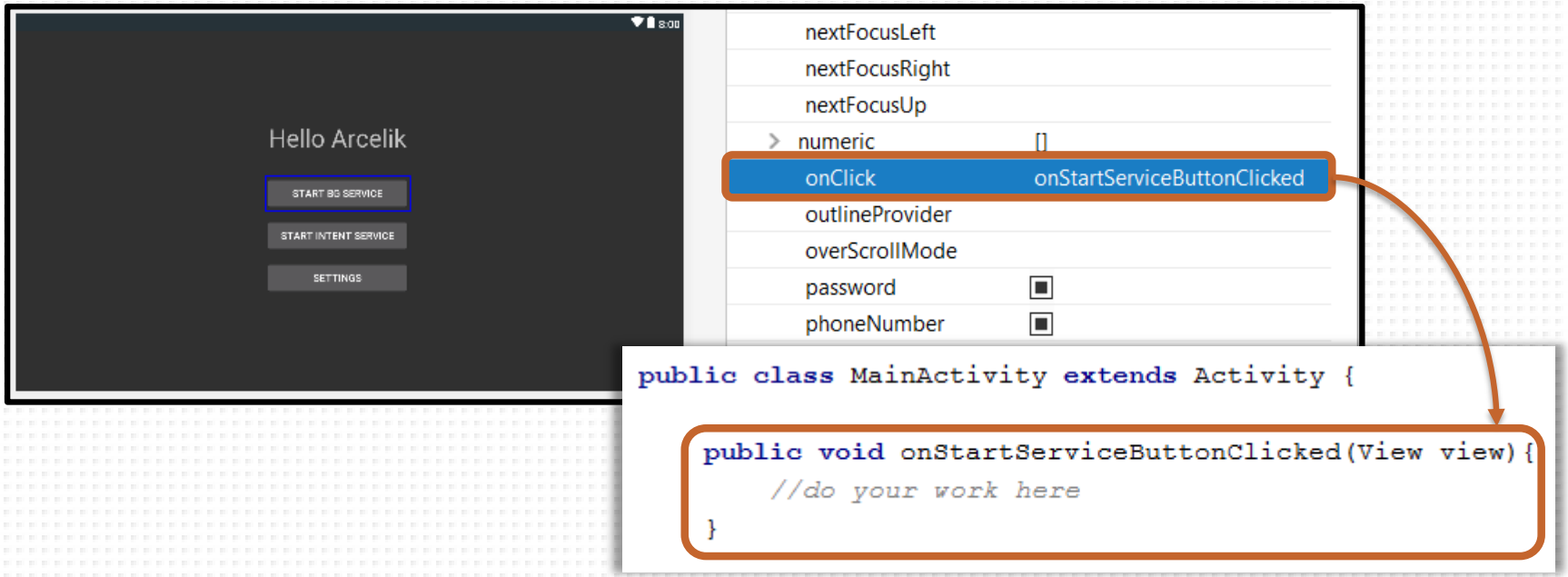
- Do not use static texts in your source code!

```
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);
```

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 a 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());
```

Starting Another (Settings) Activity

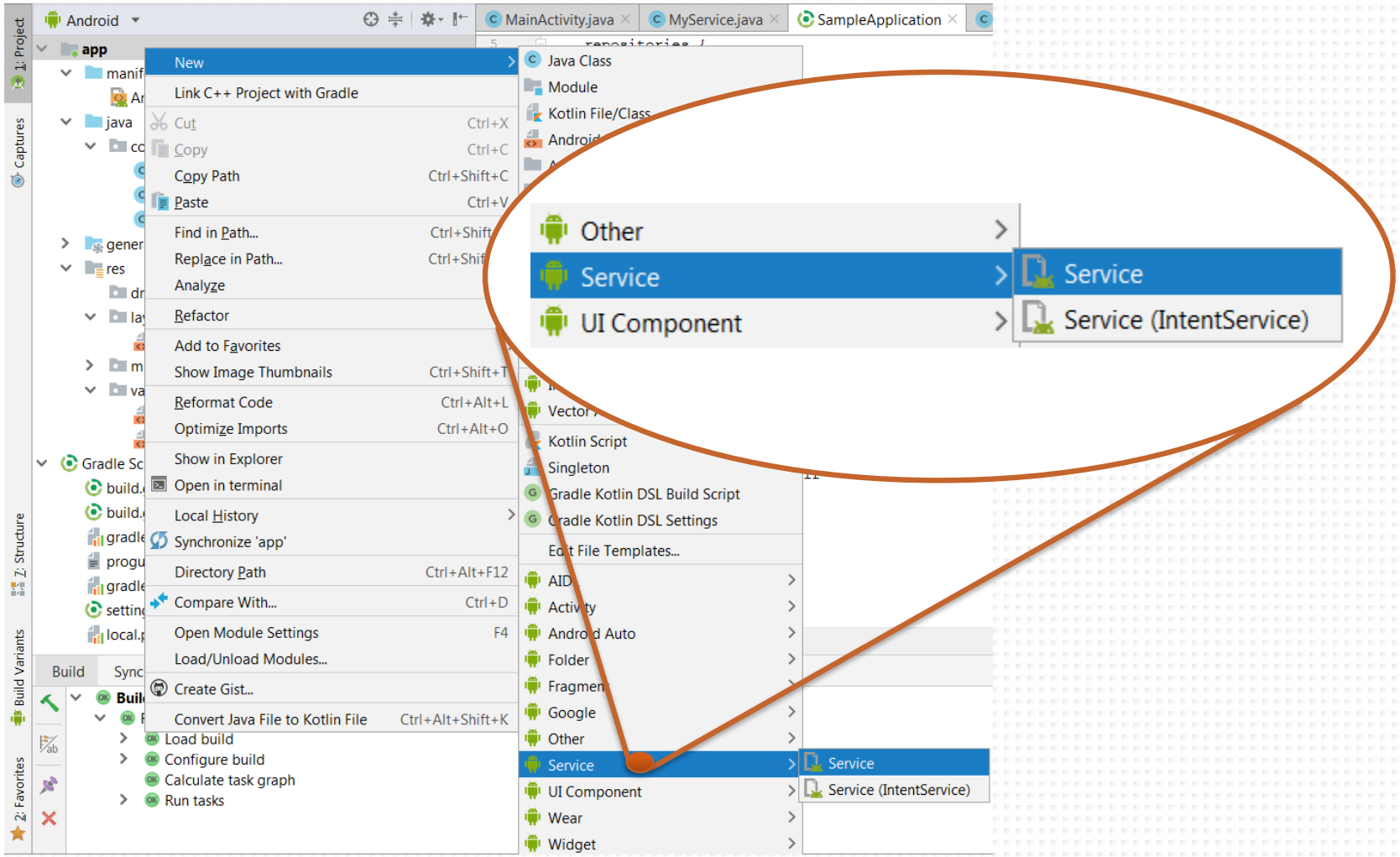
1. Add a button click listener
2. Create an explicit Intent when the button is clicked
3. Start Activity via created Intent

```
// Handle onClickListener of settings button.
Button settings = (Button)findViewById(R.id.settings);
settings.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent( packageContext: MainActivity.this, SettingsActivity.class);
        startActivity(intent)
    }
});
```

Read/Write Application Settings

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

    //get last saved value from preferences
    SharedPreferences sharedPref = getPreferences(Context.MODE_PRIVATE);
    String lastSavedDate = sharedPref.getString( s: "last_saved_date",  s1: "unsaved!");

    //Use last saved date

    // Handle onClickListener of save button.
    Button save = (Button)findViewById(R.id.save);
    save.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String lastSavedDate = getShortDate();
            SharedPreferences sharedPref = getPreferences(Context.MODE_PRIVATE);
            SharedPreferences.Editor editor = sharedPref.edit();
            editor.putString( s: "last_saved_date", lastSavedDate);
            editor.commit();
        }
    });
}
```

