

# Paixon



Before we start...

# Paixon

Wer die Android-Studio-Installation noch nicht gestartet hat, jetzt starten:

→ <https://developer.android.com/studio>

Code (Übungen und Lösungen) sowie Slides auf GitHub:

→ <https://github.com/sschoeb/OST-Mobile-App-Engineering-2022/>

## Wer ist Java Entwickler?

Eclipse?

**Intellij IDEA?**

Wer ist Android-User?

Wer hat bereits für Android  
entwickelt?

iOS?

## Wer hat Erfahrungen mit Cross-Platform Frameworks?

Xamarin, Cordova, Ionic, ...

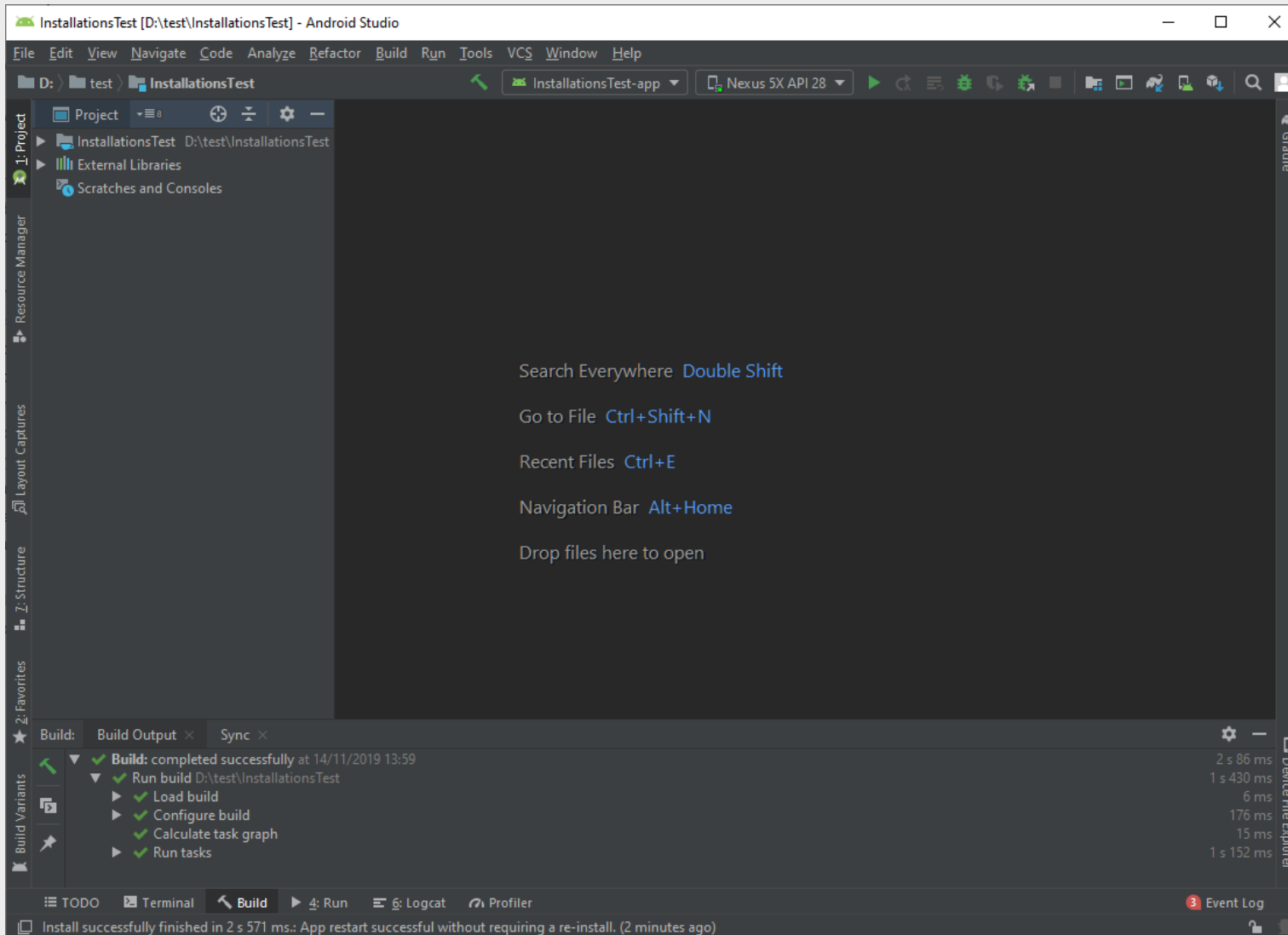
## Dev-Tools

Android Studio installiert?



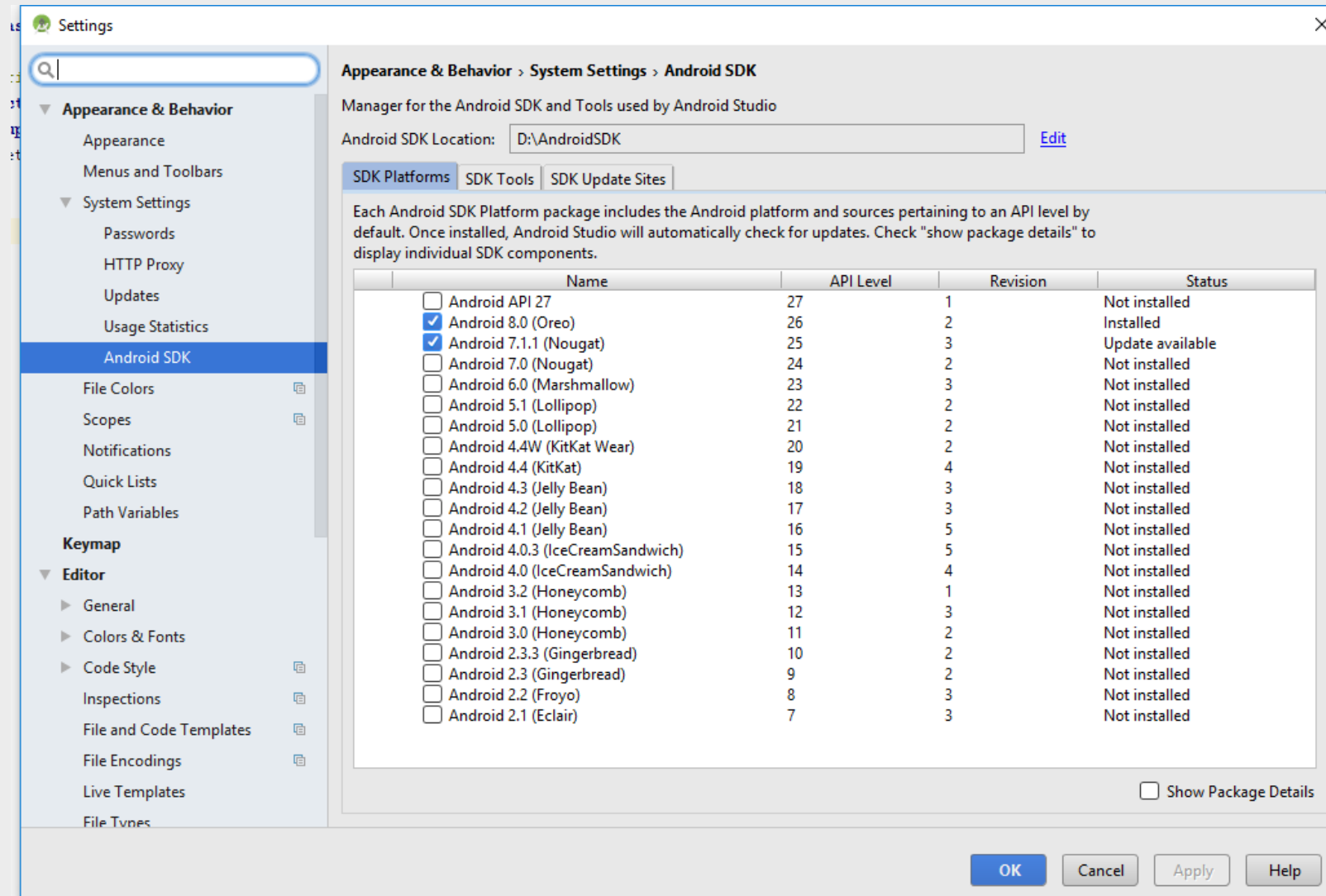
# Android Studio

# Paixon



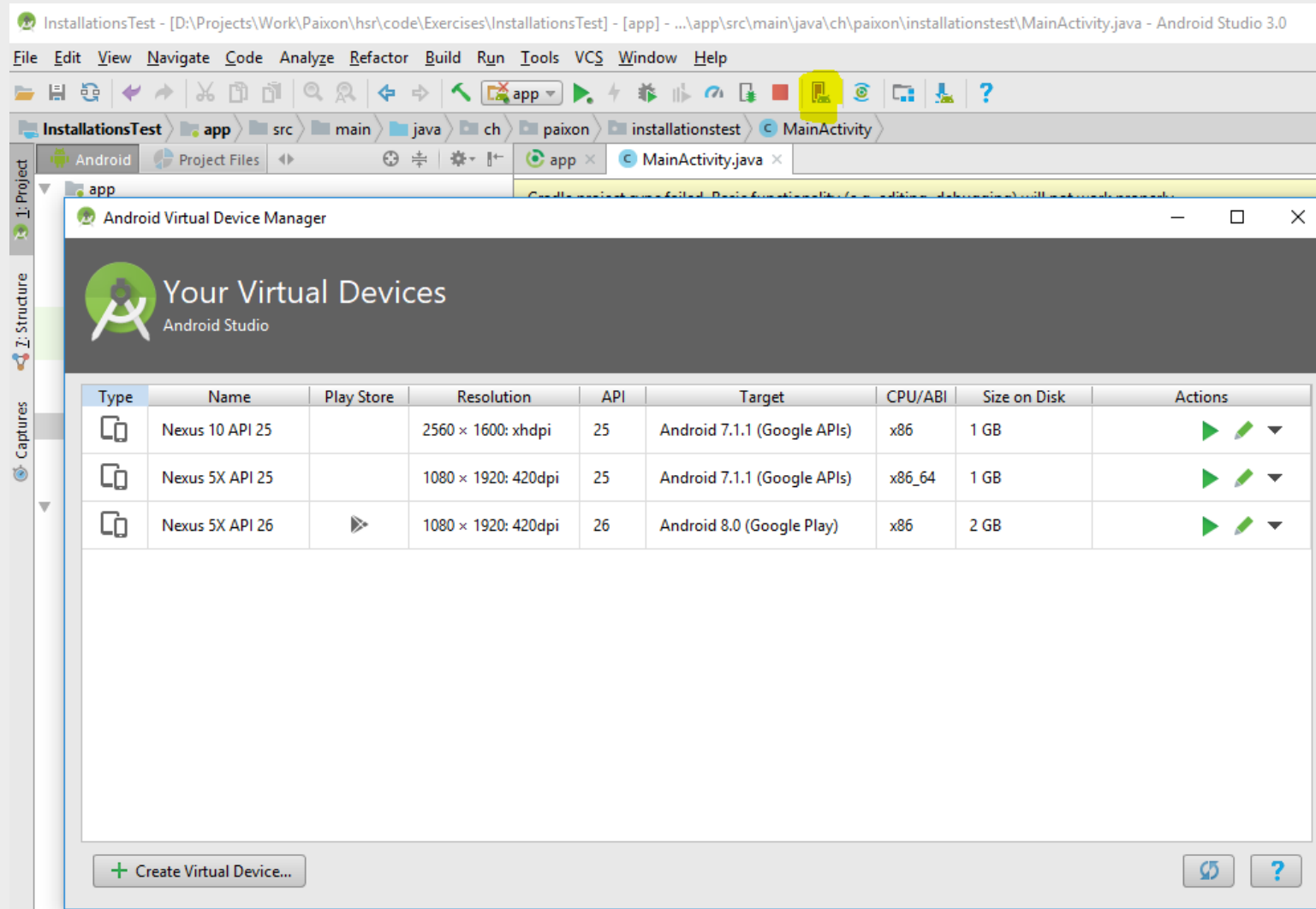
# SDK - Manager

# Paixon




# Android Virtual Device Manager

# Paixon



## Virtual Device Configuration

Virtual Device Configuration



### Select Hardware

Choose a device definition

Category

Phone

Tablet

Wear

TV

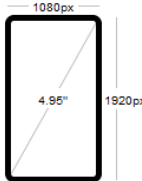
Q

Name	Size	Resolution	Density
Nexus S	4.0"	480x800	hdpi
Nexus One	3.7"	480x800	hdpi
Nexus 6P	5.7"	1440x2560	560dpi
Nexus 6	5.96"	1440x2560	560dpi
Nexus 5X	5.2"	1080x1920	420dpi
<b>Nexus 5</b>	<b>4.95"</b>	<b>1080x1920</b>	<b>xxhdpi</b>
Nexus 4	4.7"	768x1280	xhdpi
Galaxy Nexus	4.65"	720x1280	xhdpi
5.4" FWVGA	5.4"	480x854	mdpi
5.1" WVGA	5.1"	480x800	mdpi
4.7" WXGA	4.7"	720x1280	xhdpi
4.65" 720p (Galaxy Nexus)	4.65"	720x1280	xhdpi
4" WVGA (Nexus S)	4.0"	480x800	hdpi

New Hardware Profile

Import Hardware Profiles

Nexus 5



Size: normal

Ratio: notlong

Density: xxhdpi

Clone Device...

Previous


Next

Cancel

Finish

## Virtual Device Configuration

Virtual Device Configuration



System Image  
Android Studio

Select a system image


Recommended

x86 Images

Other Images

Release Name	API Level	ABI	Target
API 27 <a href="#">Download</a>	27	x86	Android API 27 (Google Play)
<b>Oreo</b>	<b>26</b>	<b>x86</b>	<b>Android 8.0 (Google Play)</b>
Nougat <a href="#">Download</a>	25	x86	Android 7.1.1 (Google Play)
Nougat <a href="#">Download</a>	24	x86	Android 7.0 (Google Play)

Oreo



API Level

**26**

Android

**8.0**

Google Inc.

System Image

**x86**

We recommend these Google Play images because this device is compatible with Google Play.

Questions on API level?  
See the [API level distribution chart](#)

Previous


**Next**


Cancel

Finish

Help



## Virtual Device Configuration

 Virtual Device Configuration


 **Android Virtual Device (AVD)**  
Android Studio


### Verify Configuration

AVD Name

 Nexus 5X	5.2 1080x1920 xxhdpi	<a href="#">Change...</a>
 Oreo	Android 8.0 x86	<a href="#">Change...</a>

Startup orientation  

 Portrait

 Landscape

Emulated Performance  
Graphics:

Device Frame ☒ Enable Device Frame

Show Advanced Settings

#### AVD Name

The name of this AVD.