Adding Service

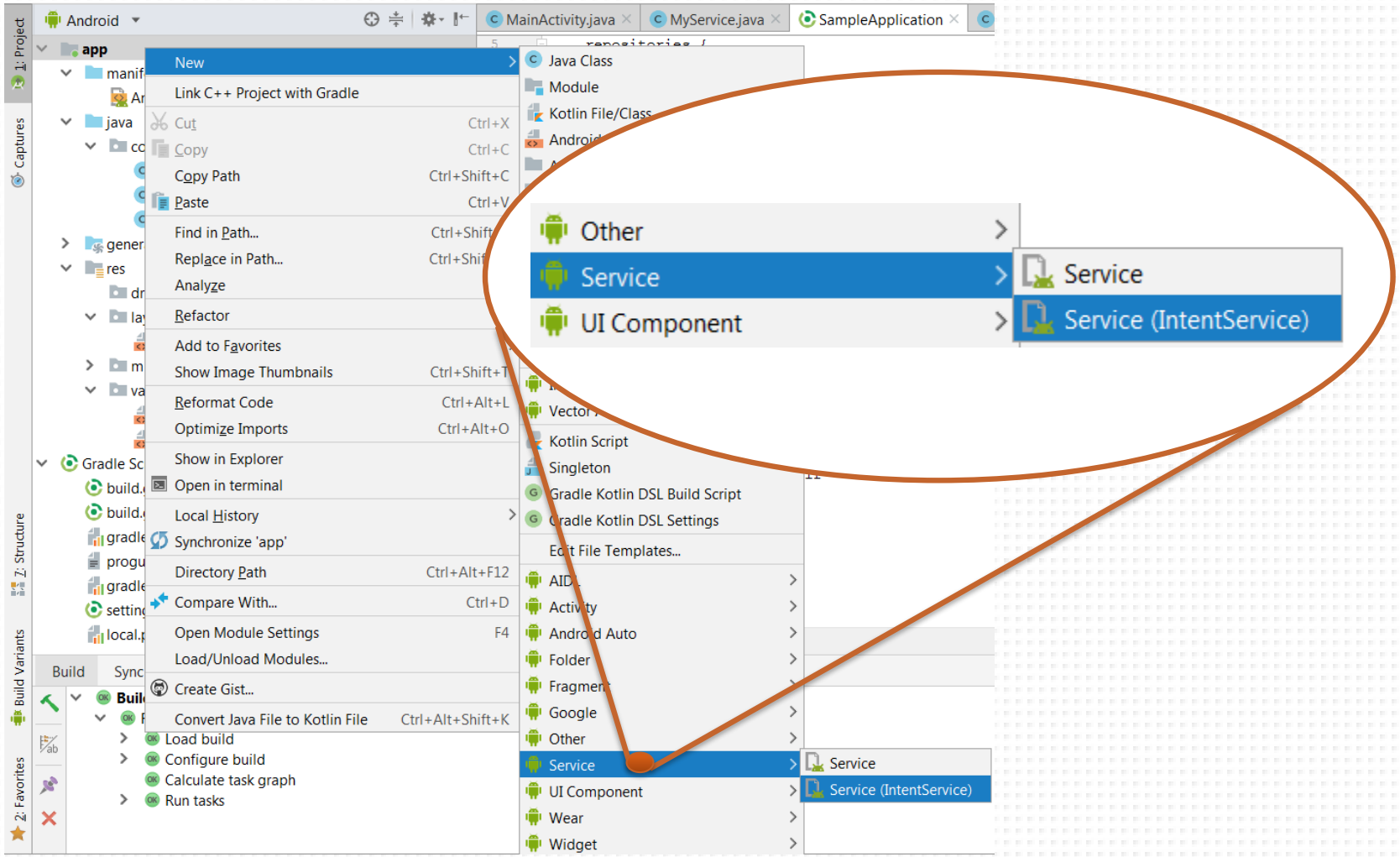


Toggle the Service

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

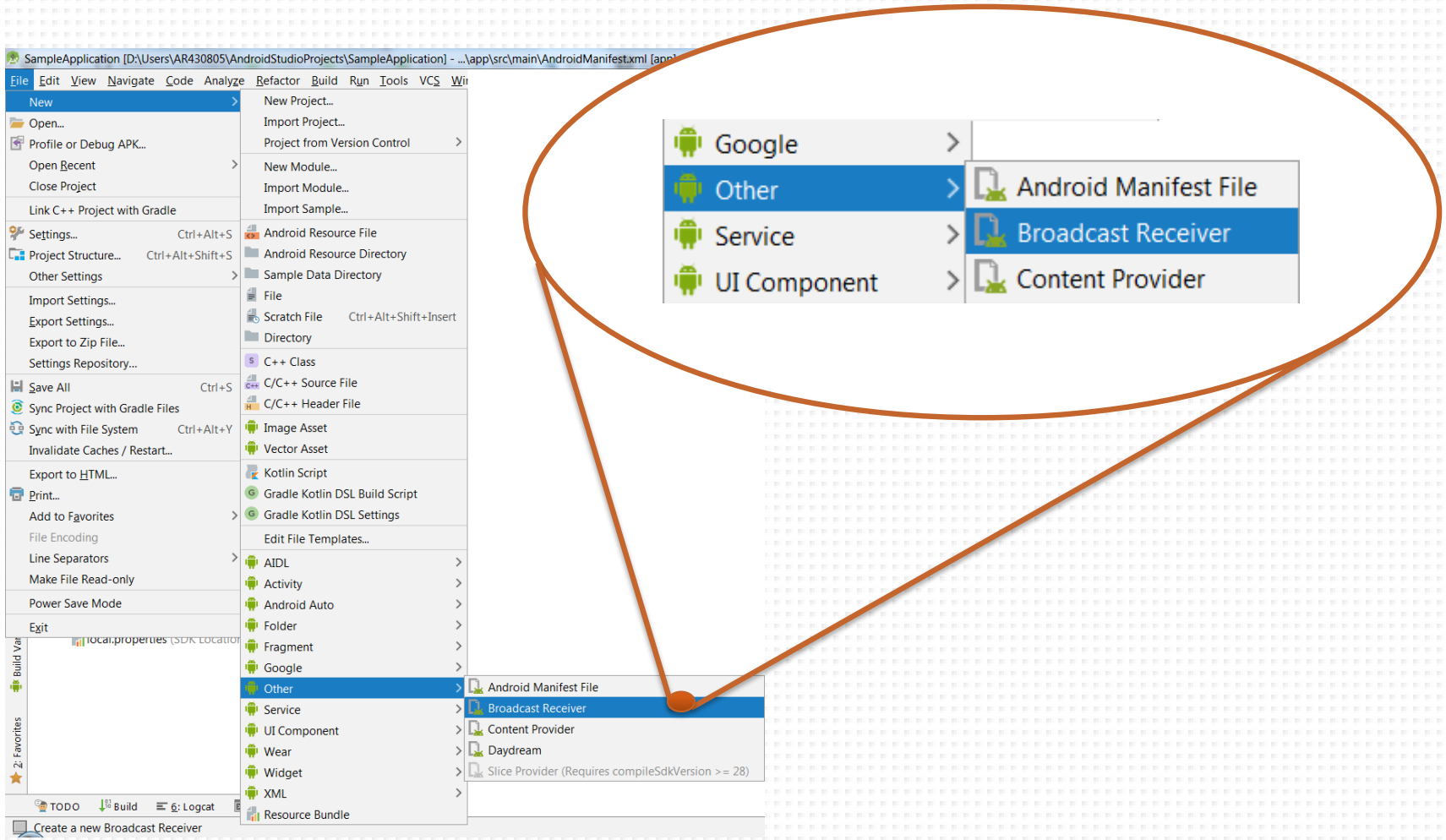


Starting Intent Service

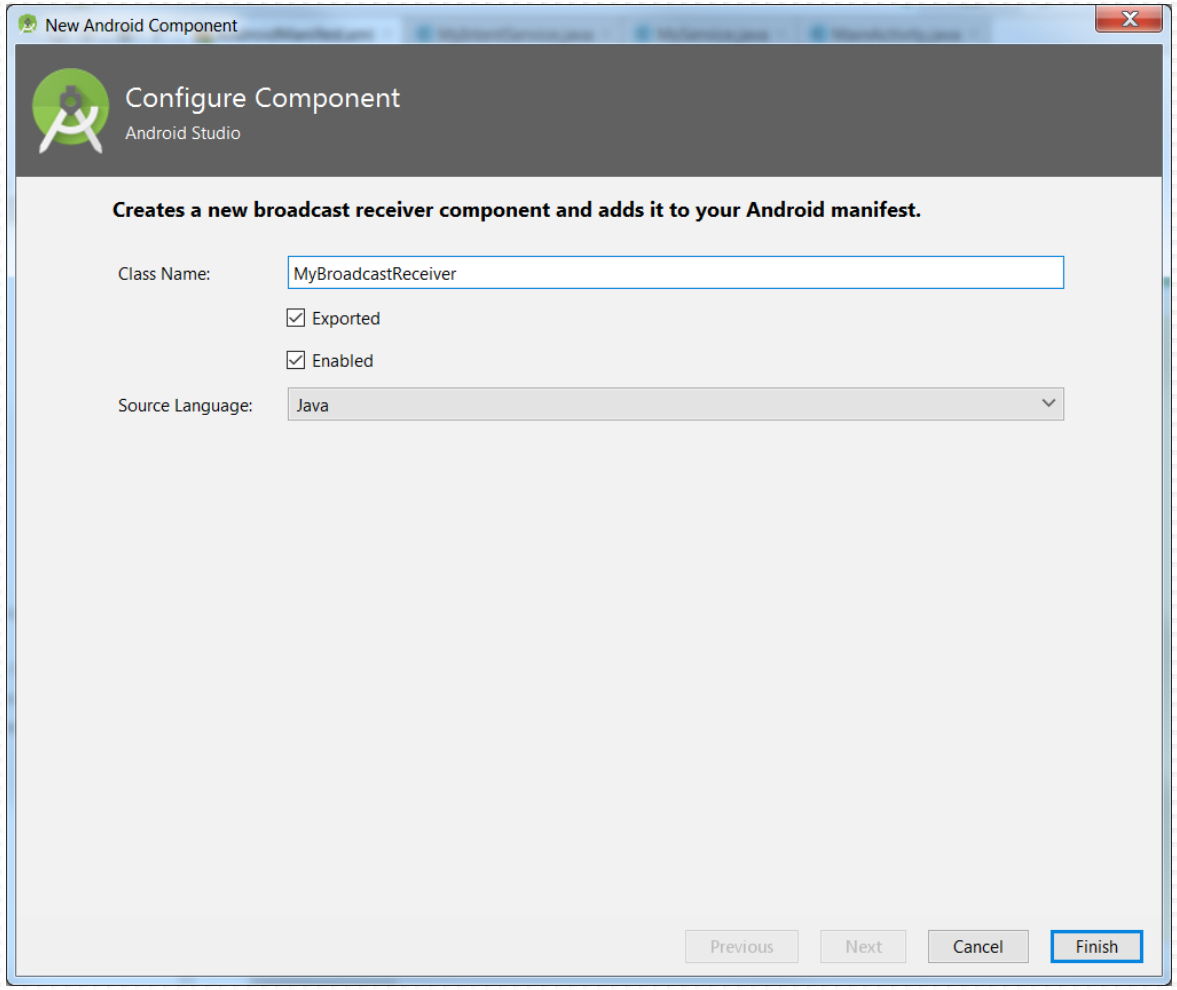
1. Add a button click listener
2. Create an explicit Intent when the button is clicked
3. Set Intent action and extra data
4. Start the Intent Service via created Intent

```
// 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

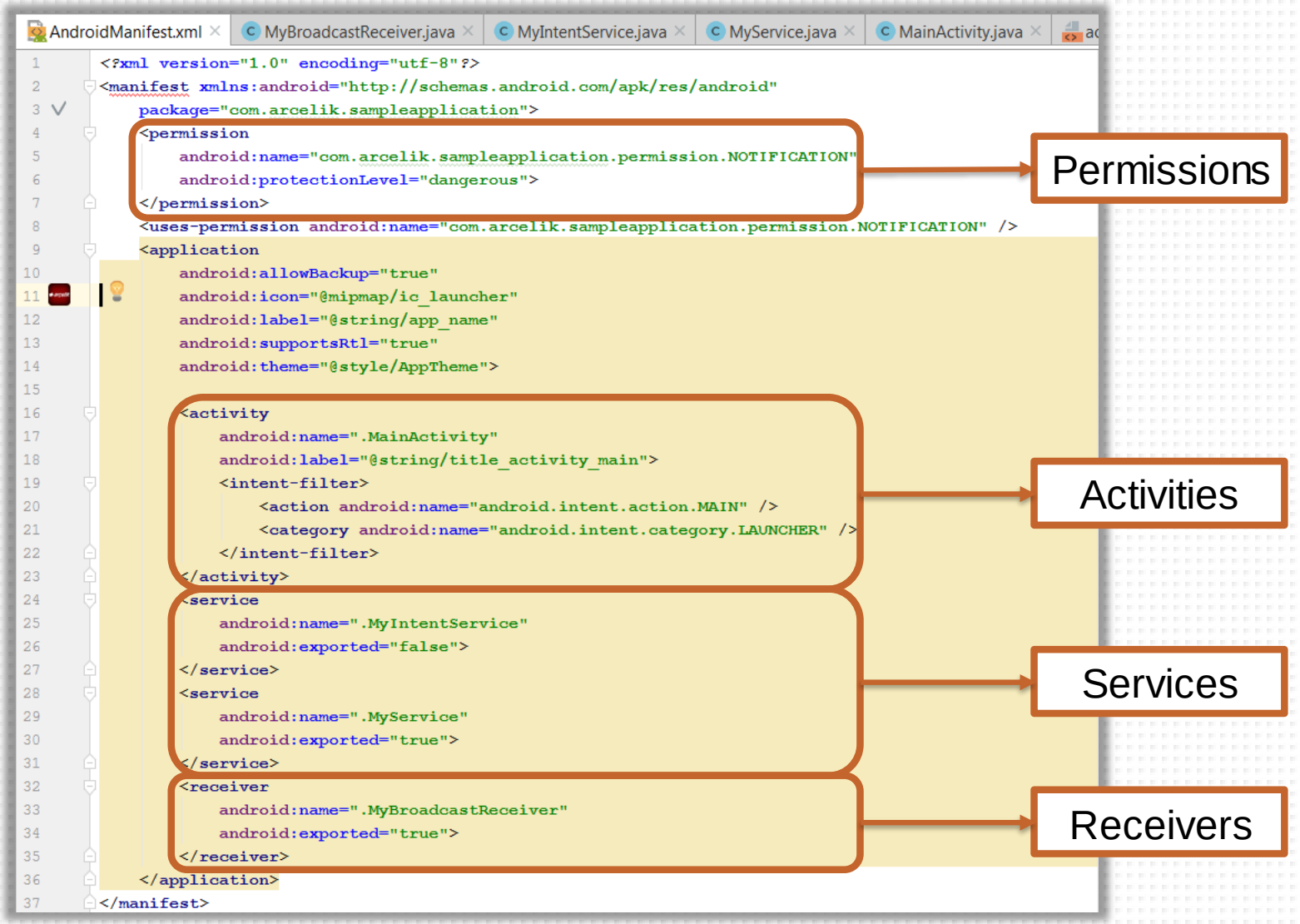
```
//Send broadcast
Log.d(TAG_SERVICE, msg: "Sending message to activity: " + notification);
final Intent intent = new Intent(BROADCAST_INTENT);
intent.putExtra( name: "msg", value: dateAsStr + ":" + notification);
sendBroadcast(intent, BROADCAST_PERMISSION);
```

Receiving Broadcast

1. Create an IntentFilter to catch proper broadcast event
2. Create BroadcastReceiver and override 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



Adjust App for Android TV Platforms

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.arcelik.sampleapplication">

    <uses-feature android:name="android.software.leanback" android:required="true" />
    <uses-feature android:name="android.hardware.touchscreen" android:required="false" />

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

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:banner="@drawable/ic_banner"
        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" />
                <category android:name="android.intent.category.LEANBACK_LAUNCHER" />
            </intent-filter>
        </activity>
```

Make it compatible with TVs

Disable mobile devices

320x180 px banner icon

To launch app from home screen

Gradle Build File

build.gradle (:app)

You can use the Project Structure dialog to view and edit your project configuration

```
1  apply plugin: 'com.android.application'
2
3  android {
4      compileSdkVersion 27
5      defaultConfig {
6          applicationId "com.arcelik.sampleapplication"
7          minSdkVersion 21
8          targetSdkVersion 27
9          versionCode 1
10         versionName "1.0"
11     }
12     buildTypes {
13         release {
14             minifyEnabled false
15             proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
16         }
17     }
18 }
19
20 dependencies {
21     implementation fileTree(dir: 'libs', include: ['*.jar'])
22     implementation 'com.android.support:leanback-v17:27.1.1'
23     implementation 'com.android.support.constraint:constraint-layout:1.1.3'
24 }
```

App Config

Build Types

Local Libs

Remote Libs

Configuring Build Types

```
android {
    defaultConfig {
        manifestPlaceholders = [hostName:"www.example.com"]
        ...
    }
    buildTypes {
        release {
            minifyEnabled true
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }

        debug {
            applicationIdSuffix ".debug"
            debuggable true
        }

        /**
         * The `initWith` property allows you to copy configurations from other build types,
         * then configure only the settings you want to change. This one copies the debug build
         * type, and then changes the manifest placeholder and application ID.
         */
        staging {
            initWith debug
            manifestPlaceholders = [hostName:"internal.example.com"]
            applicationIdSuffix ".debugStaging"
        }
    }
```

Configuring Build Variants

- Build variants are the result of Gradle using a specific set of rules to combine settings, code, and resources configured in your build types and product flavors.
- You can create and configure build types in the module-level **build.gradle** file inside the android block.
- When you make changes to a build configuration file, Android Studio requires that you sync your project with the new configuration.
- You can change the build variant to whichever one you want to build and run, just go to **Build > Select Build Variant** and select one from the drop-down menu.

Configuring Product Flavors

```
android {
    ...
    defaultConfig {...}
    buildTypes {
        debug{...}
        release{...}
    }
    // Specifies one flavor dimension.
    flavorDimensions "version"
    productFlavors {
        demo {
            // Assigns this product flavor to the "version" flavor dimension.
            // If you are using only one dimension, this property is optional,
            // and the plugin automatically assigns all the module's flavors to
            // that dimension.
            dimension "version"
            applicationIdSuffix ".demo"
            versionNameSuffix "-demo"
        }
        full {
            dimension "version"
            applicationIdSuffix ".full"
            versionNameSuffix "-full"
        }
    }
}
```

build.gradle File from Real Application

app

```
1  apply plugin: 'com.android.application'
2
3  android {
4      compileSdkVersion 27
5      defaultConfig {
6          applicationId "com.arcelik.html5testapp"
7          minSdkVersion 22
8          targetSdkVersion 27
9          versionCode 1
10         versionName "1.0"
11         //common app settings
12         resValue "bool", "usesCeHtml", "false"
13         resValue "bool", "overrideKeyCodes", "true"
14         resValue "bool", "overrideUserAgent", "true"
15         resValue "bool", "enableCors", "false"
16         resValue "bool", "disableBack", "true"
17         resValue "integer", "timeout", "30000"
18     }
19     buildTypes {
20         release {
21             minifyEnabled false
22             proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
23         }
24     }
25
26     // Specifies one flavor dimension.
27     flavorDimensions "html5_app"
```

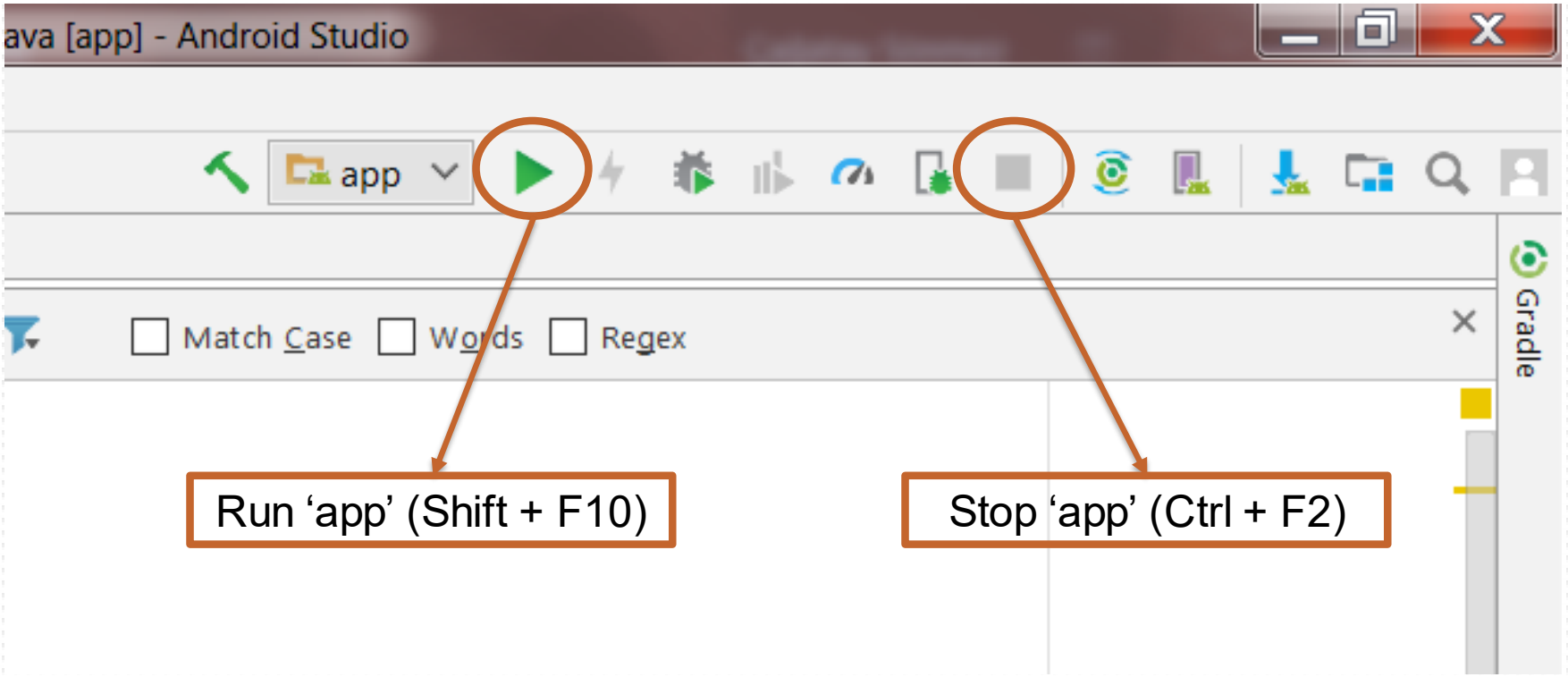
build.gradle File from Real Application cont.

app

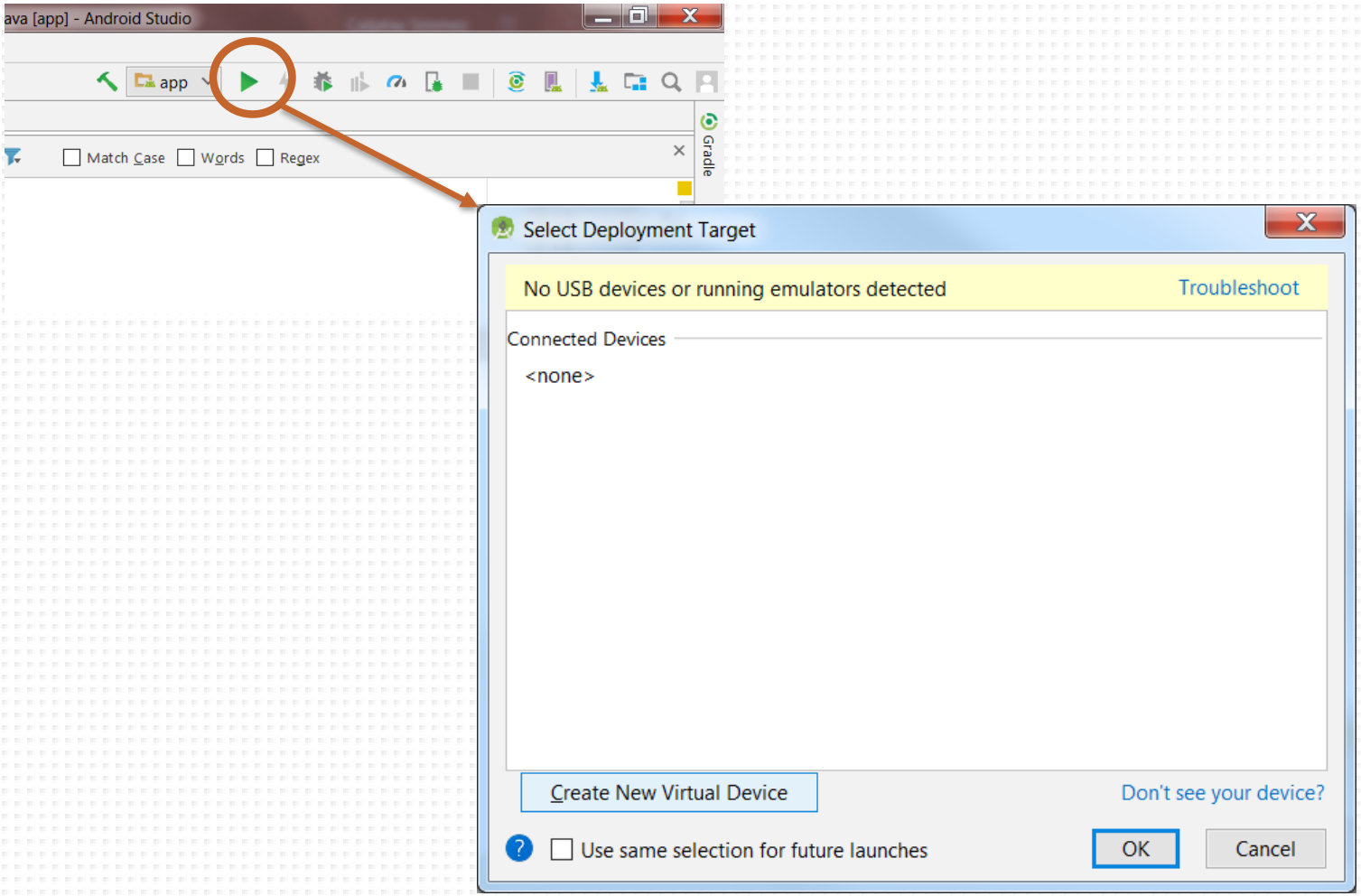
```
26      // Specifies one flavor dimension.
27      flavorDimensions "html5_app"
28      productFlavors {
29          demo {
30              dimension "html5_app"
31              applicationId "com.example.a10433435.iorl28"
32              resValue "string", "app_name", "demo"
33              manifestPlaceholders = [
34                      appIcon: "@drawable/demo"
35              ]
36              resValue "string", "app_url", "file:///android_asset/portal/portal.html"
37              resValue "bool", "usesArSmartTV", "true"
38          }
39          trttv {
40              dimension "html5_app"
41              applicationId "com.arcelik.trttv"
42              resValue "string", "app_name", "TRT TV"
43              manifestPlaceholders = [
44                      appIcon: "@drawable/trt"
45              ]
46              resValue "string", "app_url", "http://smapp.trt.tv/TvApp/arcelik"
47              resValue "bool", "usesArSmartTV", "false"
48          }
49          puhutv {
50              dimension "html5_app"
51              applicationId "com.arcelik.puhutv"
52              resValue "string", "app_name", "puhutv"
```