Previous

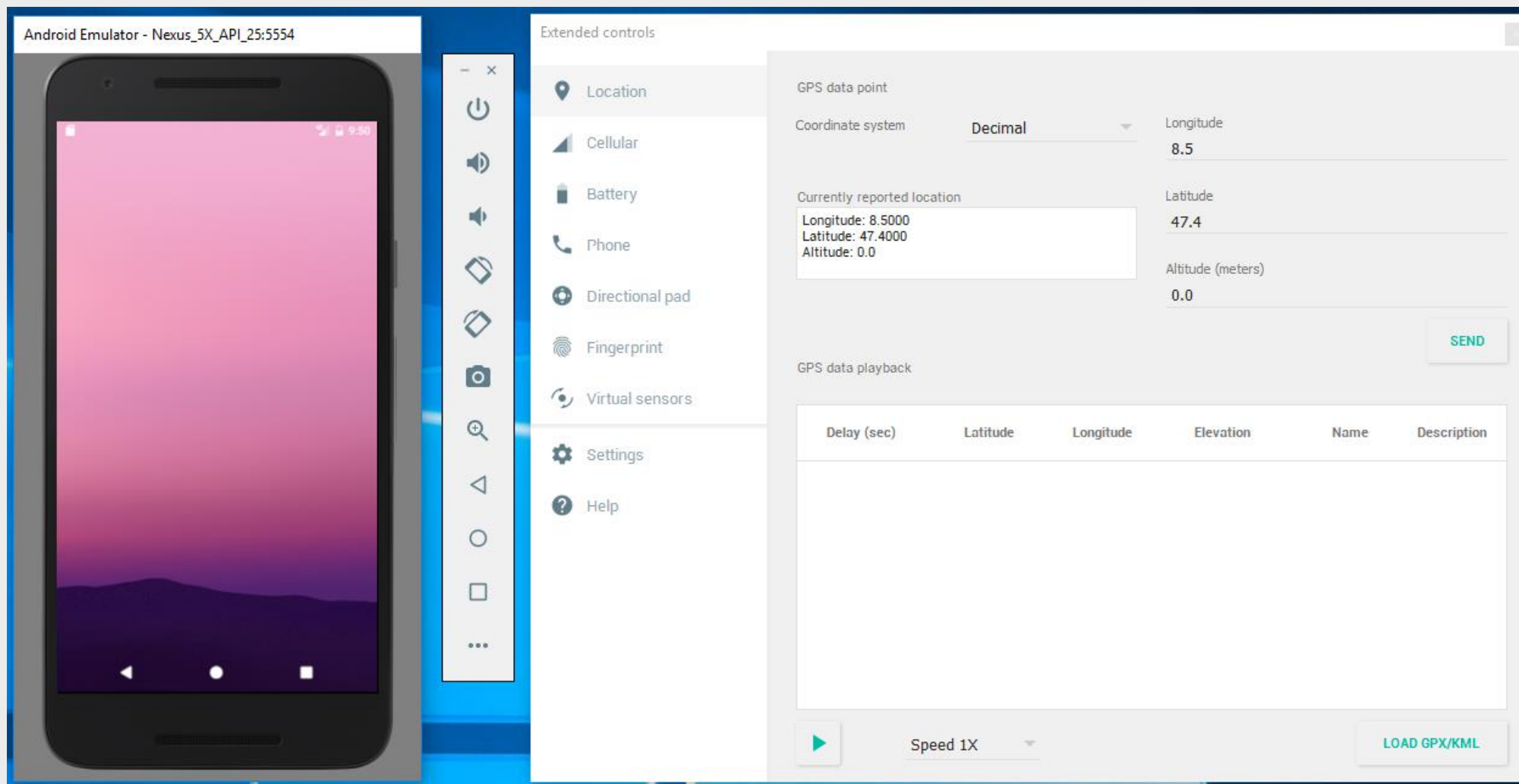
Next

Cancel

Finish

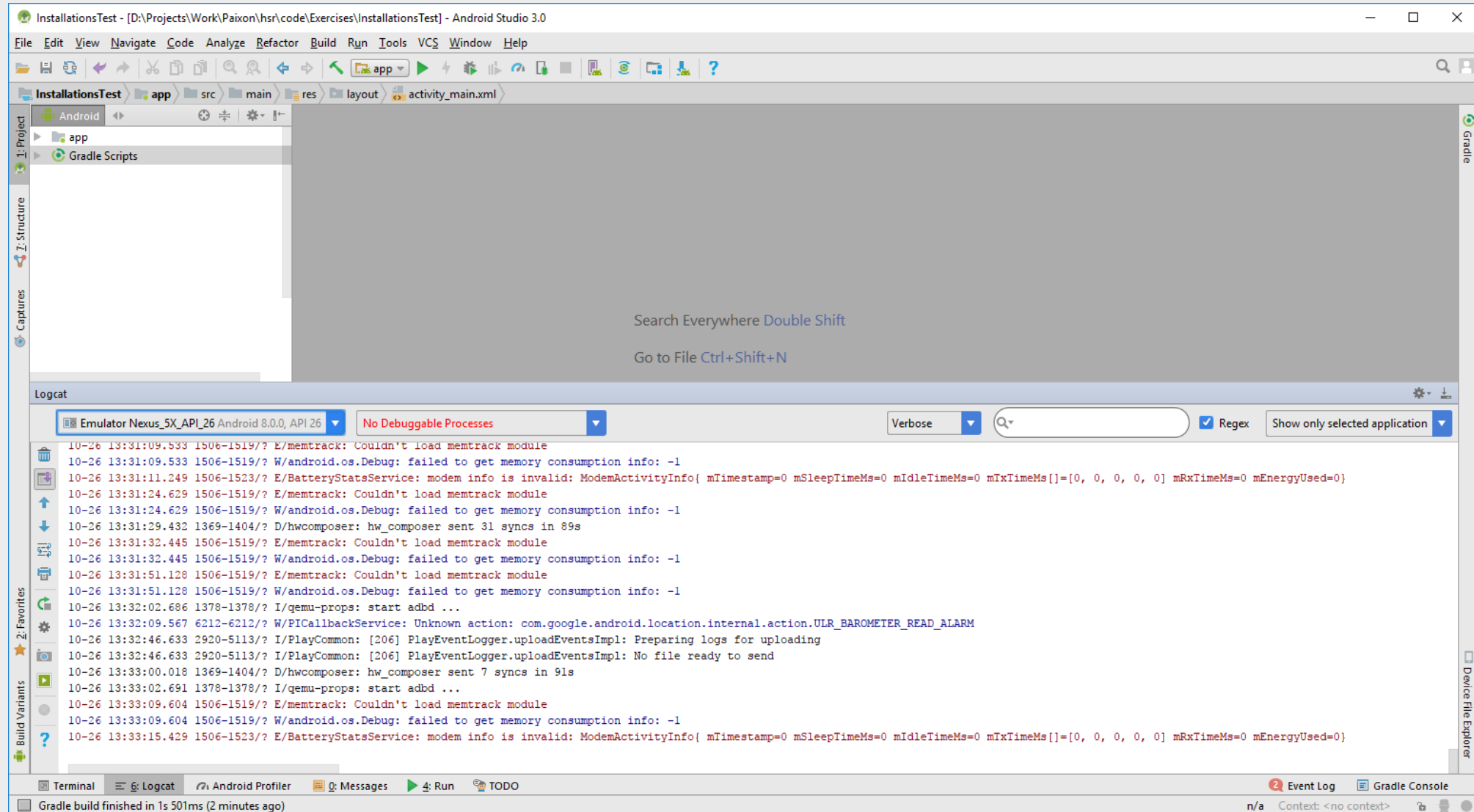
Help

## Android Emulator

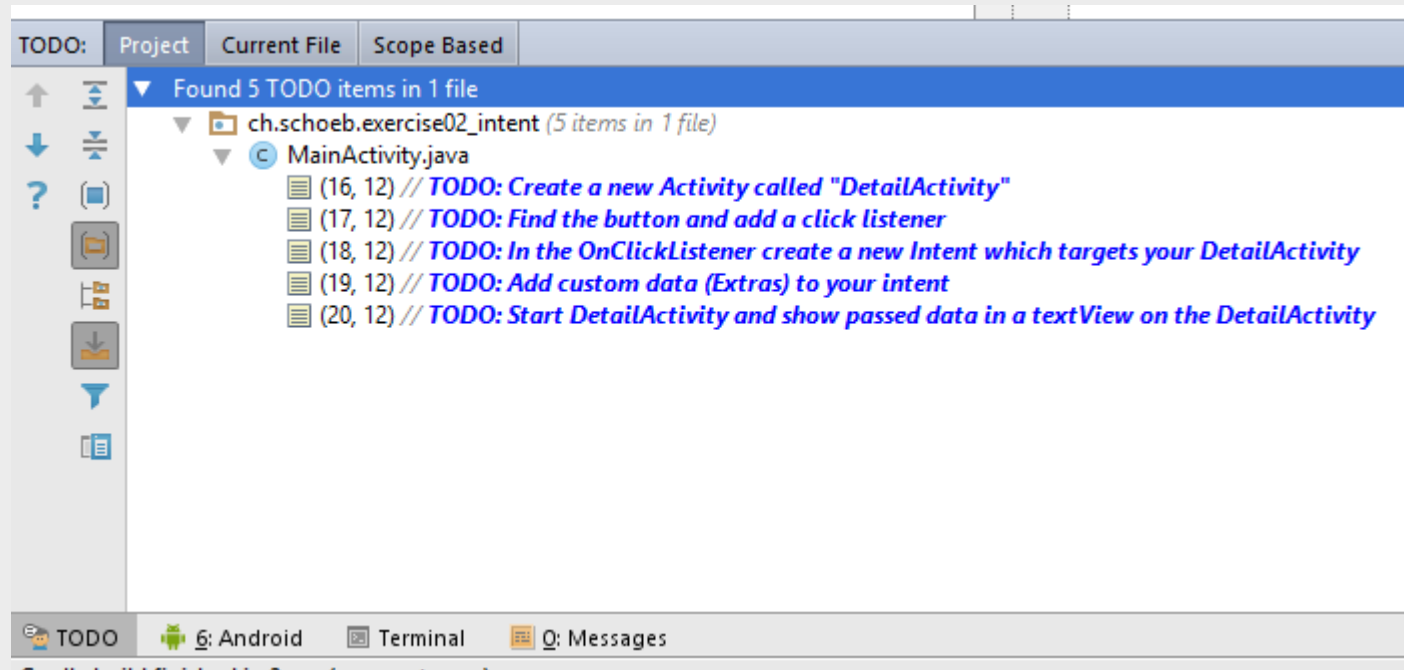


# Logcat

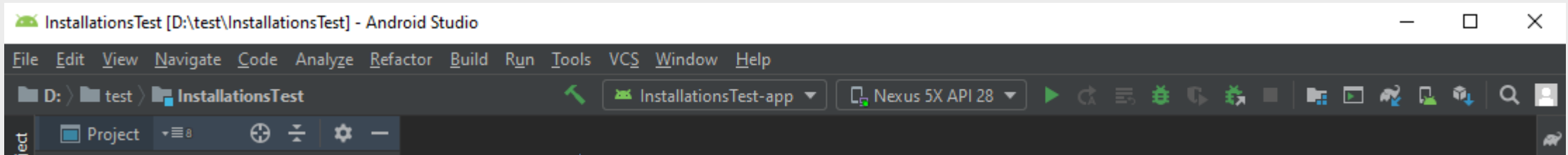
# Paixon







## Android Studio – Toolbar



Build

Build-Konfiguration

Device

Run  
Apply Changes (Code & Resources)

Apply Changes (Code)  
Debug

Stop

Projekt Einstellungen

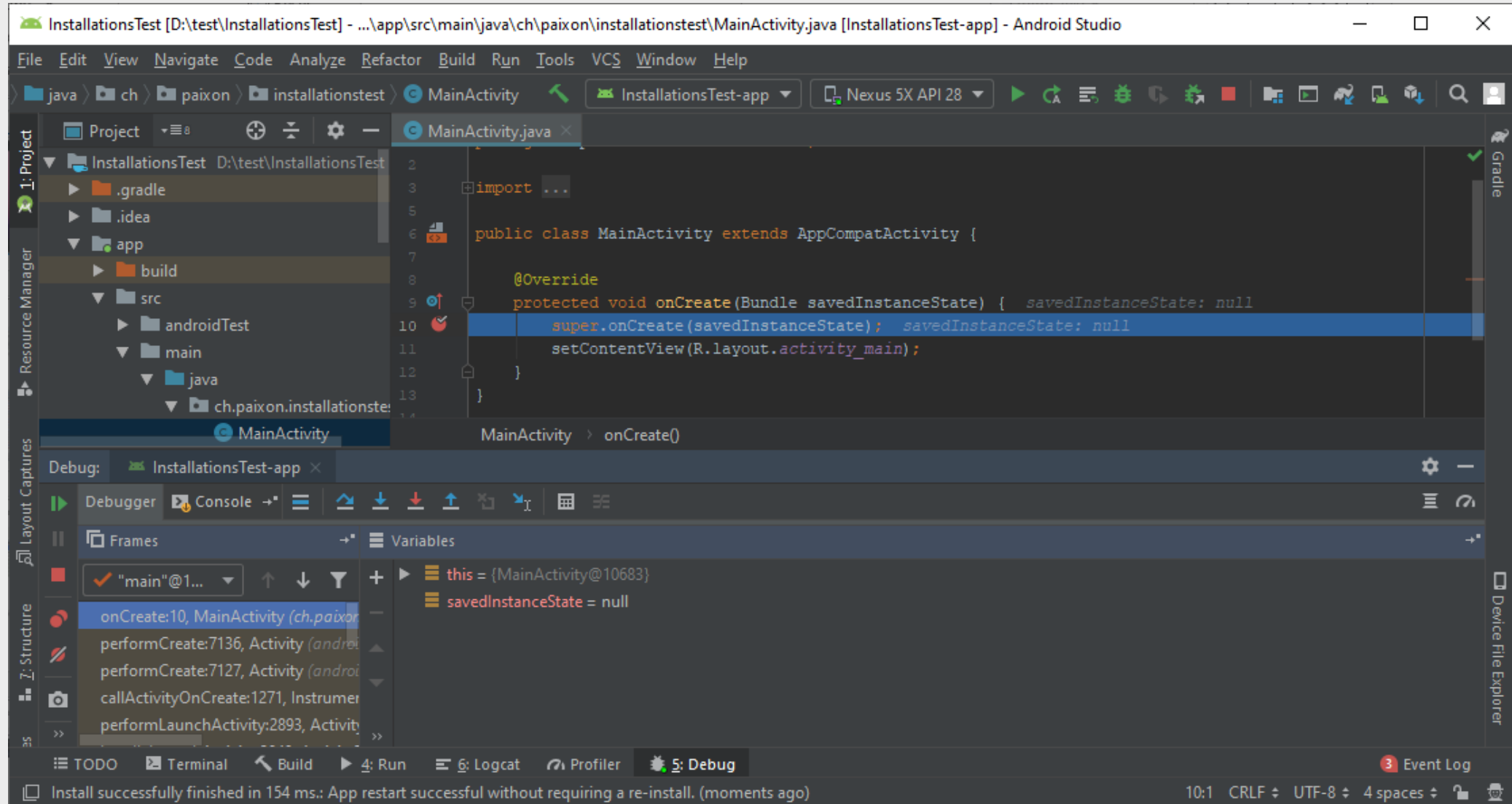
Gradle Sync

AVD Manager

SDK Manager

# Android Studio – Debugging

# Paixon



Let's start with a simple App

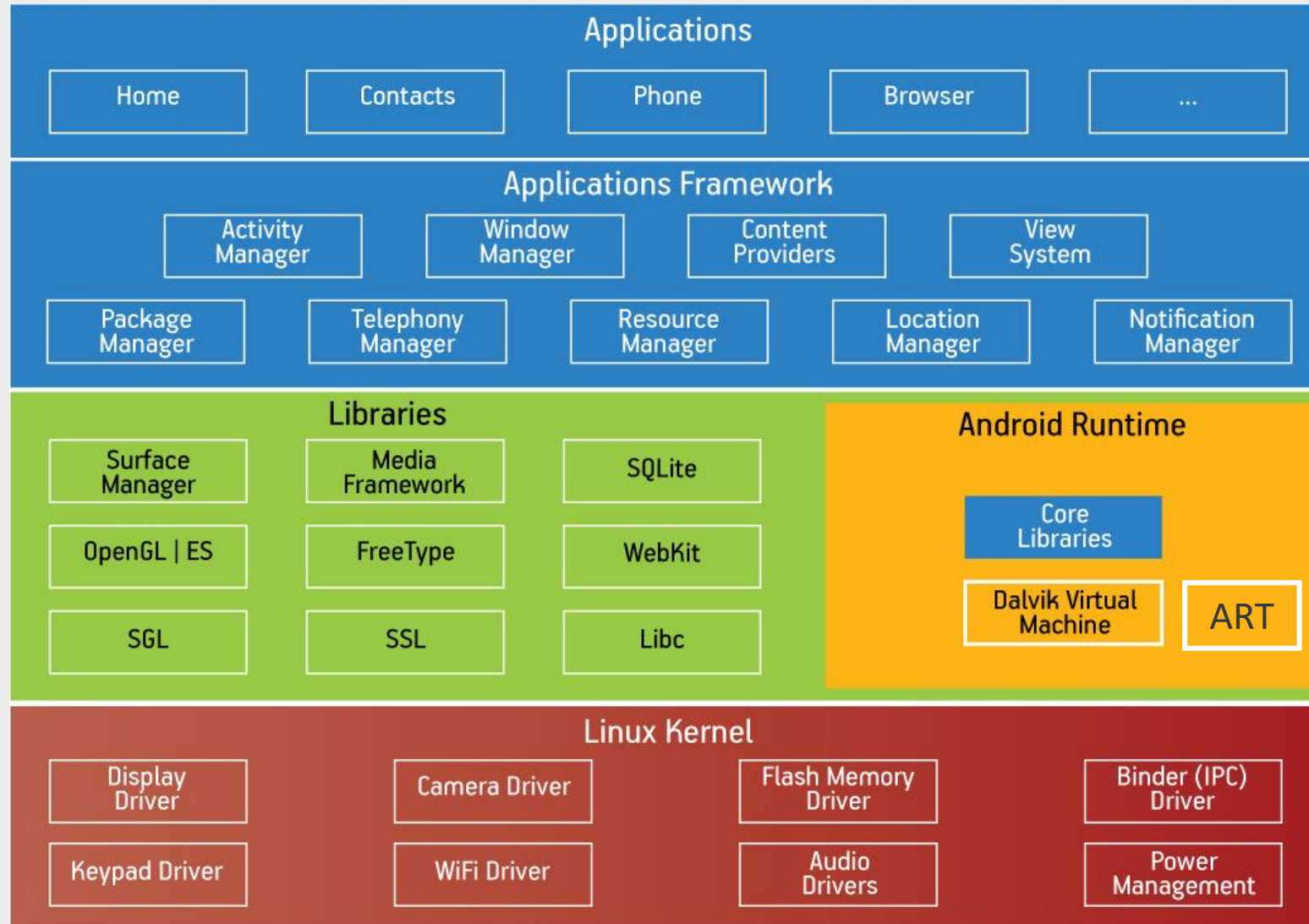
## **Gemeinsamer Teil**

- Neues Android Projekt erstellen
- Projekt auf dem Emulator starten

## **Aufgabe**

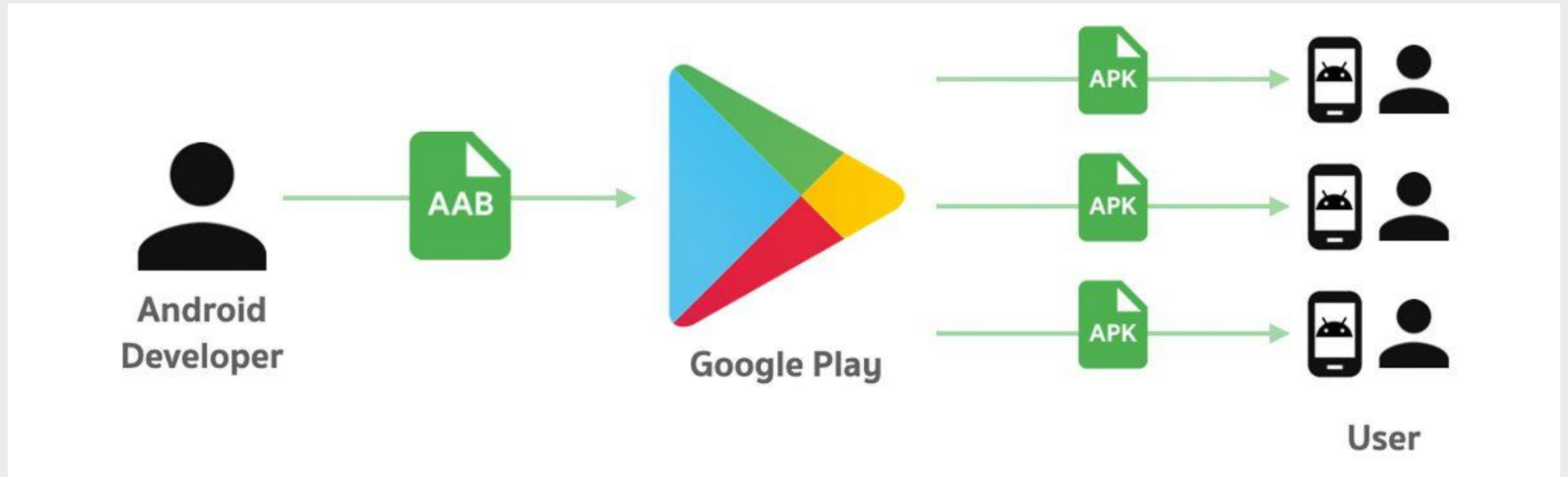
- Verschiedene Projekt-Ansichten analysieren
- Einen eigenen Filter im Android Monitor erstellen
- App im Debug-Modus starten

## Android System / App Basics



- App besteht aus
  - Code (Kotlin, Java oder C++)
  - Ressourcen
  - Manifest-File
- Android SDK Tools kompilieren alles in eine AAB-Datei (\*.aab)
  - Beinhaltet alle Informationen der App (Kompilierter Code, Ressourcen, ...)
- App läuft in einem eigenen Prozess
- App hat eigenen Linux User → Filesystem-Sandbox





- Alles was ihr benötigt aber nicht Sourcecode ist