Running & Debugging Android Application

Running an Android Application I

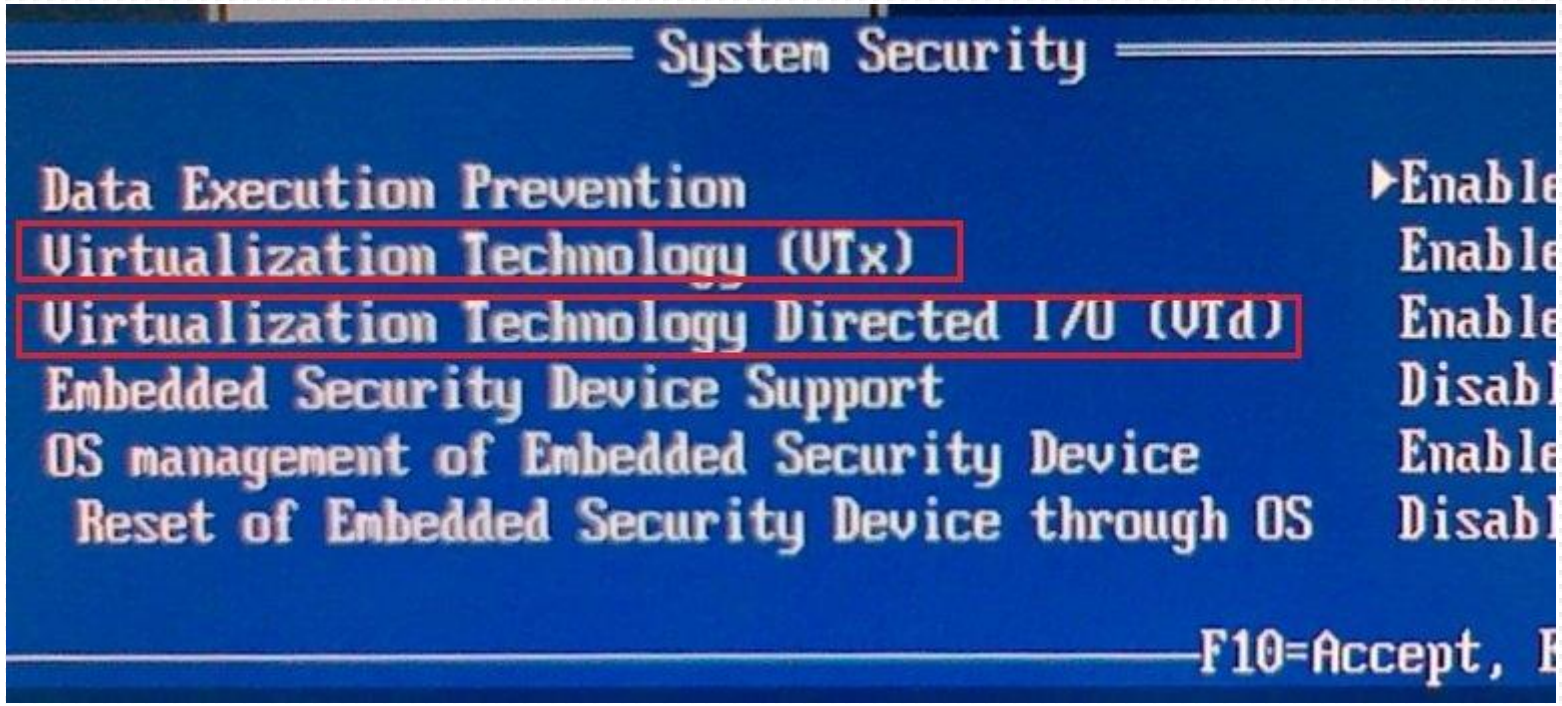


Running an Android Application II



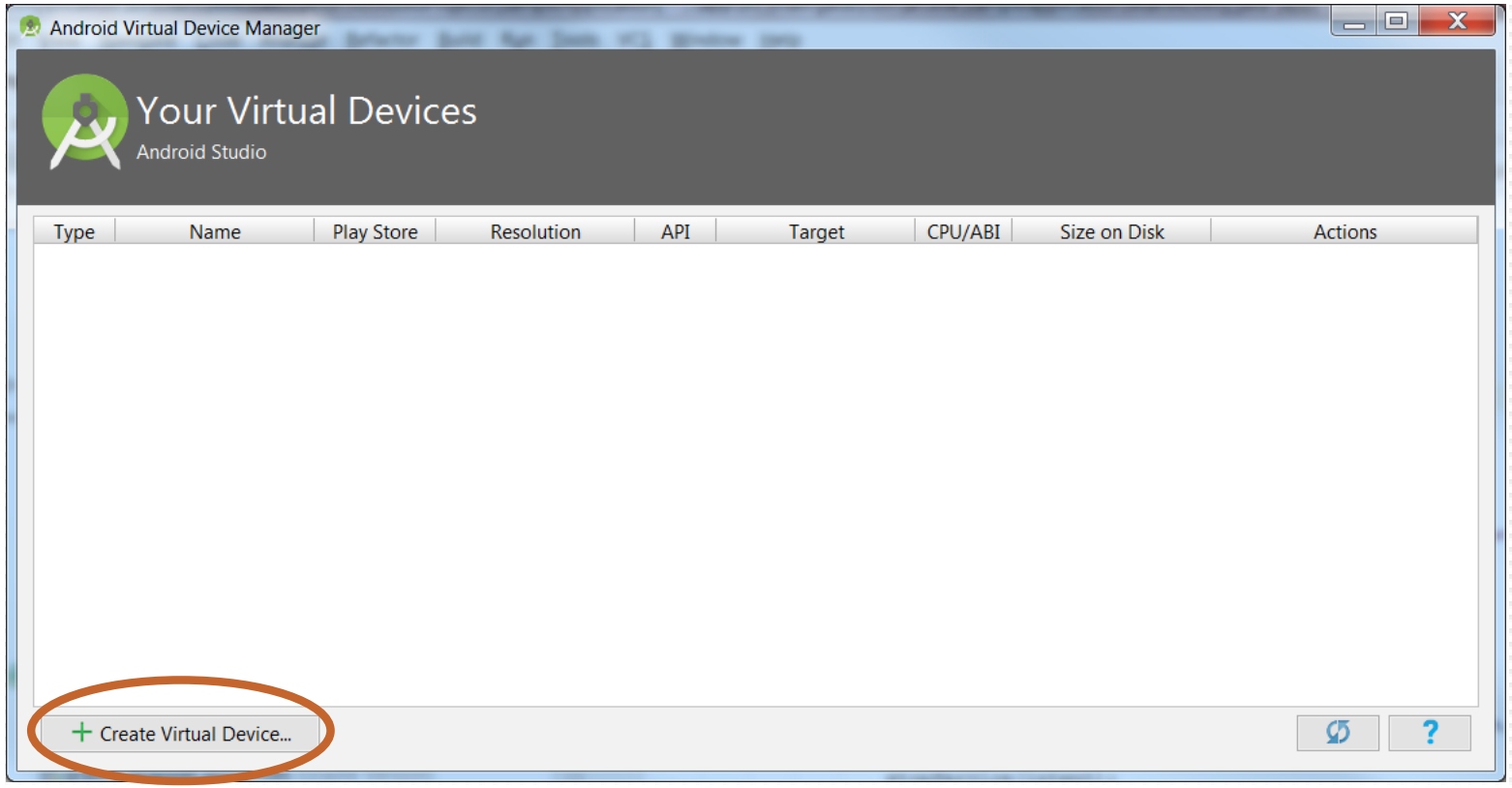
Running App on Emulator

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

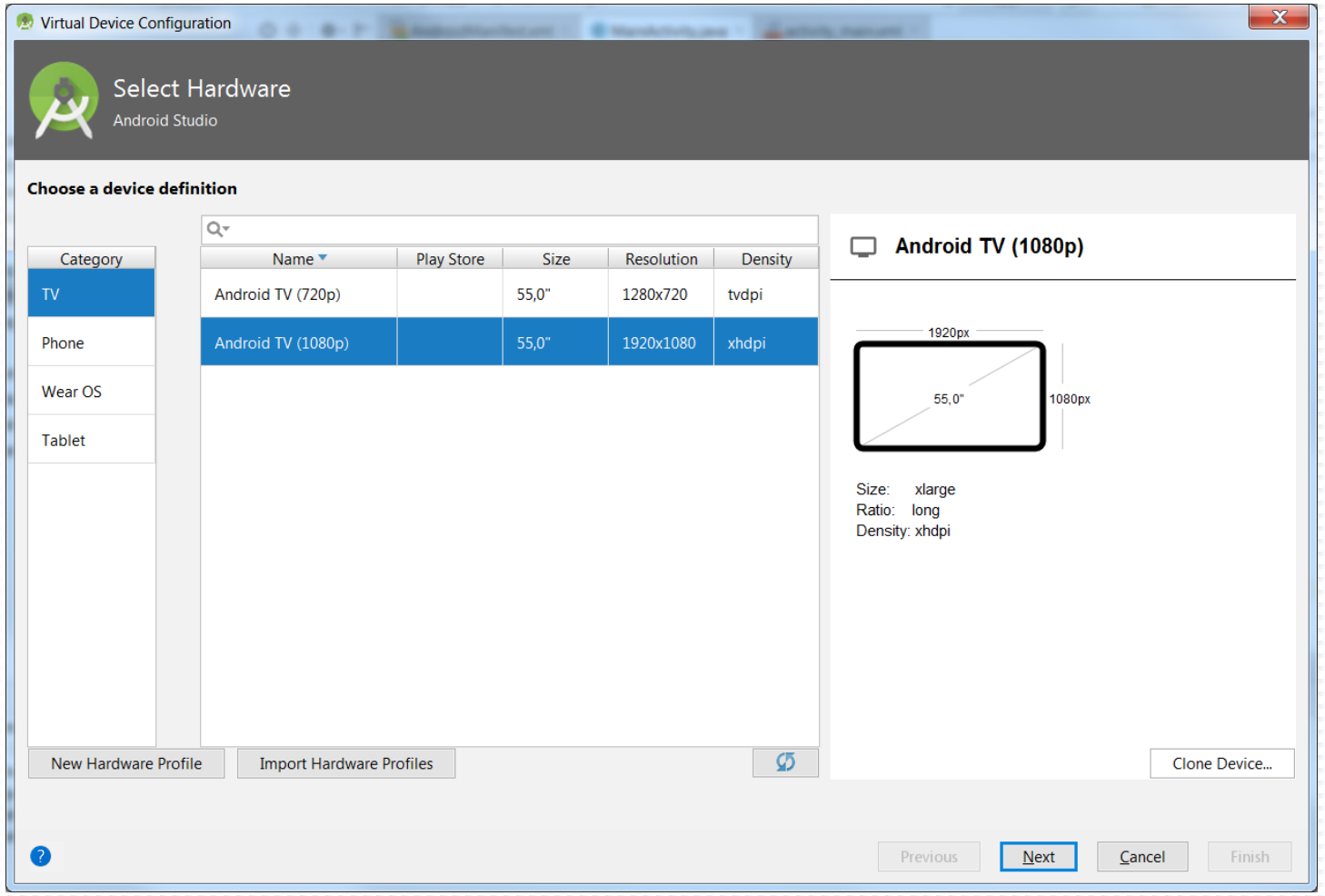


Creating Virtual Device I

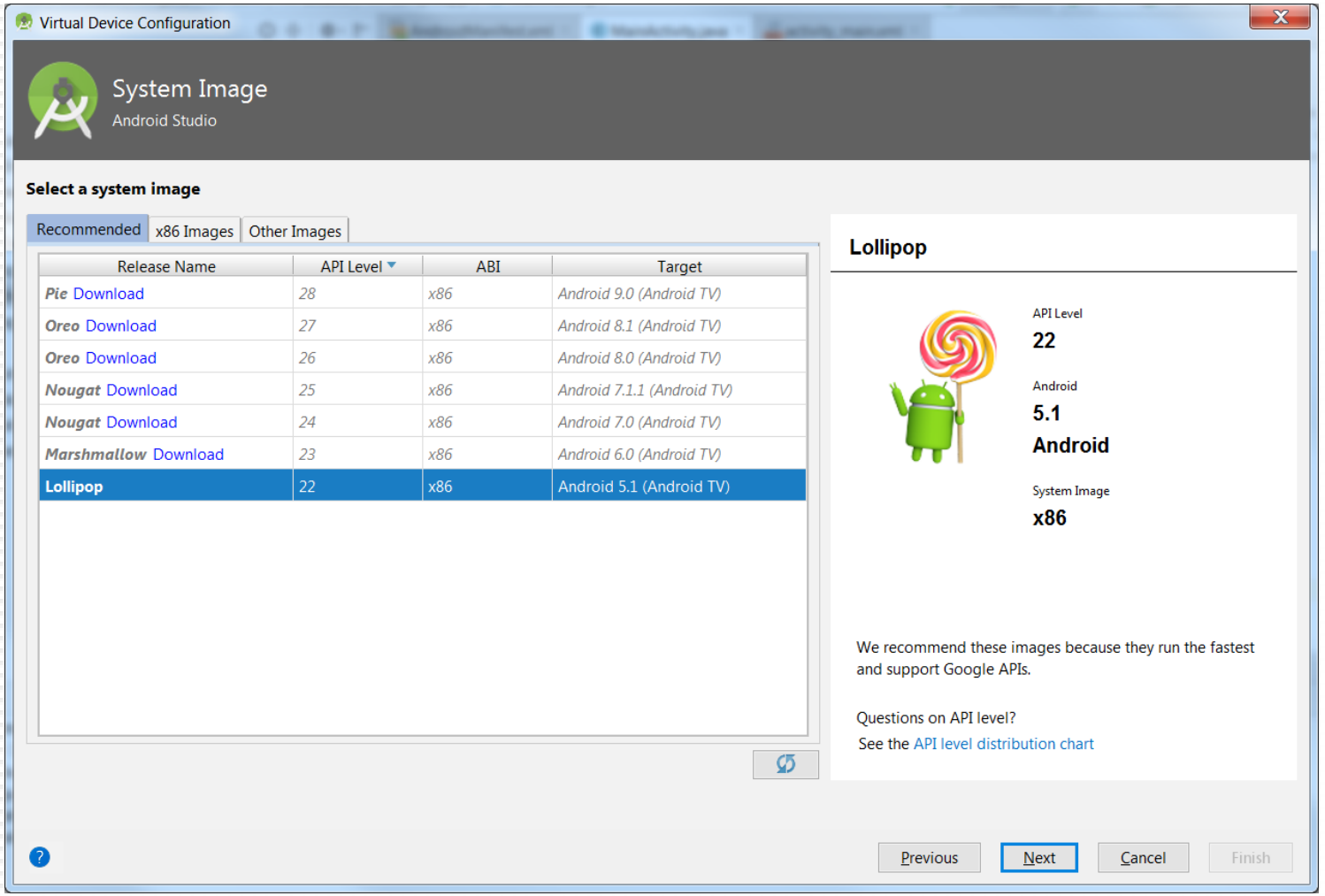
Creating Virtual Device II



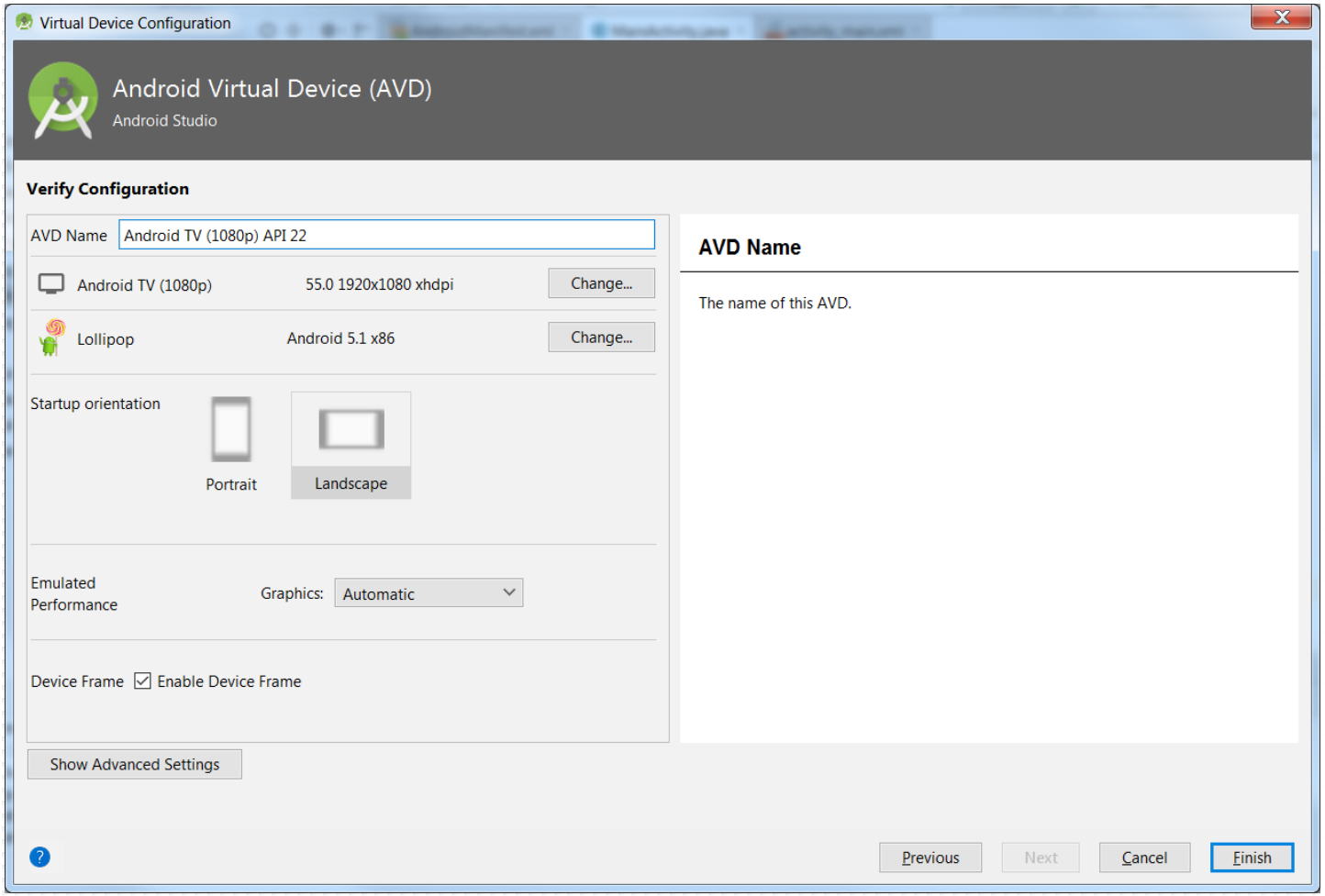
Creating Virtual Device III



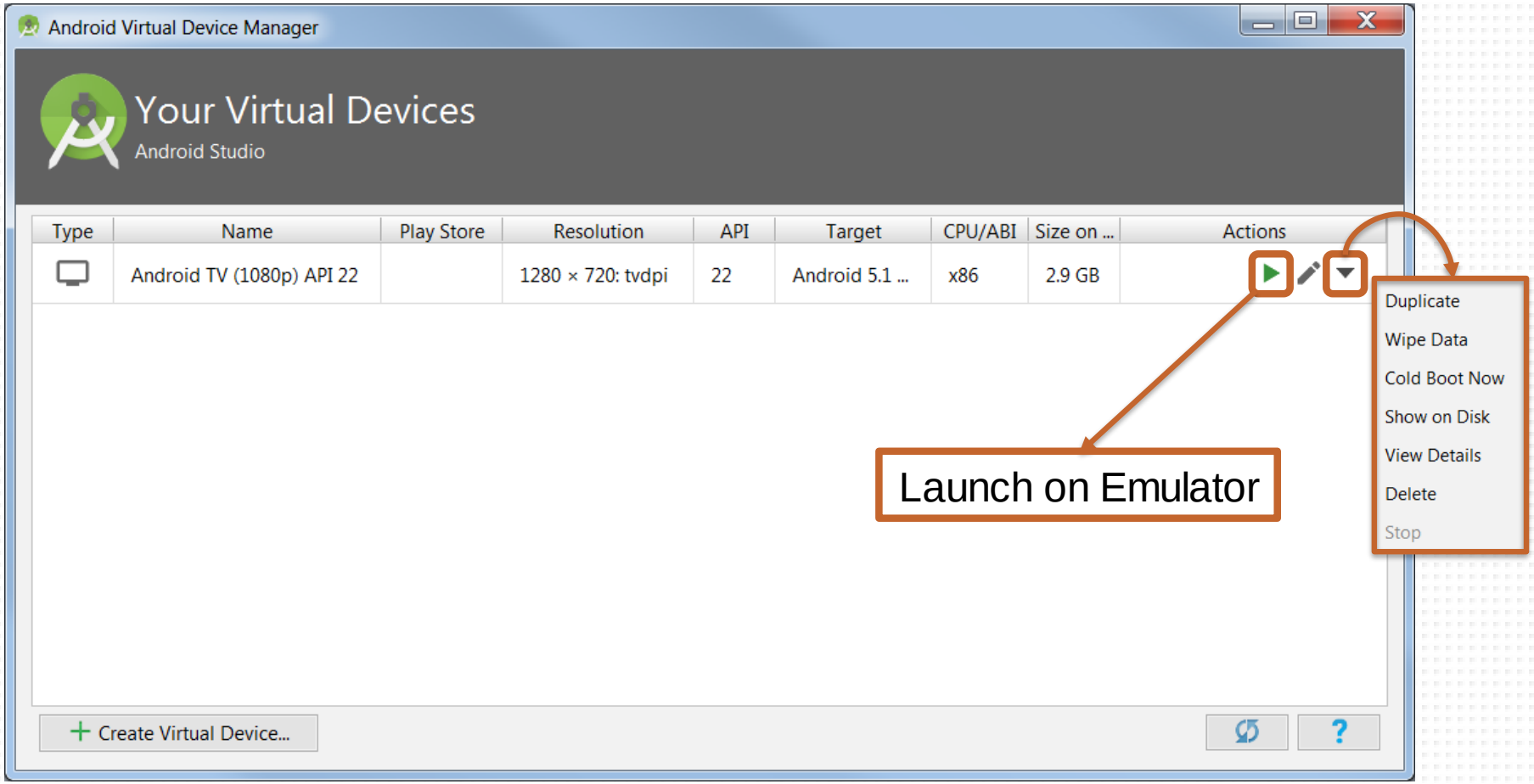
Creating Virtual Device IV



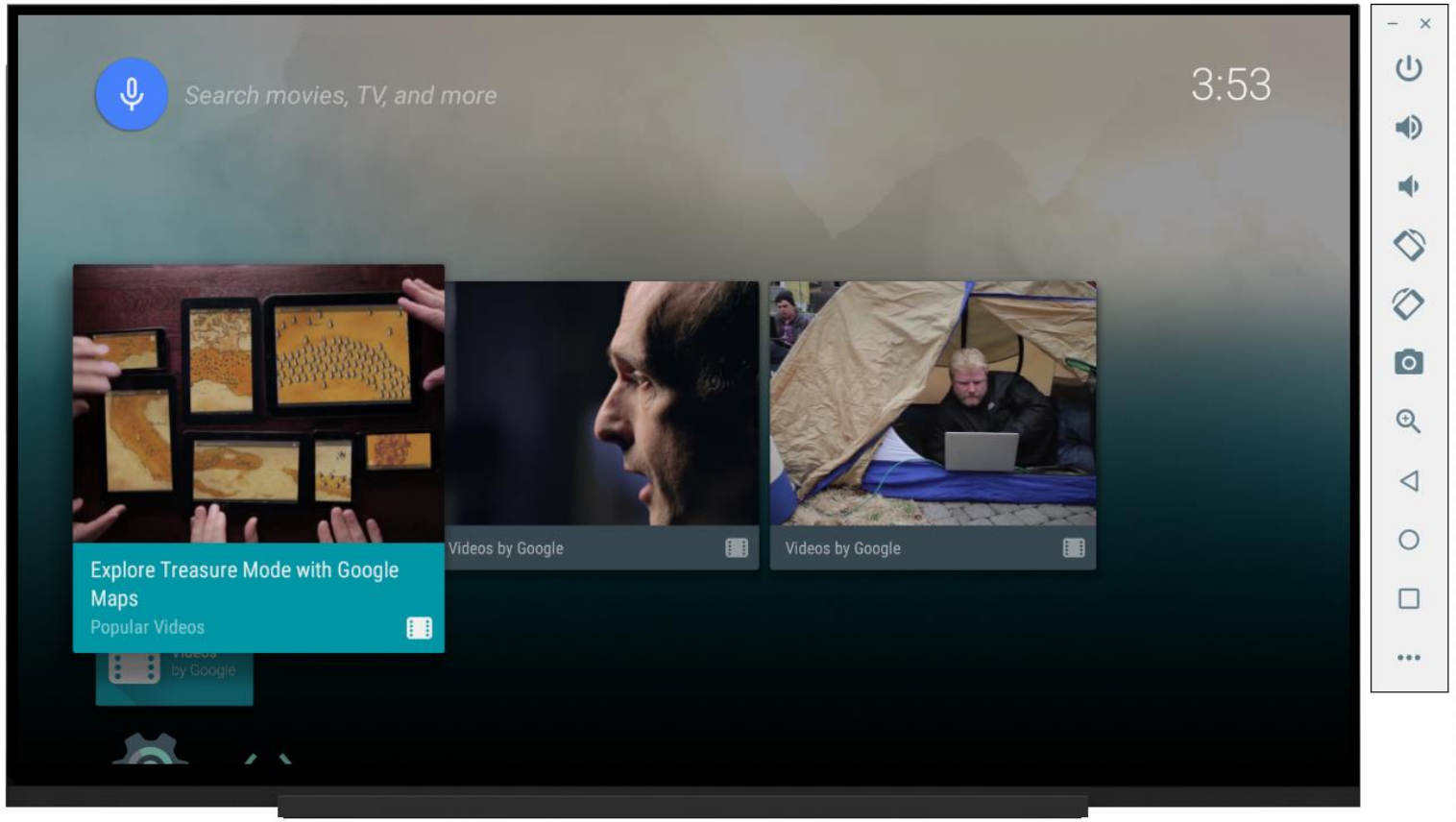
Creating Virtual Device V