  - Texte, Bilder, Layoutdefinitionen, ...
- Abgelegt im «res»-Ordner
- Bestehen hauptsächlich aus XML-Files

Declaration in Strings.xml:

```
<string name="app_name">Services</string>
```

Usage in any other XML-File:

```
android:label="@string/app_name"
```

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="ch.schoeb.services"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="8" android:targetSdkVersion="17" />

    <application android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity android:name="ch.schoeb.services.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

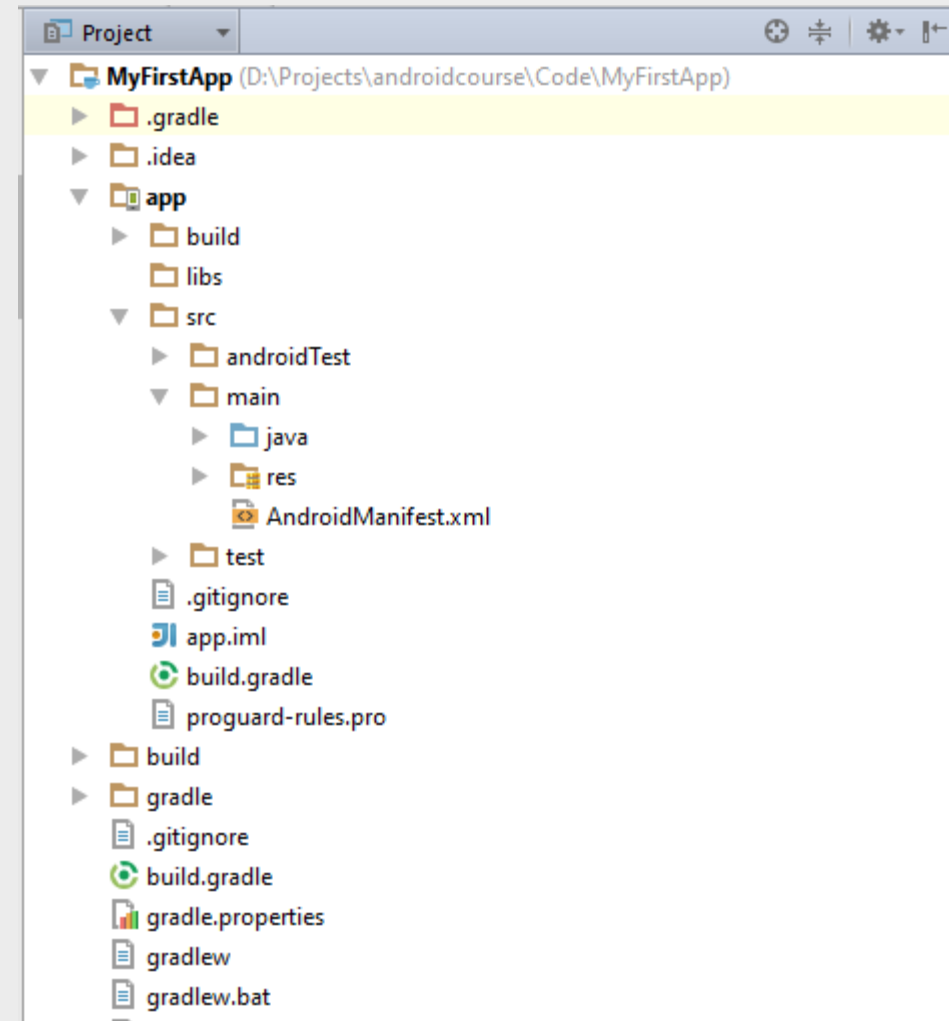
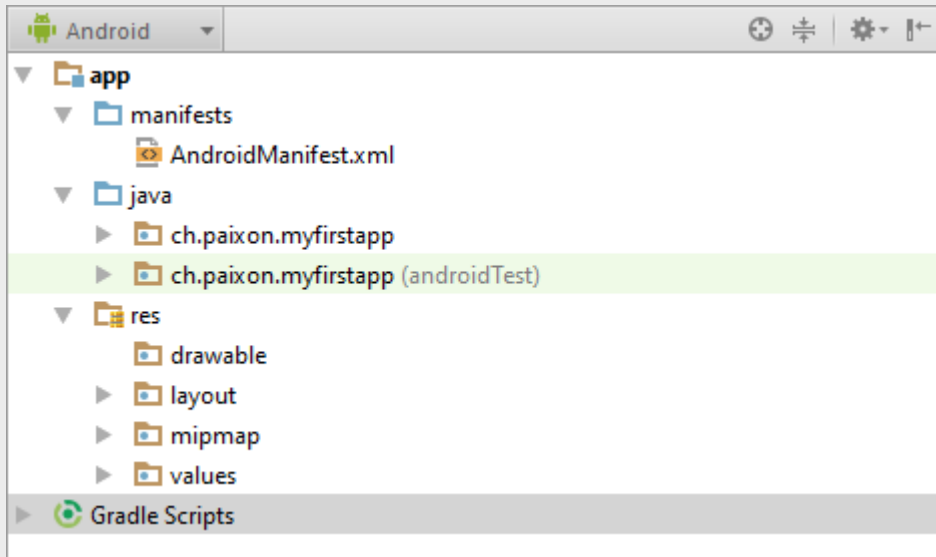
**Jede Applikation hat genau ein AndroidManifest.xml**

**Liefert Informationen über die App**

- Names the Java-Package
- Describes the components (Activities, Services, ...)
- Permissions
- Android API requirements
- List of linked libraries