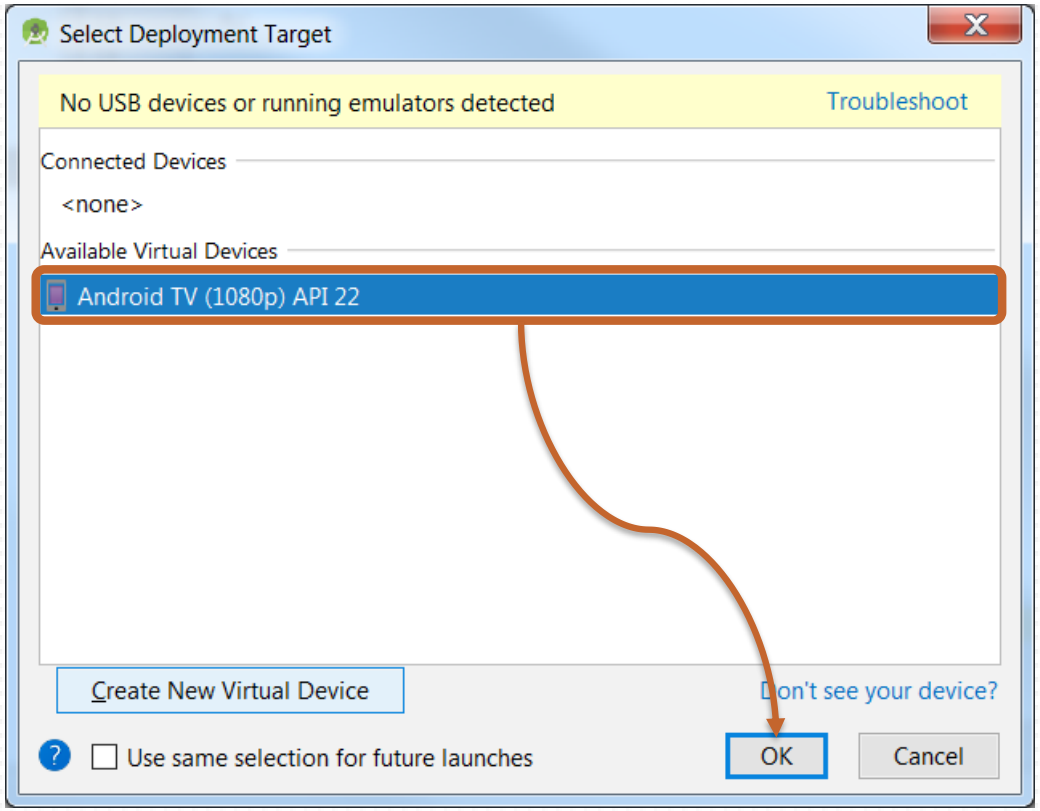
Launching Virtual Device on Emulator I



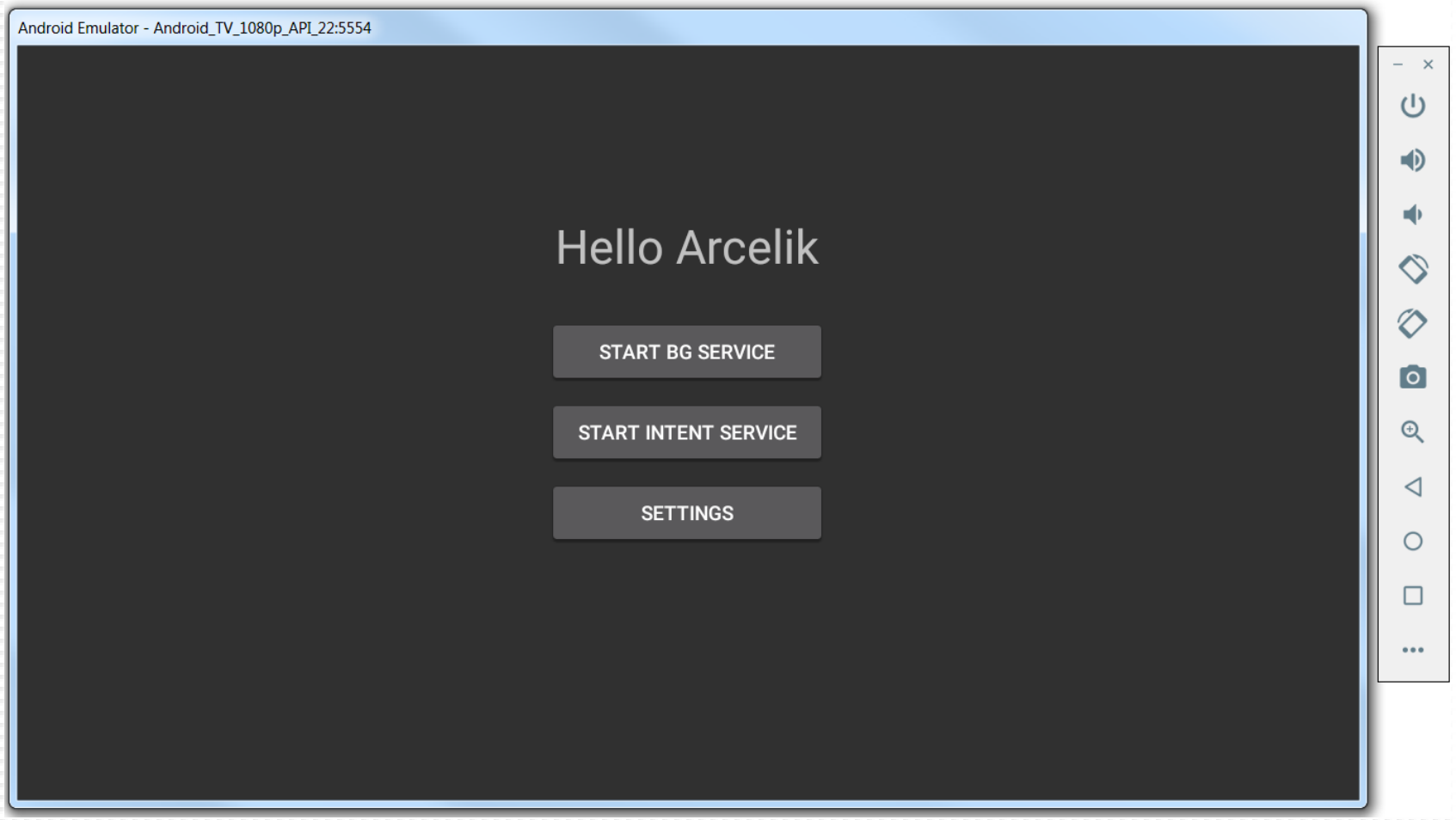
Launching Virtual Device on Emulator II



Running App on Emulator I



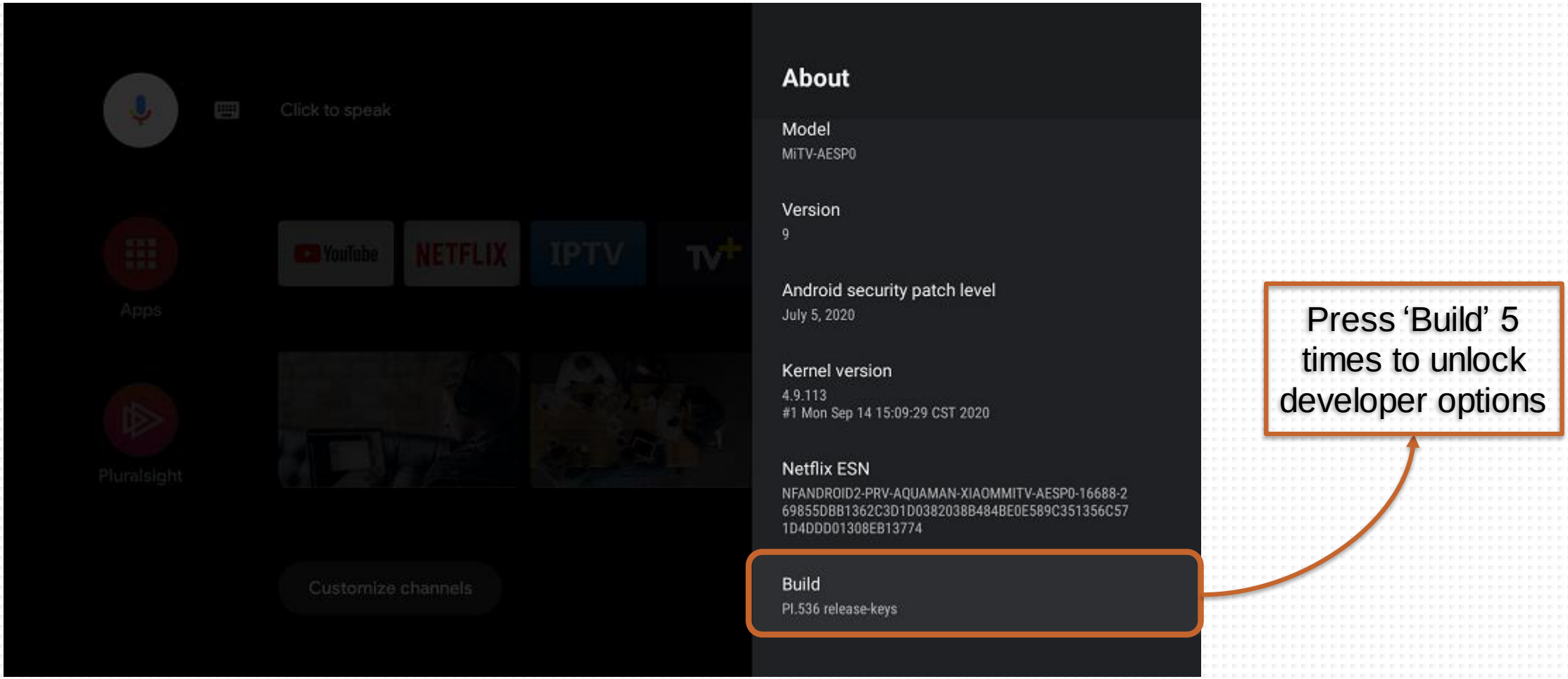
Running App on Emulator II



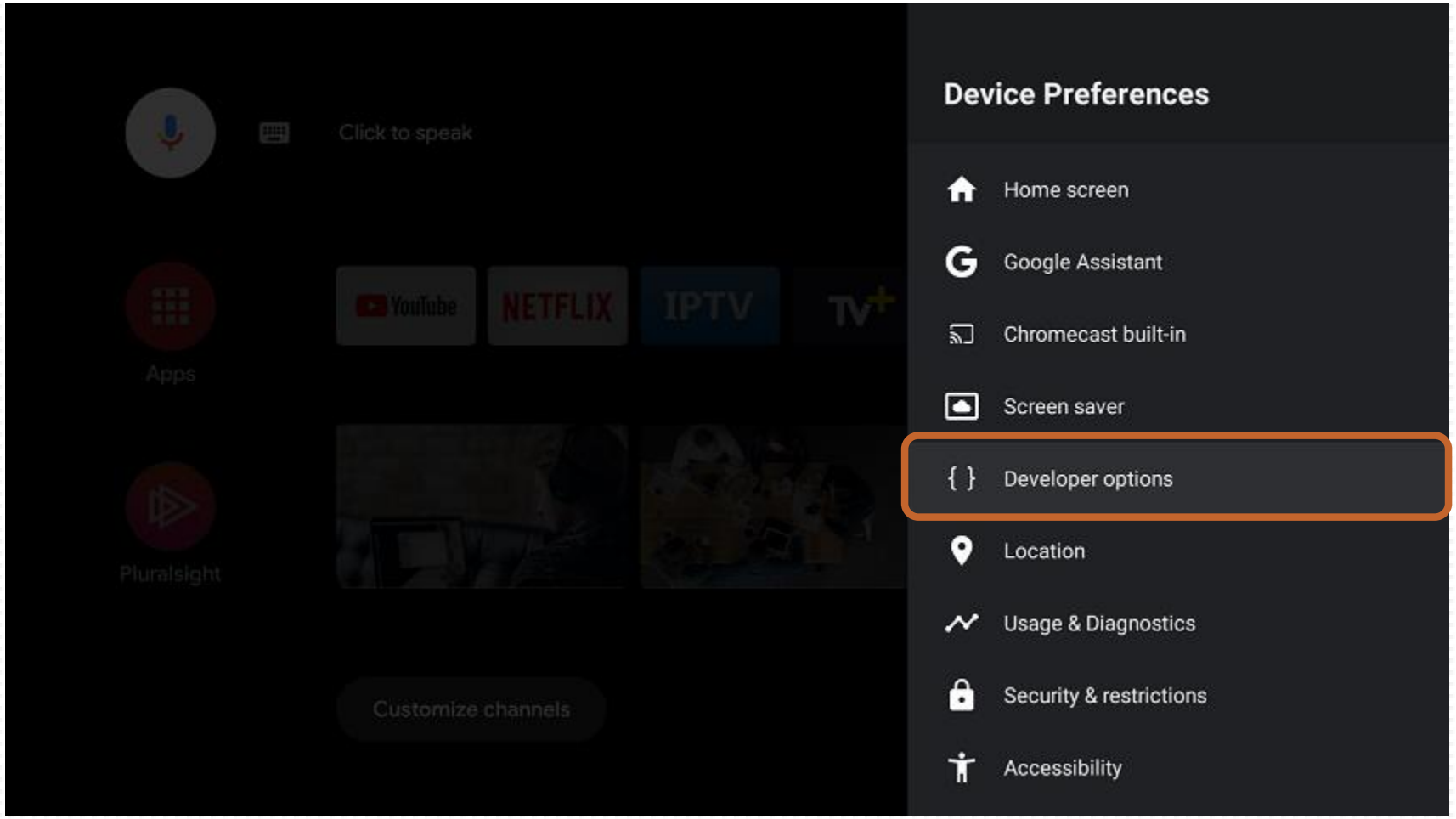
Running App on Device

- 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

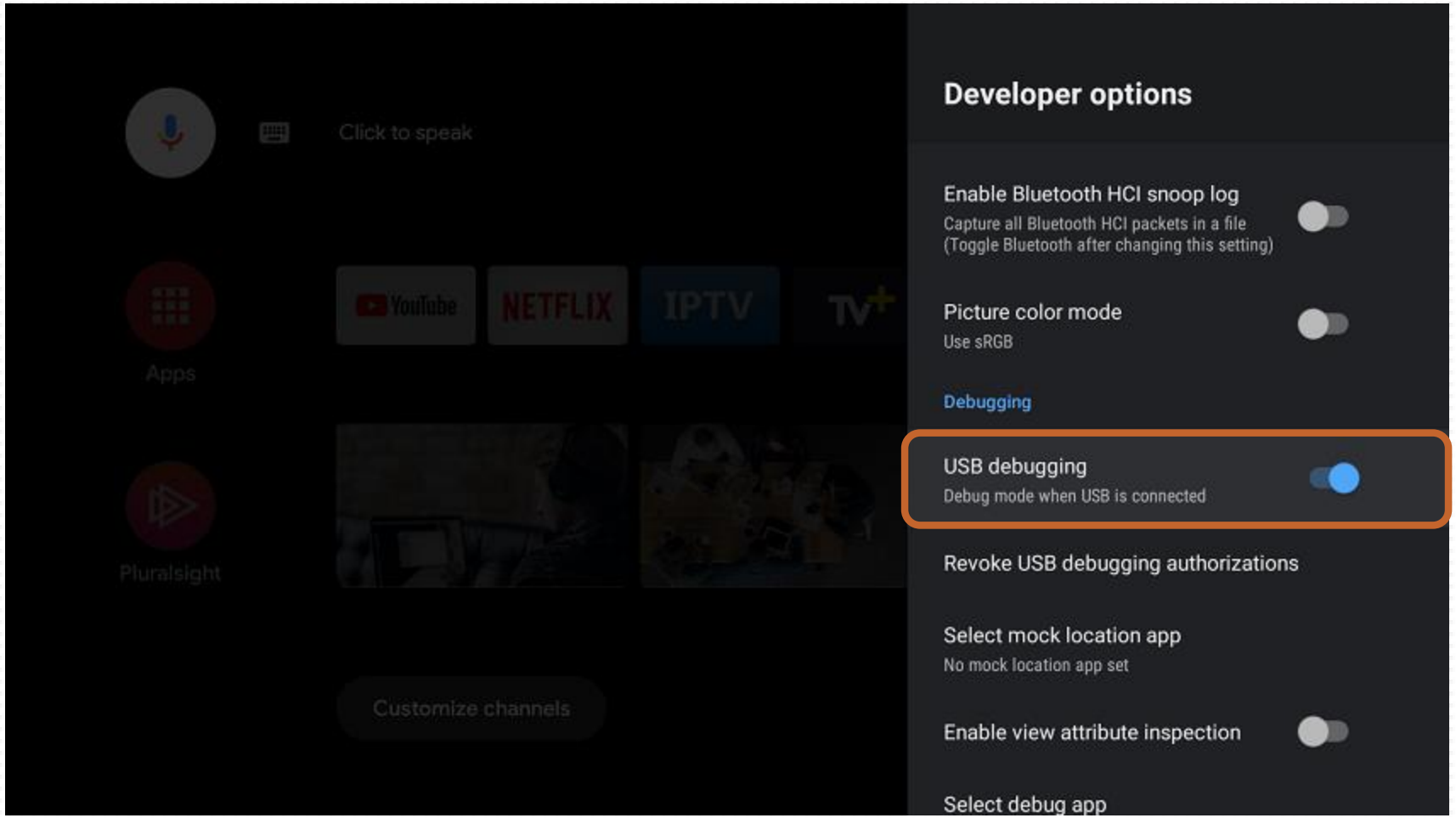
Connect Device (Android TV) over USB I



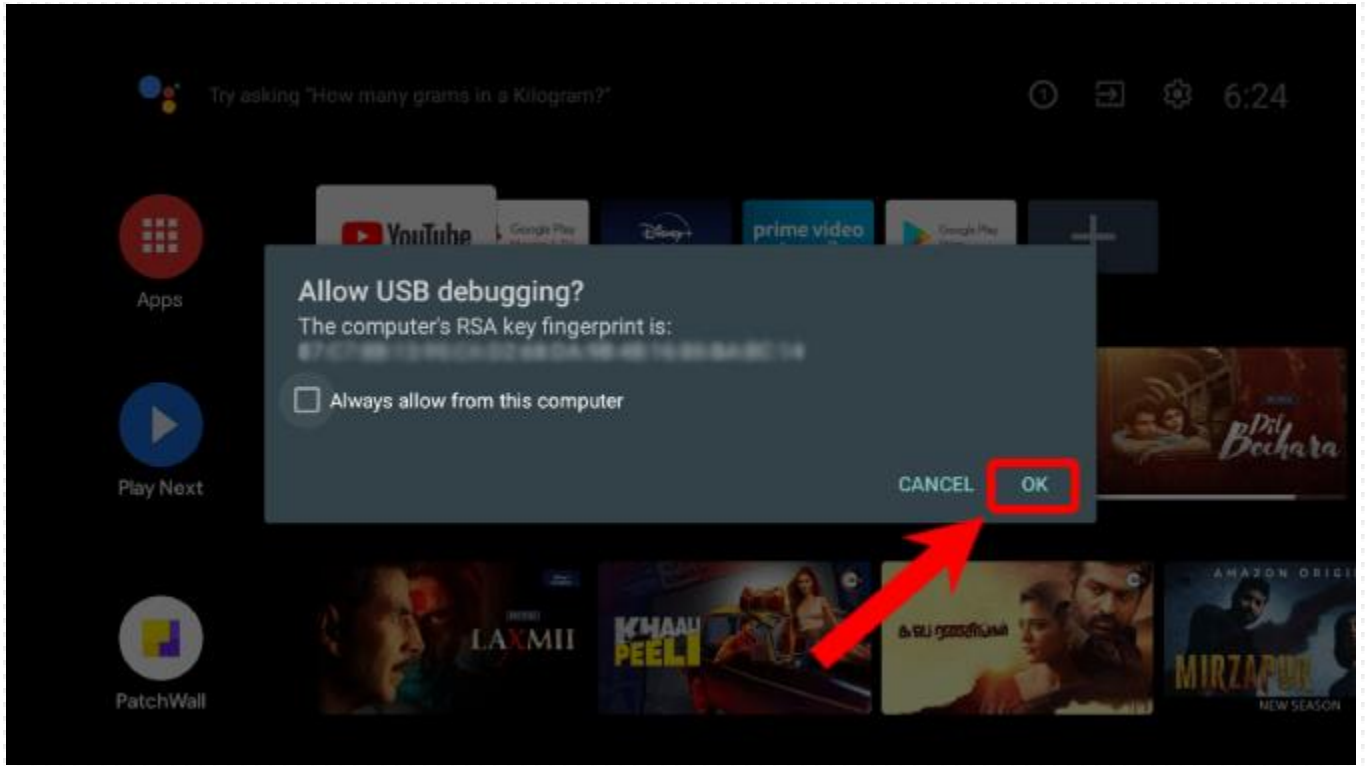
Connect Device (Android TV) over USB II



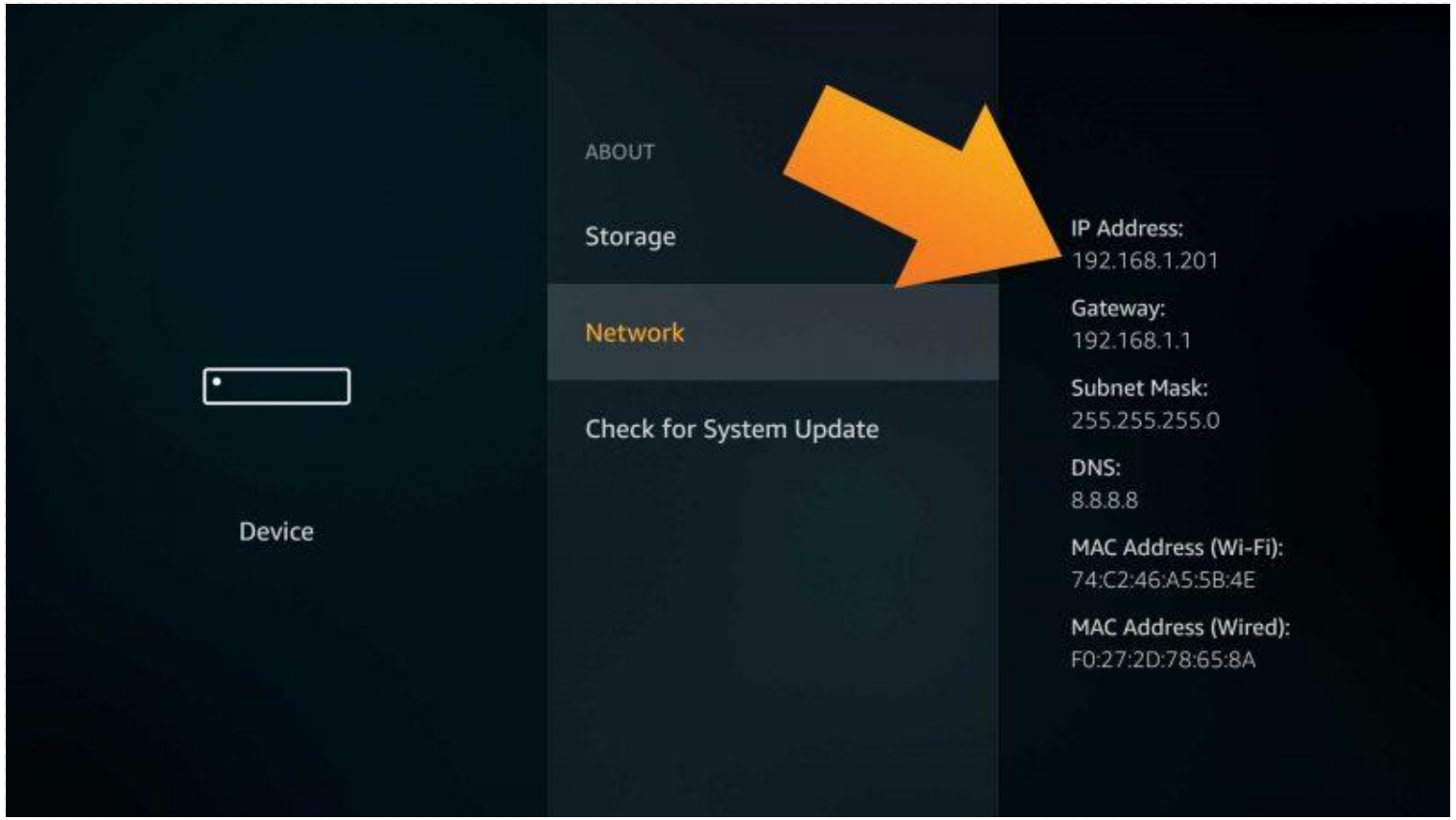
Connect Device (Android TV) over USB III



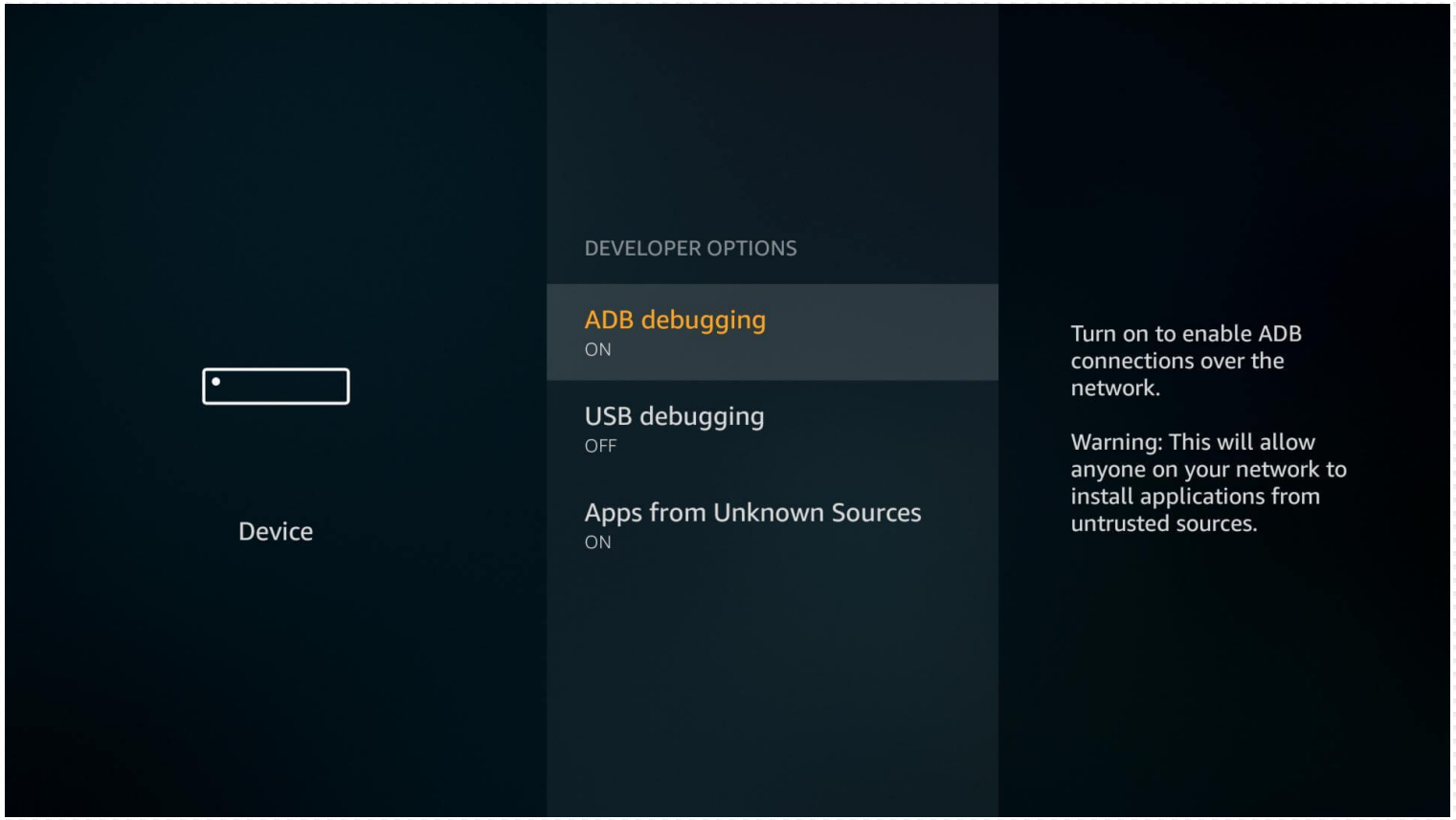
Connect Device (Android TV) over USB IV



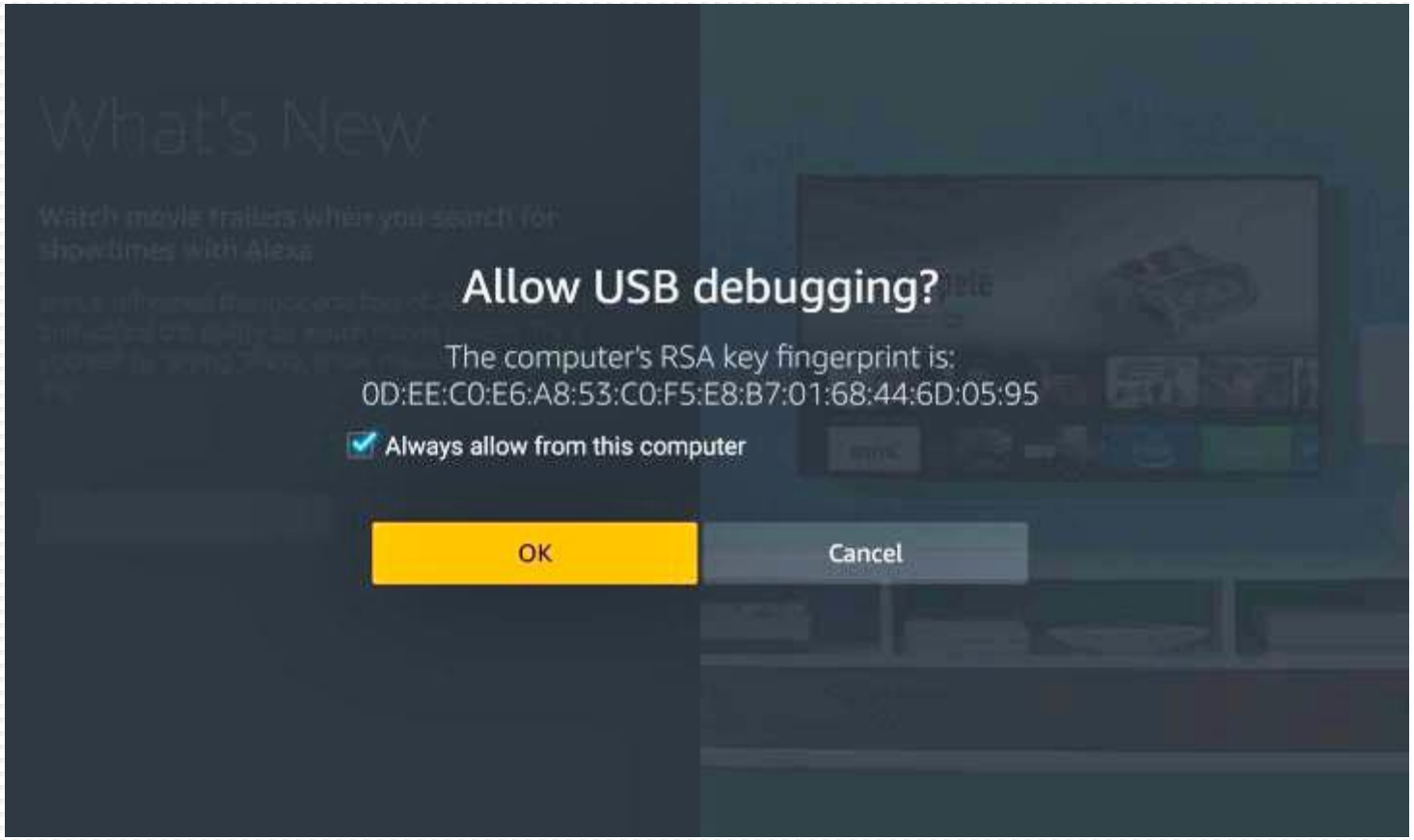
Connect Device (Fire TV) over Network I



Connect Device (Fire TV) over Network II



Connect Device (Fire TV) over Network III

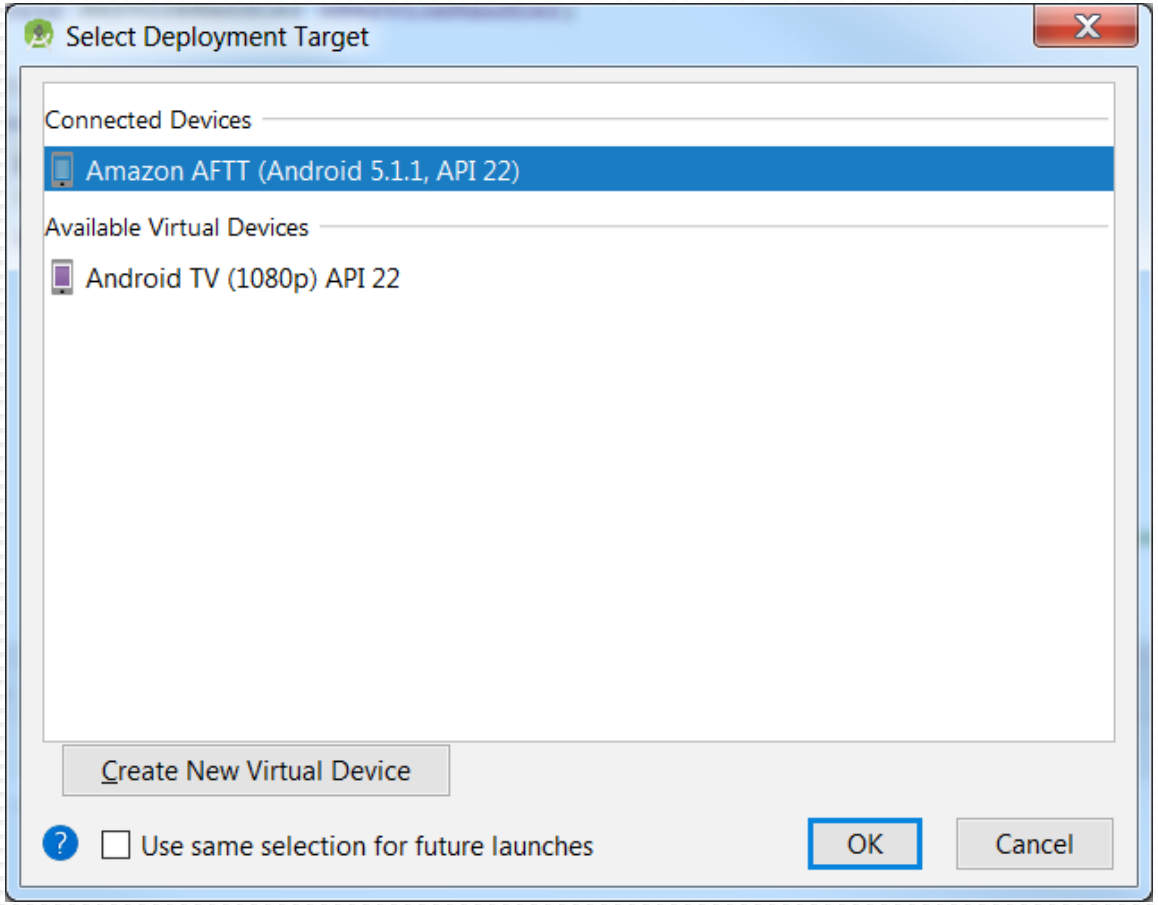


Connect Device (Fire TV) over Network IV

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)



Installing App on Device (Fire TV)

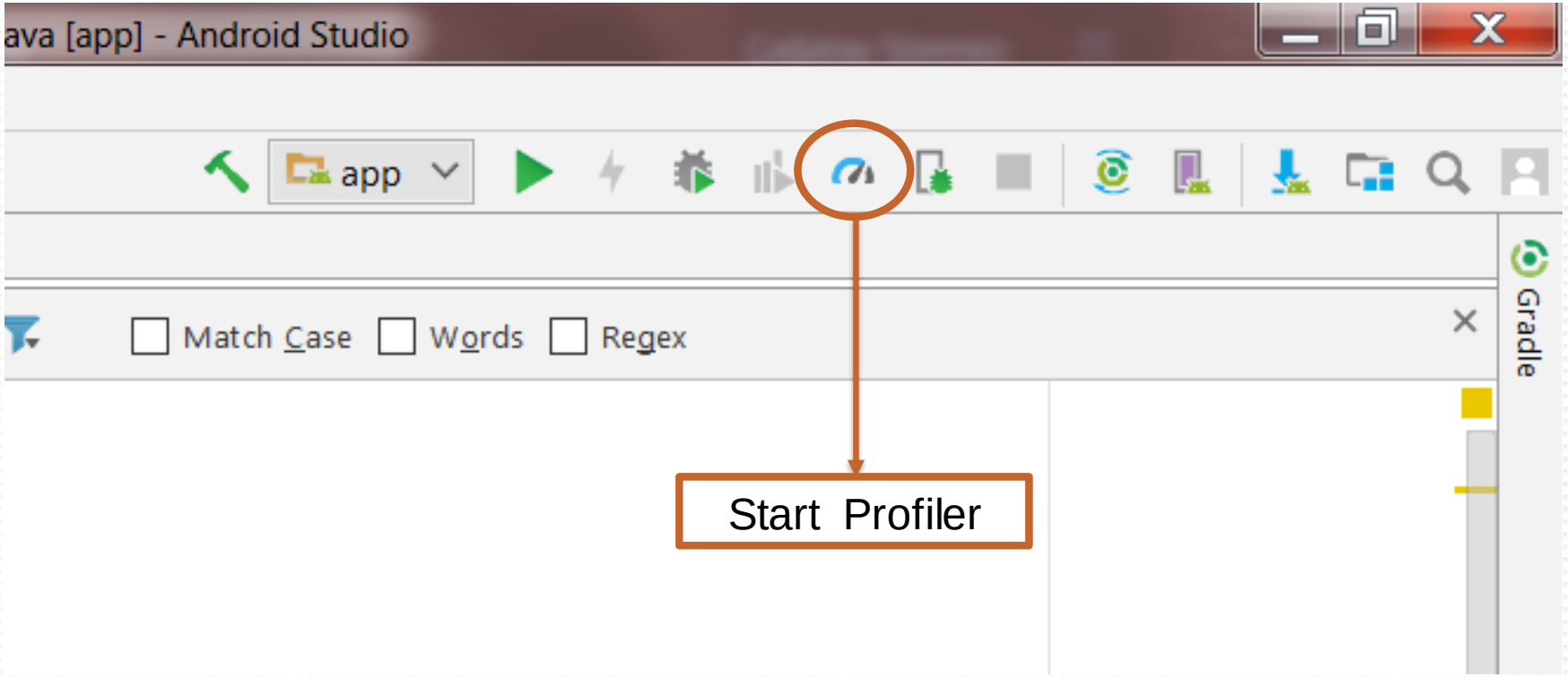
- You can also install external apk files to Android devices
- "Apps from Unknown Sourced" option should be ‘ON’ in developer options menu

adb install options

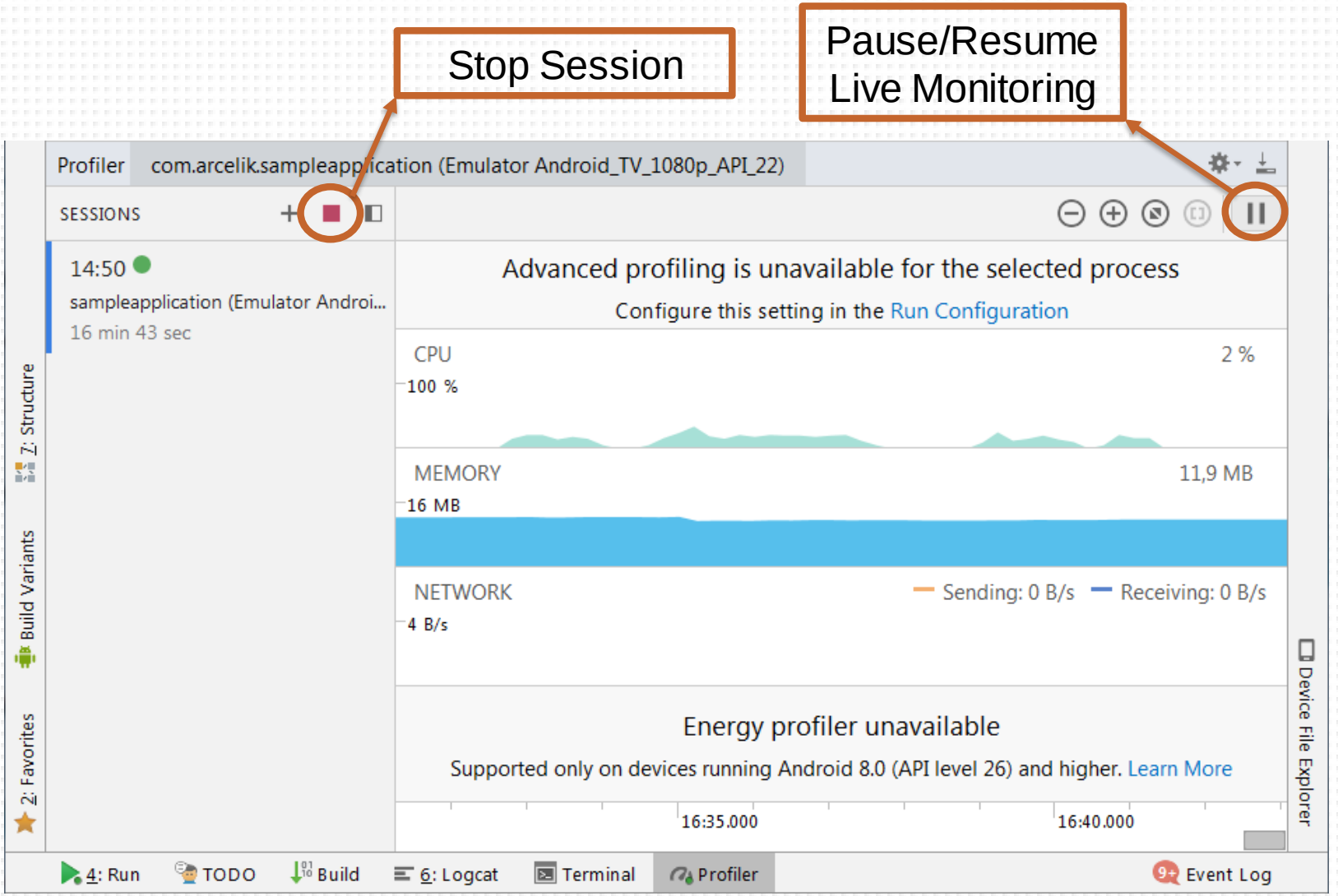
Command	Action
adb install test.apk	
adb install **-l** test.apk	forward lock application
adb install **-r** test.apk	replace existing application
adb install **-t** test.apk	allow test packages
adb install **-s** test.apk	install application on sdcard
adb install **-d** test.apk	allow version code downgrade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