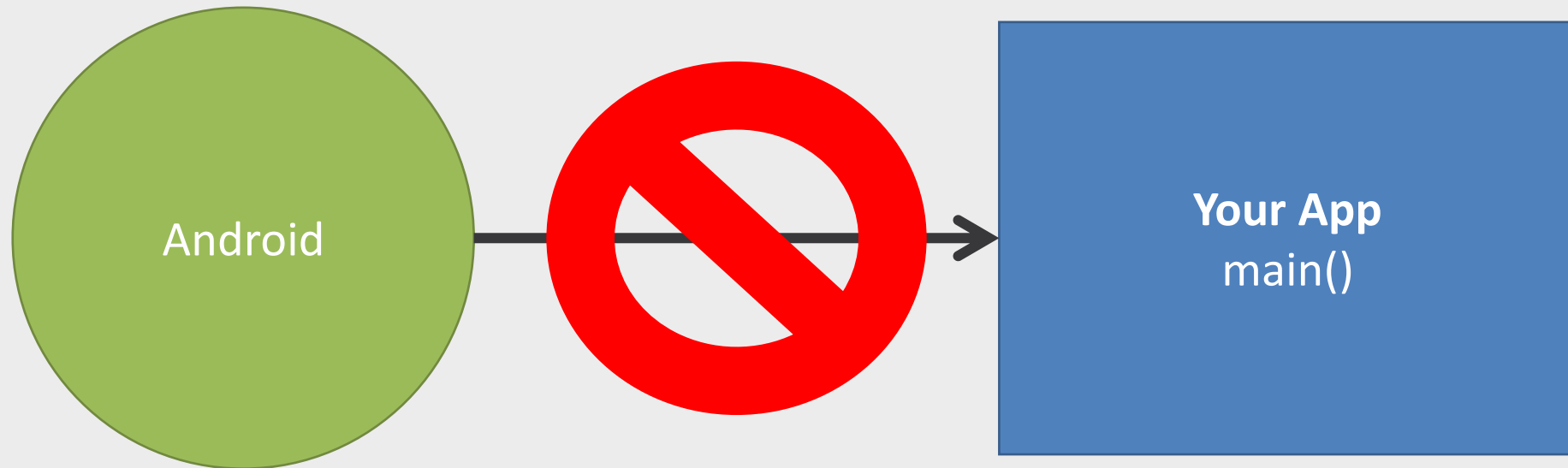
## App Basics – Projekt Aufbau

# Paixon



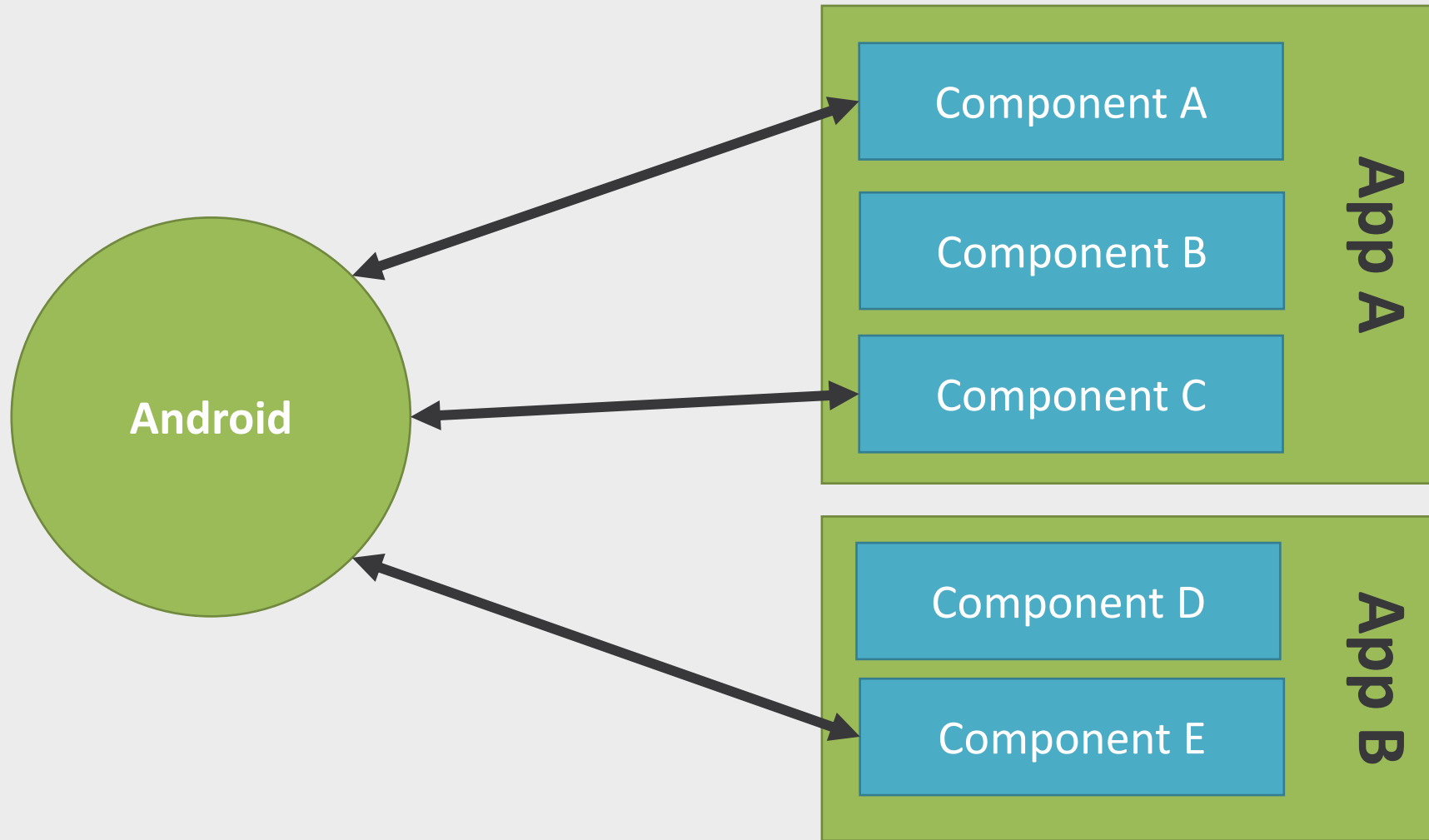
Component driven

# Paixon



Component driven

# Paixon



## Components

Activity

UI

Service

Background

ContentProvider

Provide data

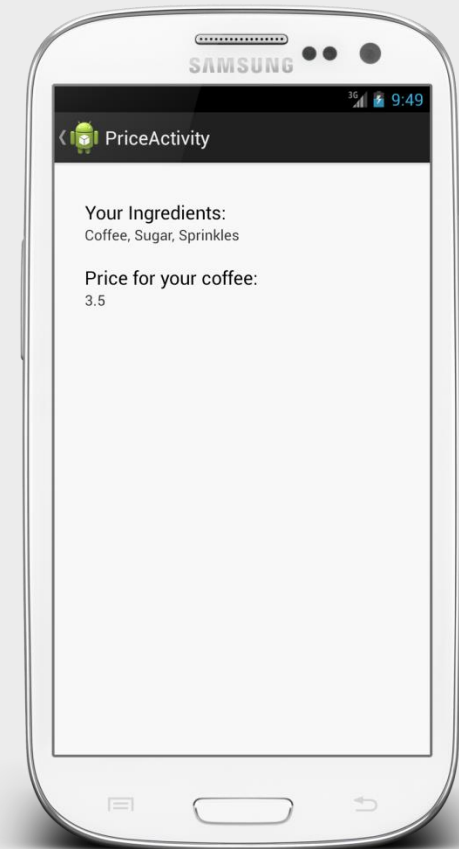
BroadcastReceiver

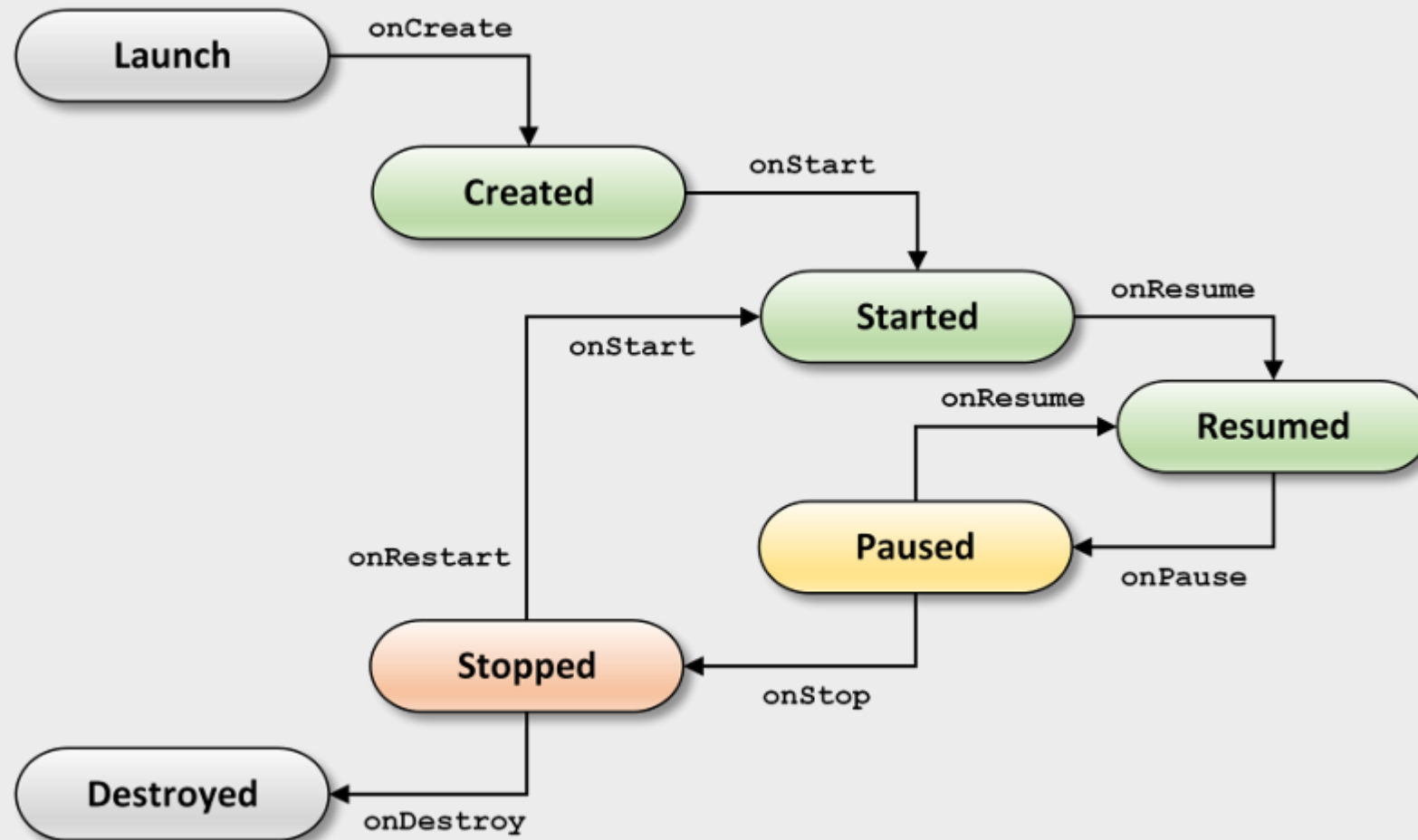
Get notified about events

Activity



- Class extending *Activity* (AppCompatActivity)
- Represents a single screen a user can interact with
- Userinterface defined in a xml file or direct in code
- Application normally has multiple activities
- Activity lifecycle





**onCreate()** Called when the activity is first created. This is where you should do all of your normal static set up: create views, bind data to lists, etc.

**onRestart()** Called after your activity has been stopped, prior to it being started again.

**onStart()** Called when the activity is becoming visible to the user.

**onResume()** Called when the activity will start interacting with the user. At this point your activity is at the top of the activity stack, with user input going to it.

**onPause()** Called when the system is about to start resuming a previous activity. This is typically used to commit unsaved changes to persistent data, stop animations and other things that may be consuming CPU, etc. Implementations of this method must be very quick because the next activity will not be resumed until this method returns.