Monitoring App Performance

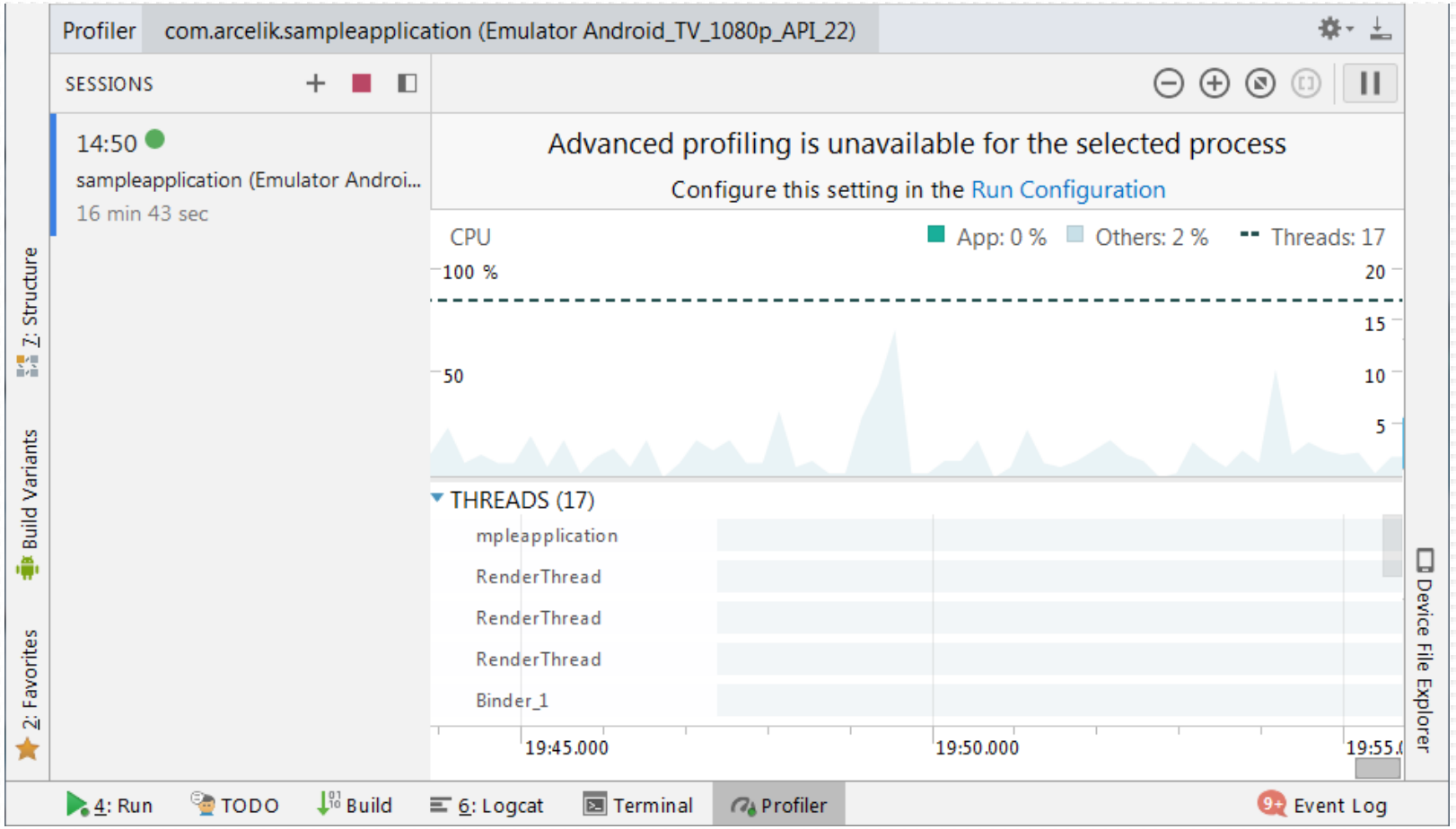
- Use Profiler to monitor the resource usage of your application



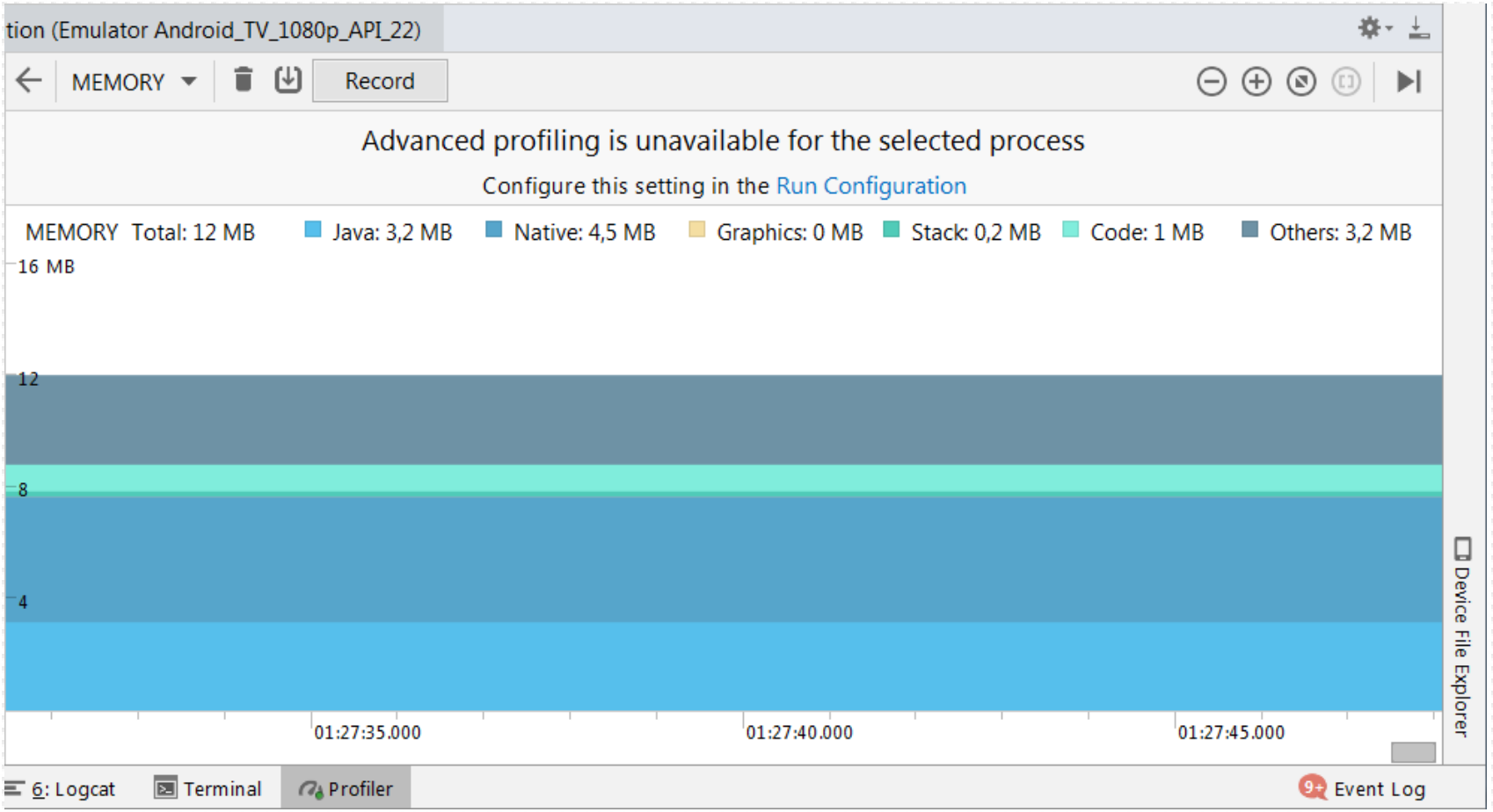
Profiler


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

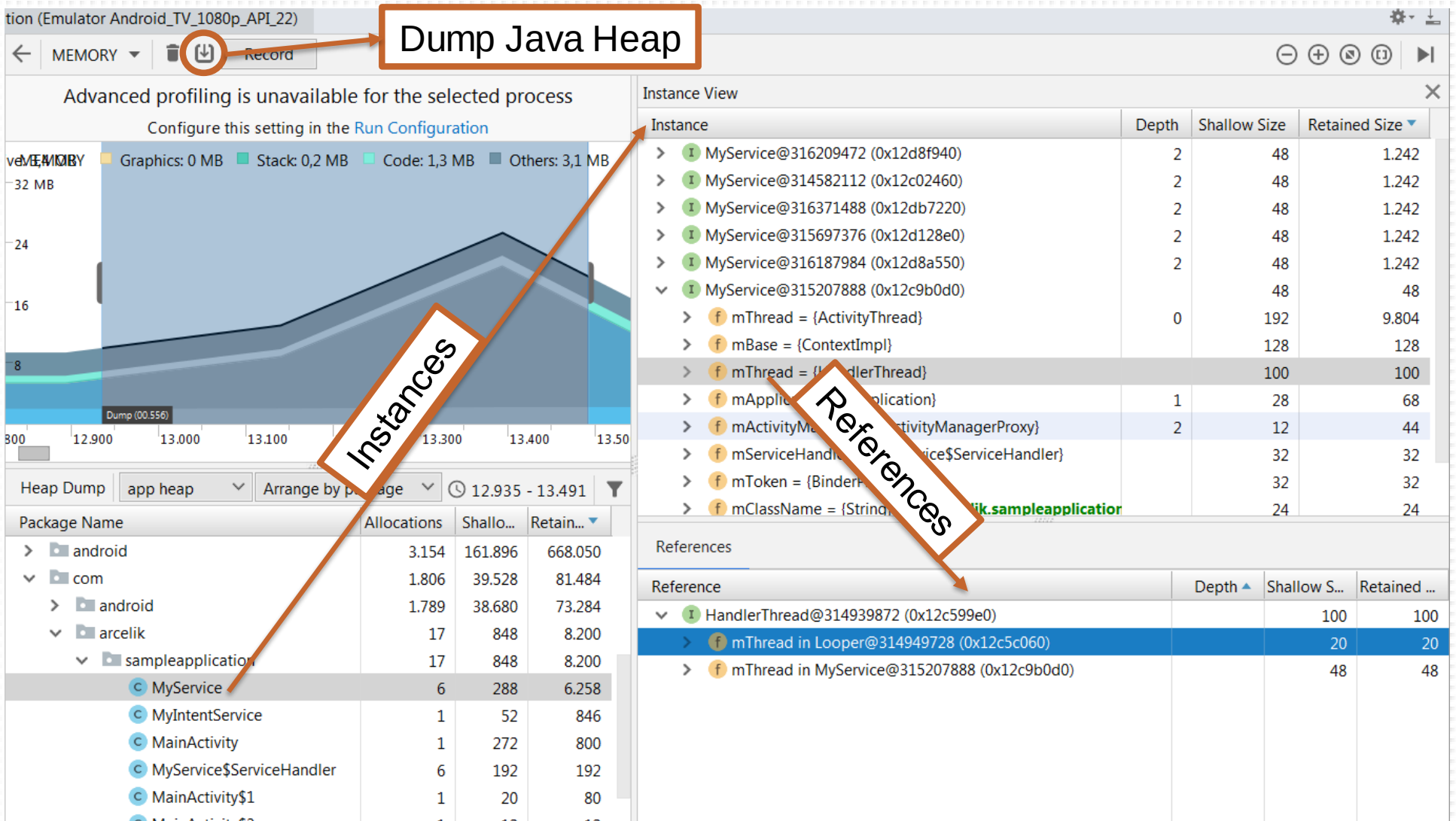
Profiler - CPU Usage



Profiler –Memeory Usage



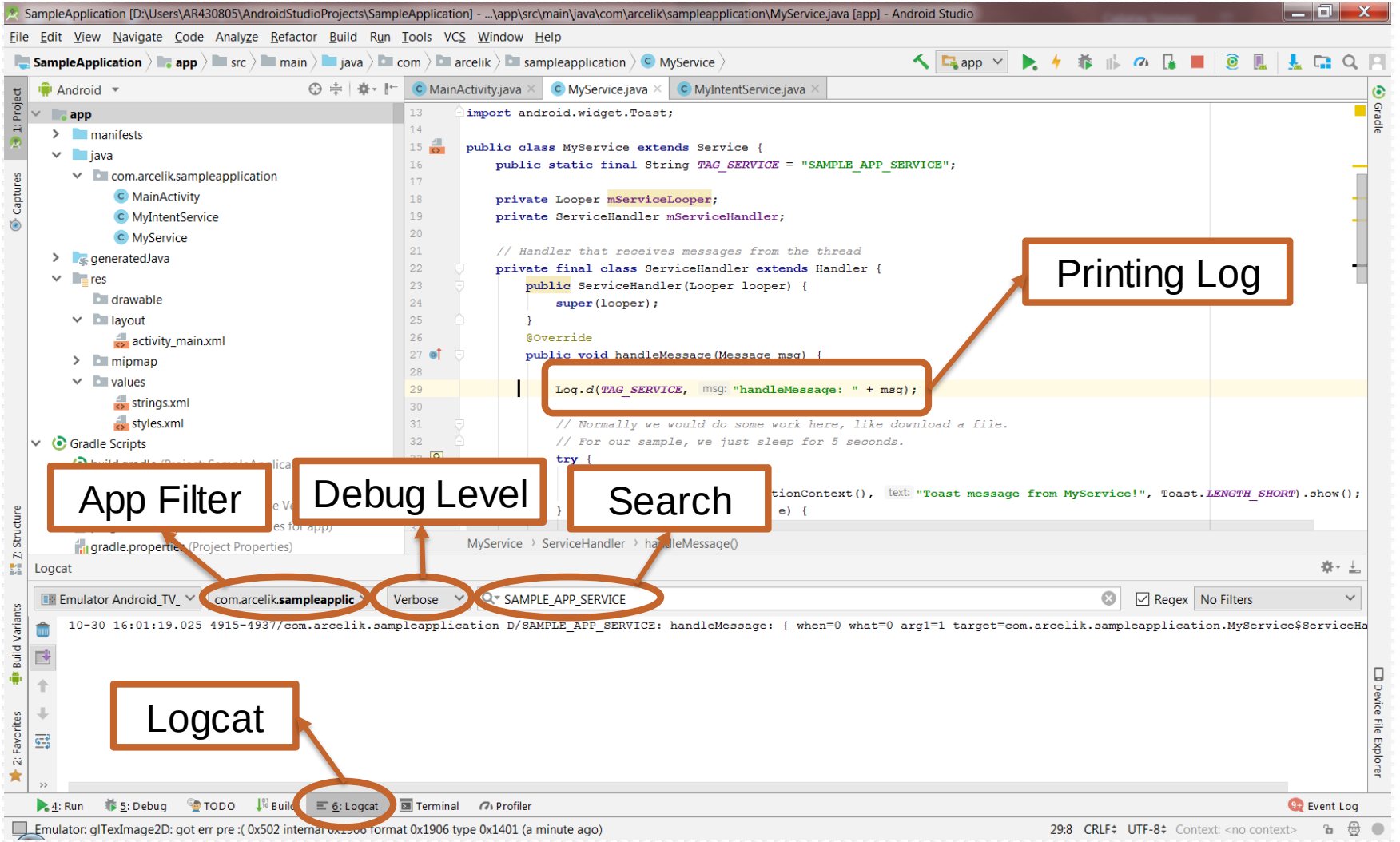
Profiler - Java Heap



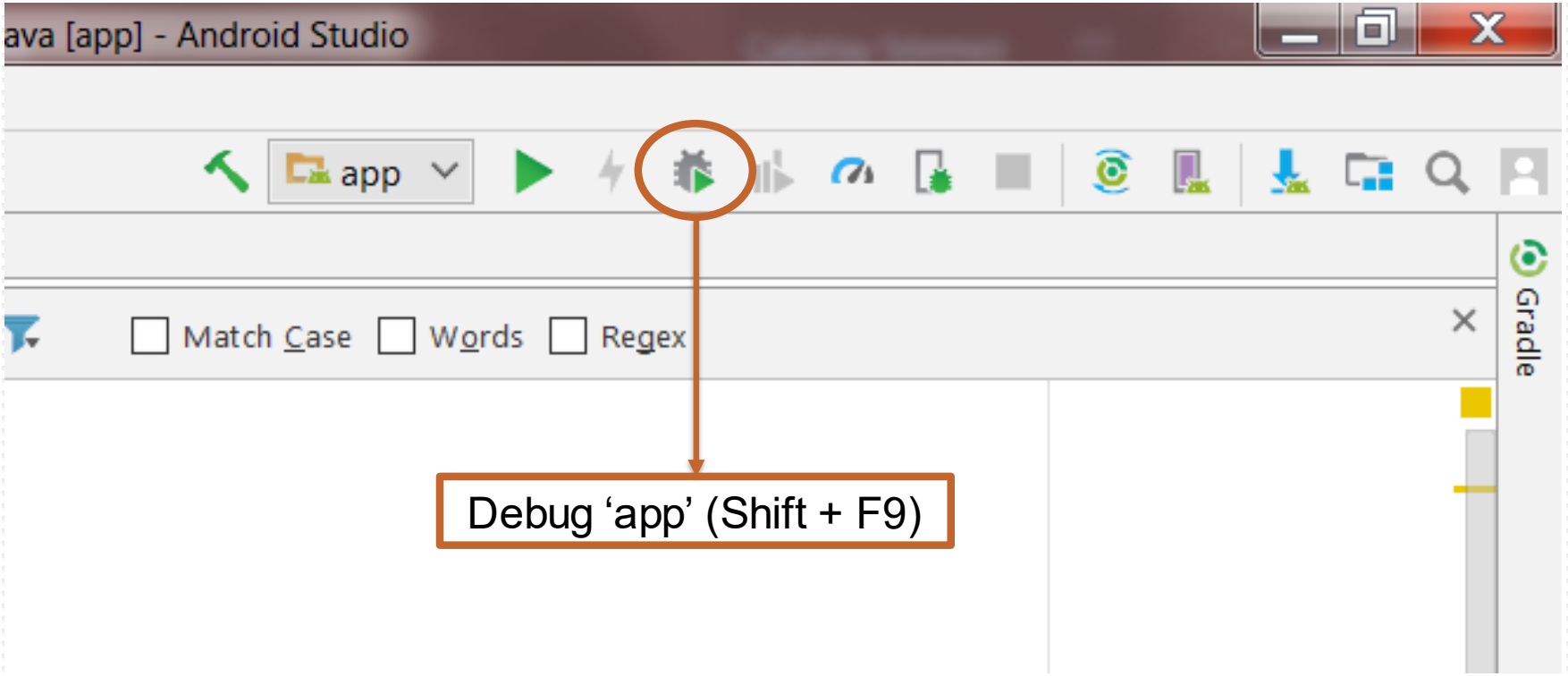
Debugging

- 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

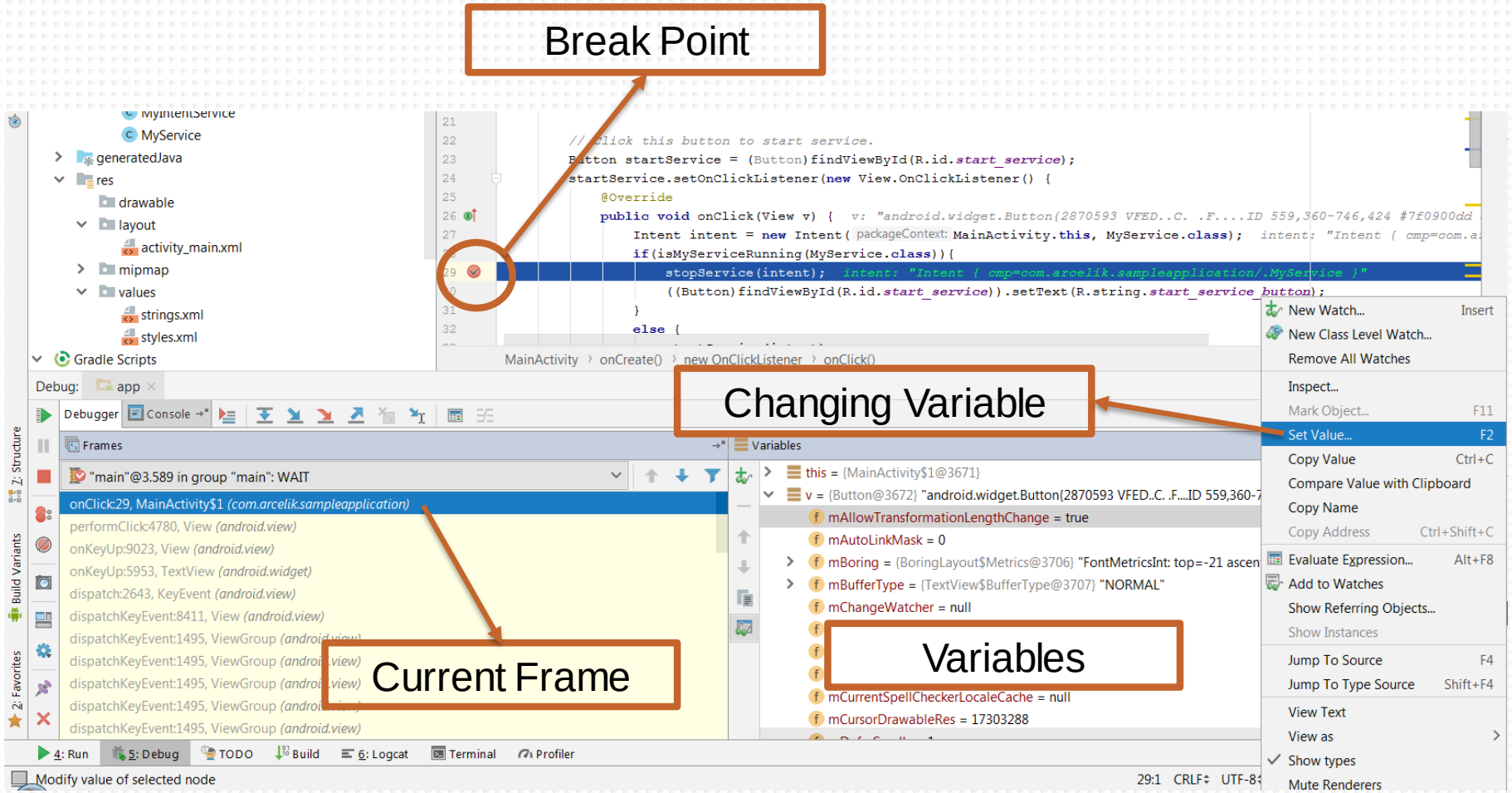
Viewing System Logs



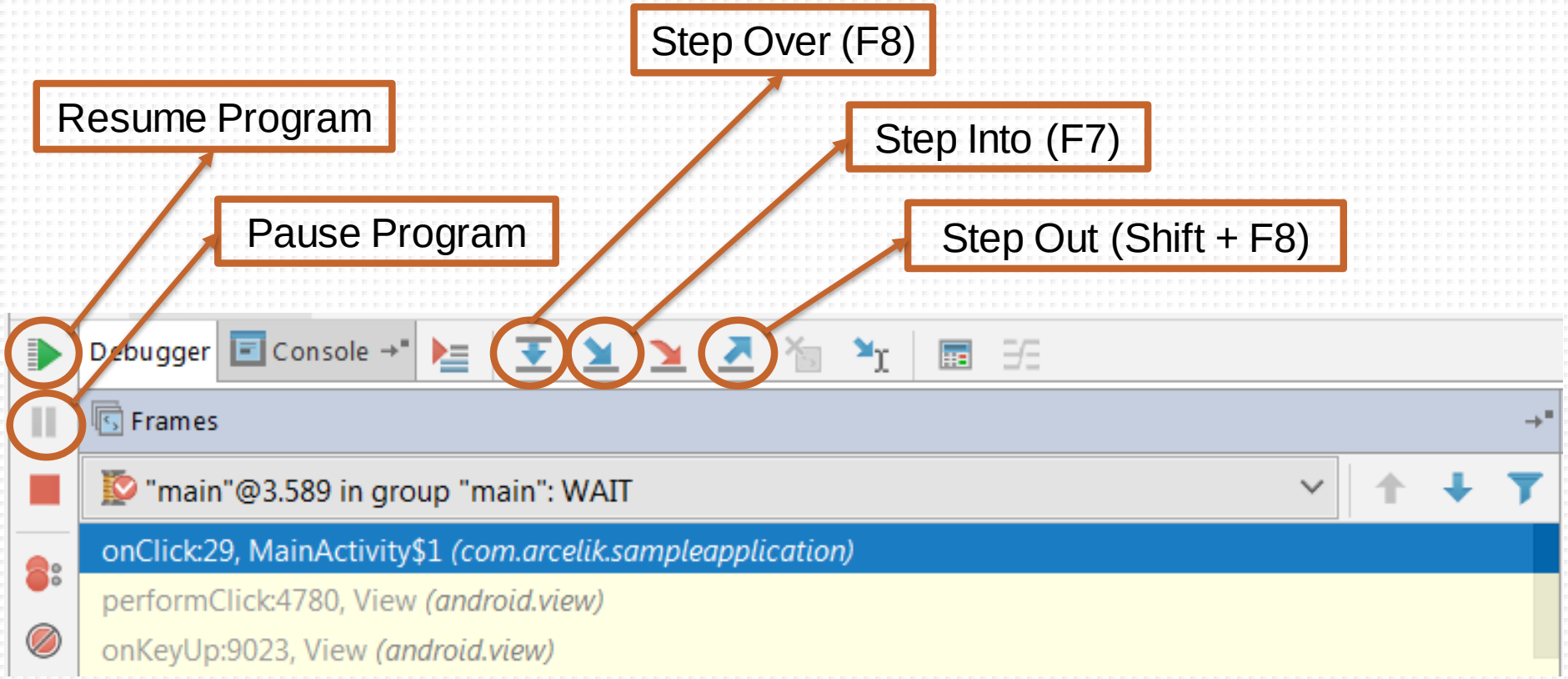
Starting Debugger



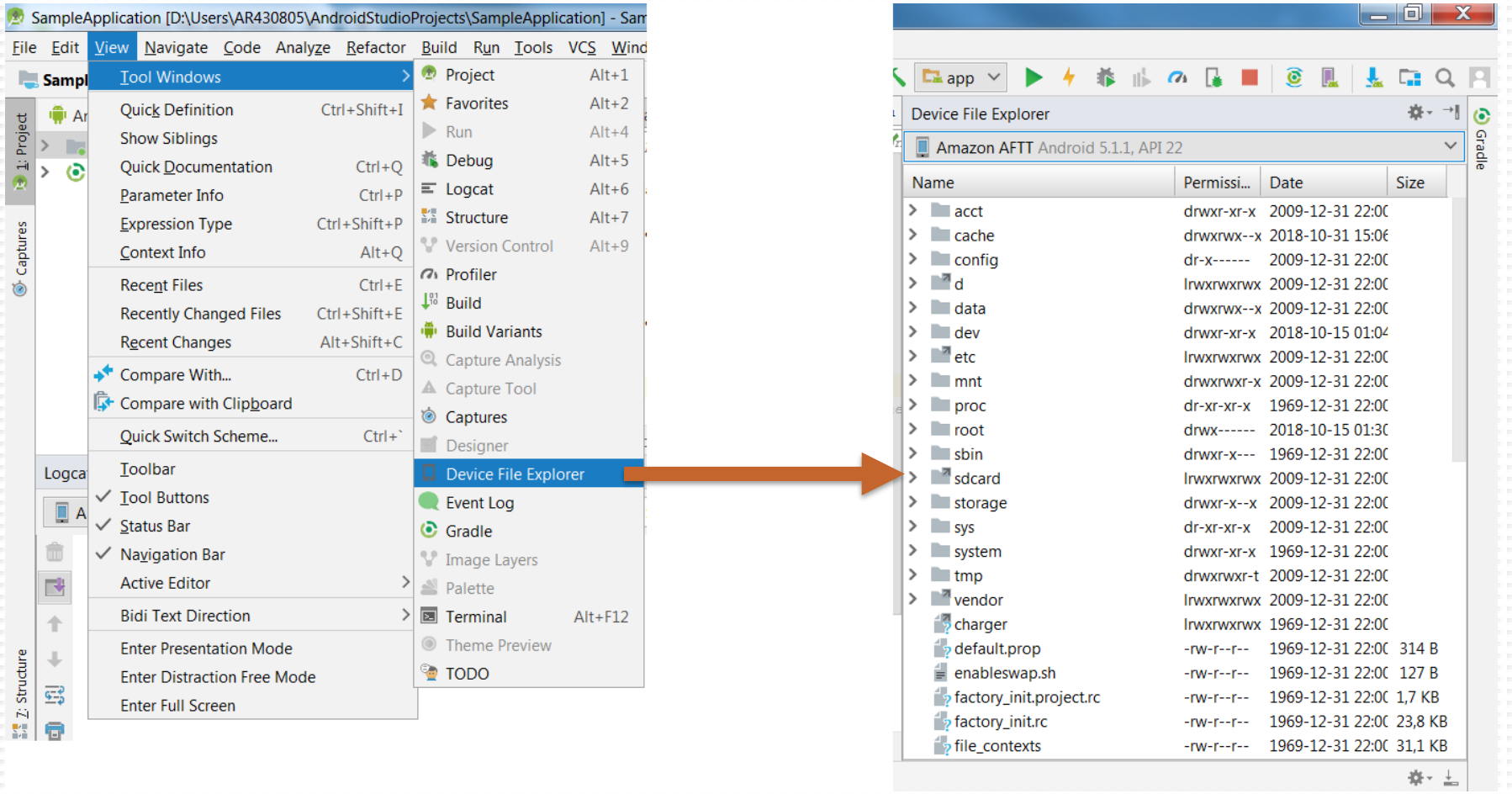
Using Breakpoint I



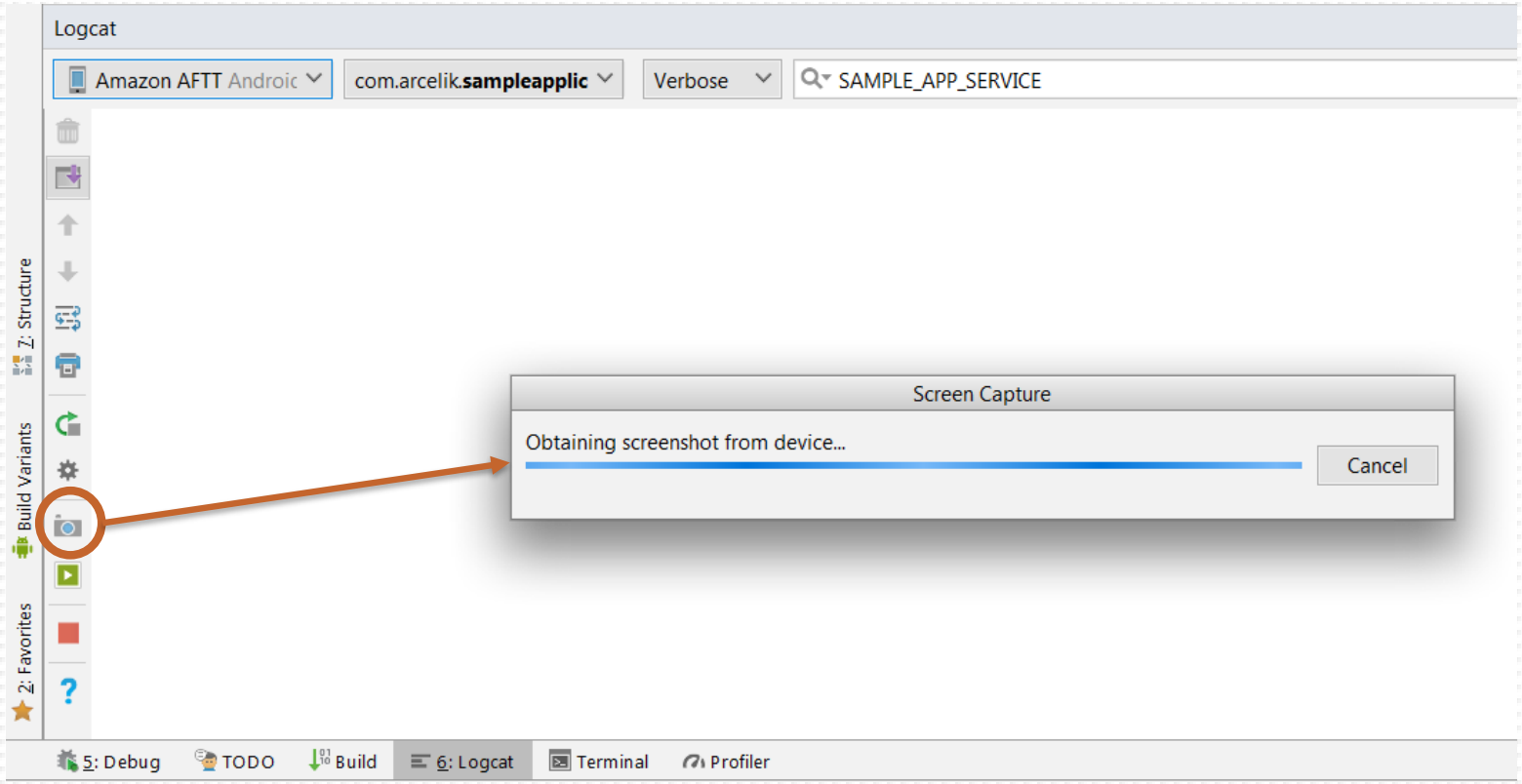
Using Breakpoint II



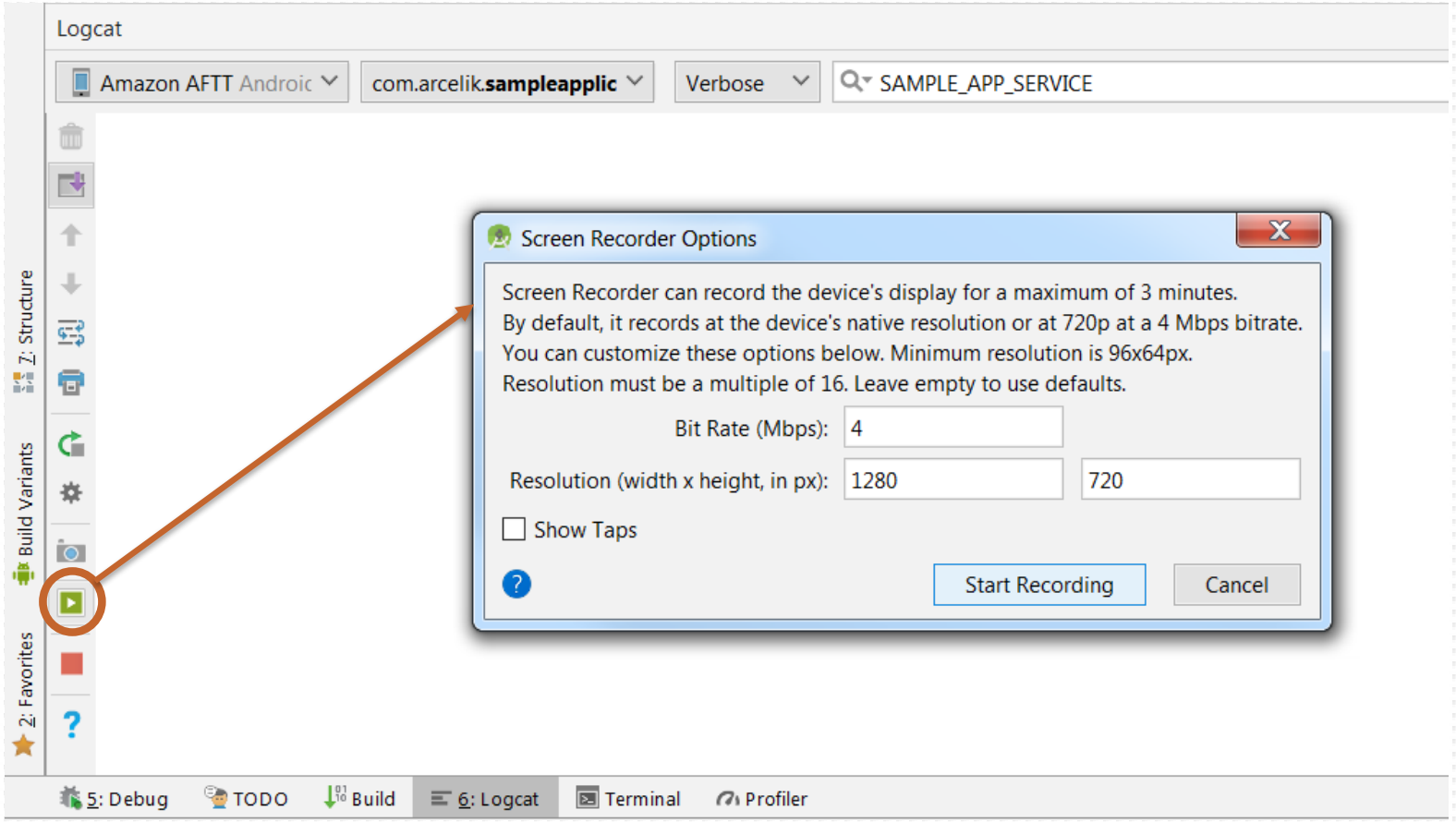
View On-Device Files



Take screenshot



Record Video



QUESTIONS?