**onStop()** Called when the activity is no longer visible to the user, because another activity has been resumed and is covering this one. This may happen either because a new activity is being started, an existing one is being brought in front of this one, or this one is being destroyed.

**onDestroy()** The final call you receive before your activity is destroyed. This can happen either because the activity is finishing (someone called `finish()` on it, or because the system is temporarily destroying this instance of the activity to save space.

# DEMO

Lifecycle methods

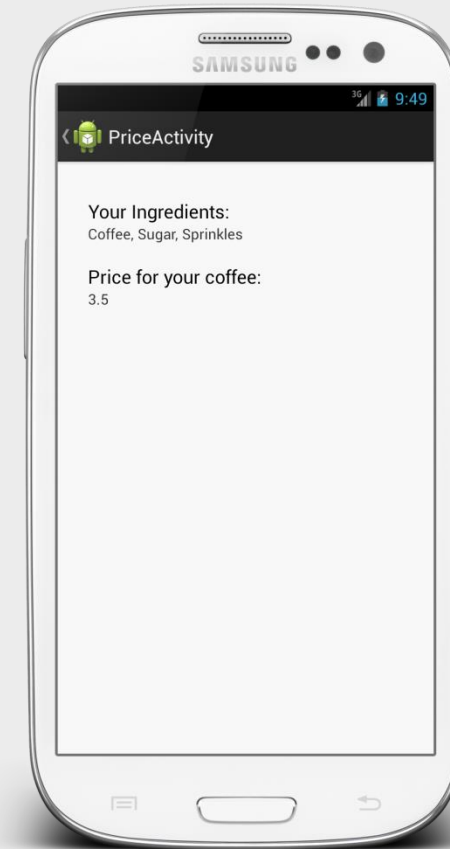
### **PriceActivity.java**

- Extend Activity class
- Use setContentView(...) to connect to XML
- Use findViewById(...) to access view



### **activity\_price.xml**

- Declarative xml to define UI
- Define ID's for every view



## activity\_price.xml

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <Button
        android:id="@+id/demoButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="MyButton" />

</RelativeLayout>
```

Define id using @+id/yourId

Set layout for activity:

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

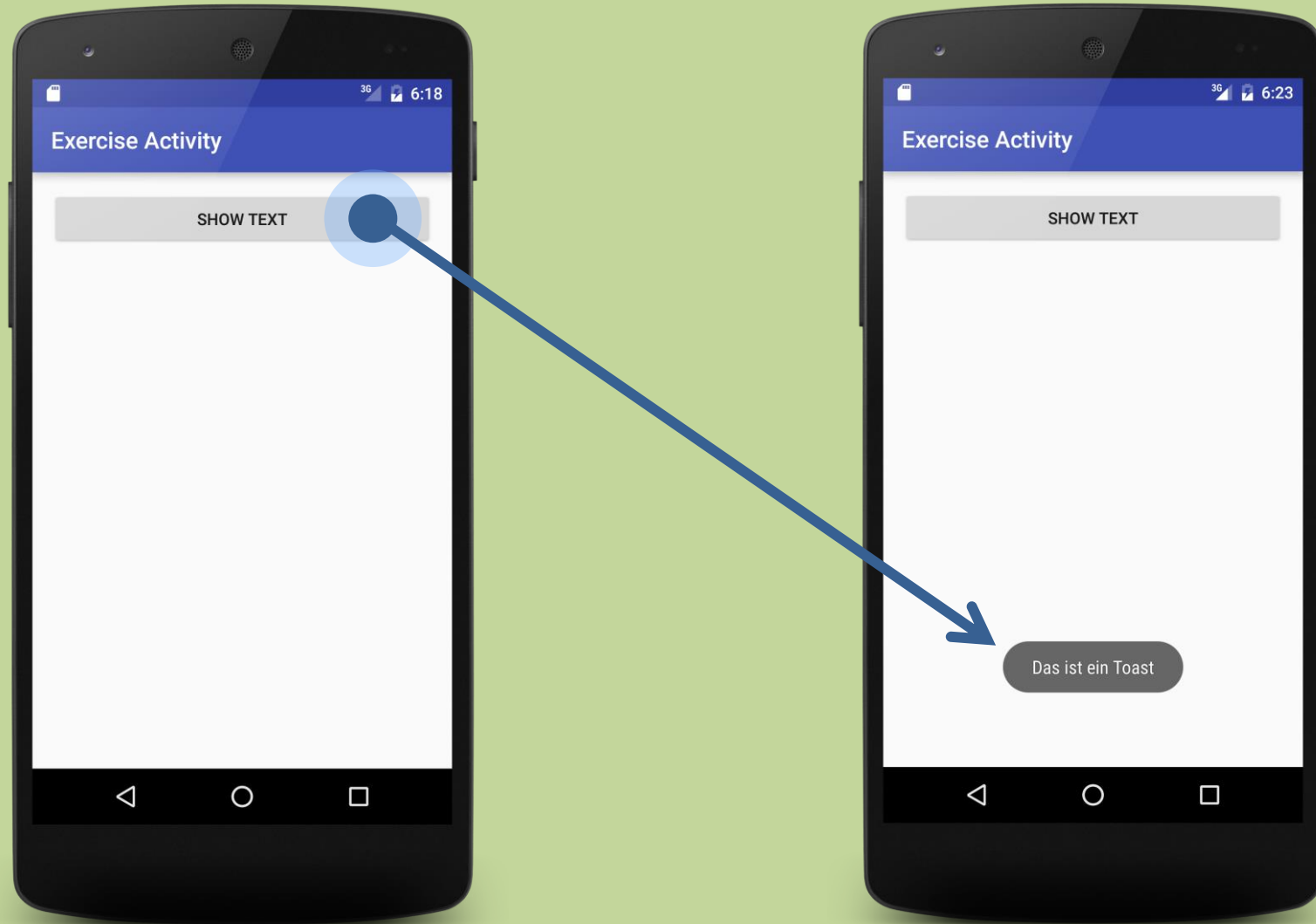
Access views in activity:

```
Button demoButton = (Button) findViewById(R.id.demoButton);
```

- Automatically generated
- Contains all resources ID's from your res-folder as public constants
- Subclasses for all Resources-Types:
  - R.drawable / R.layouts / R.id / R.string / ...
- Used to access resources in code

```
public final class R {  
    public static final class string {  
        public static final int action_settings=0x7f050001;  
        public static final int app_name=0x7f050000;  
        public static final int hello_world=0x7f050002;  
    }  
}
```





## Aufgaben:

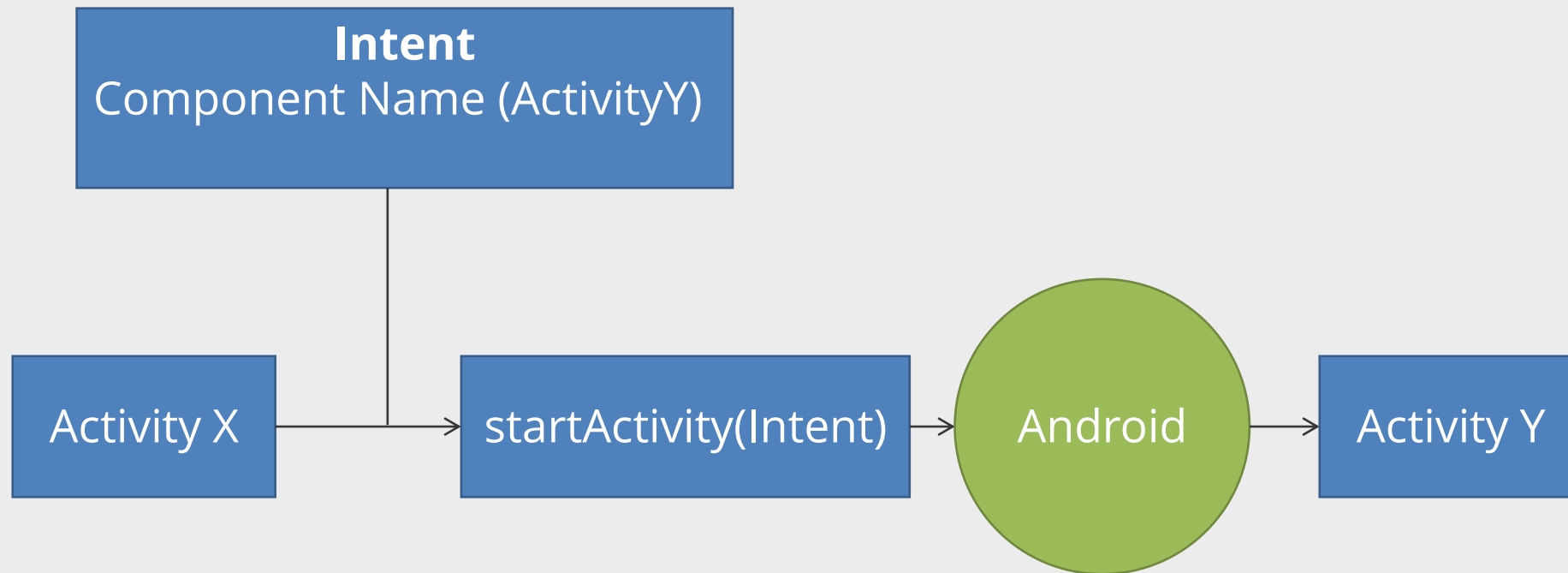
1. Verstehe den Aufbau der Android App
  - Wo werden die Layout-files gespeichert?
  - Wie wird eine Activity definiert?
  - Für was ist das AndroidManifest.xml?
  - Wie können Layout und Activity-Klasse verbunden werden?
  - Wie können die Activity-Lifecycle-Methoden verwendet werden?
2. Stelle sicher, dass wenn der Benutzer auf den "Show text"-Button klickt der Text "Button clicked" als "Toast" angezeigt wird.
  - Auf dem Button kann ein OnClickListener gesetzt werden
  - Ein Toast kann mit Hilfer der «Toast»-Klasse erstellt werden

## Projekt: Exercise\_Activity

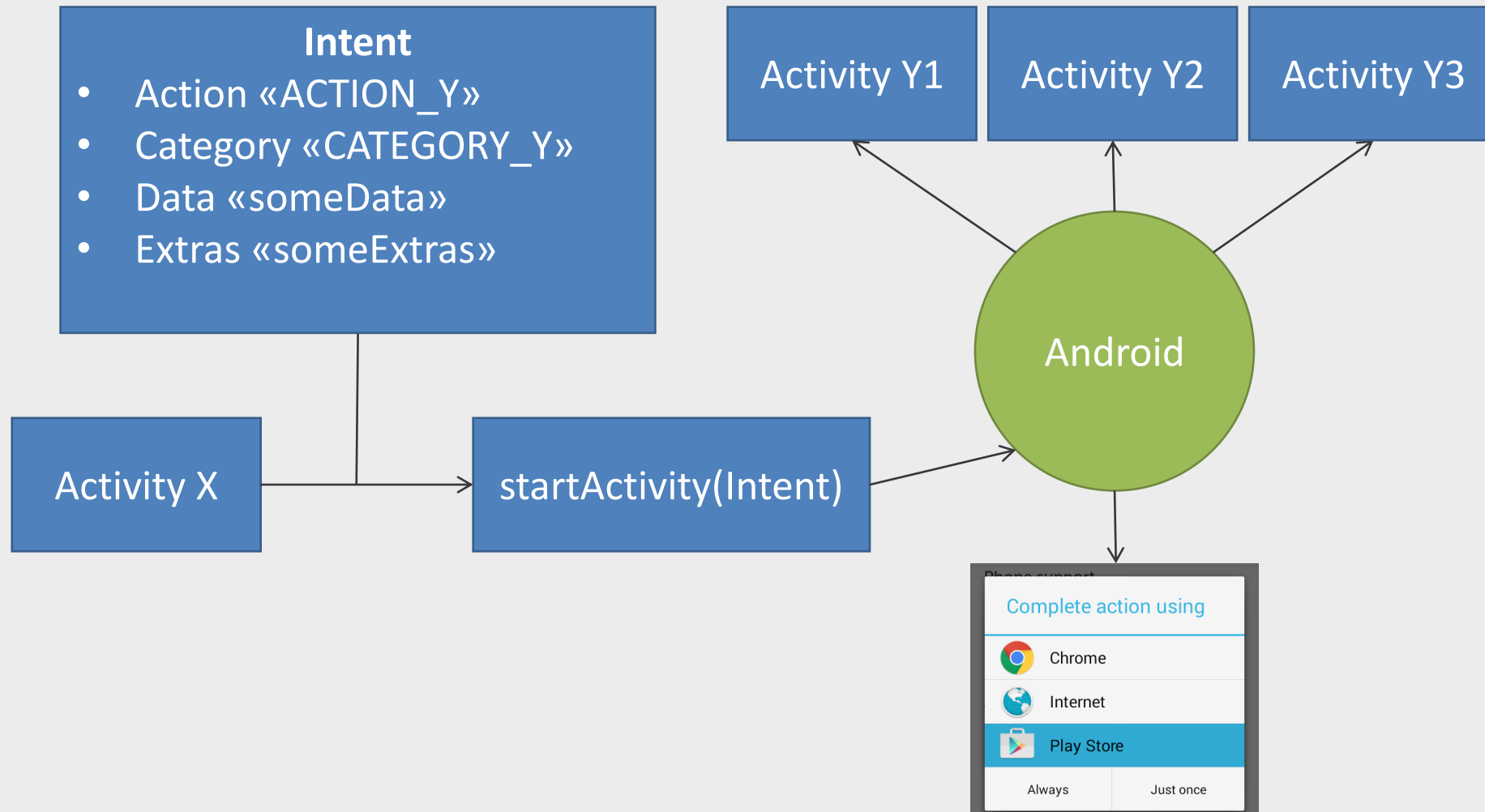
Connect multiple Activities

- Message to start another component
- Two types of intents available
  - Explicit Intents
  - Implicit Intents
- Contains data for the target Component
  - Component name, action, category, extras, flags

Target component can be defined in a different application!



## Connect multiple Activities – Implicit Intent



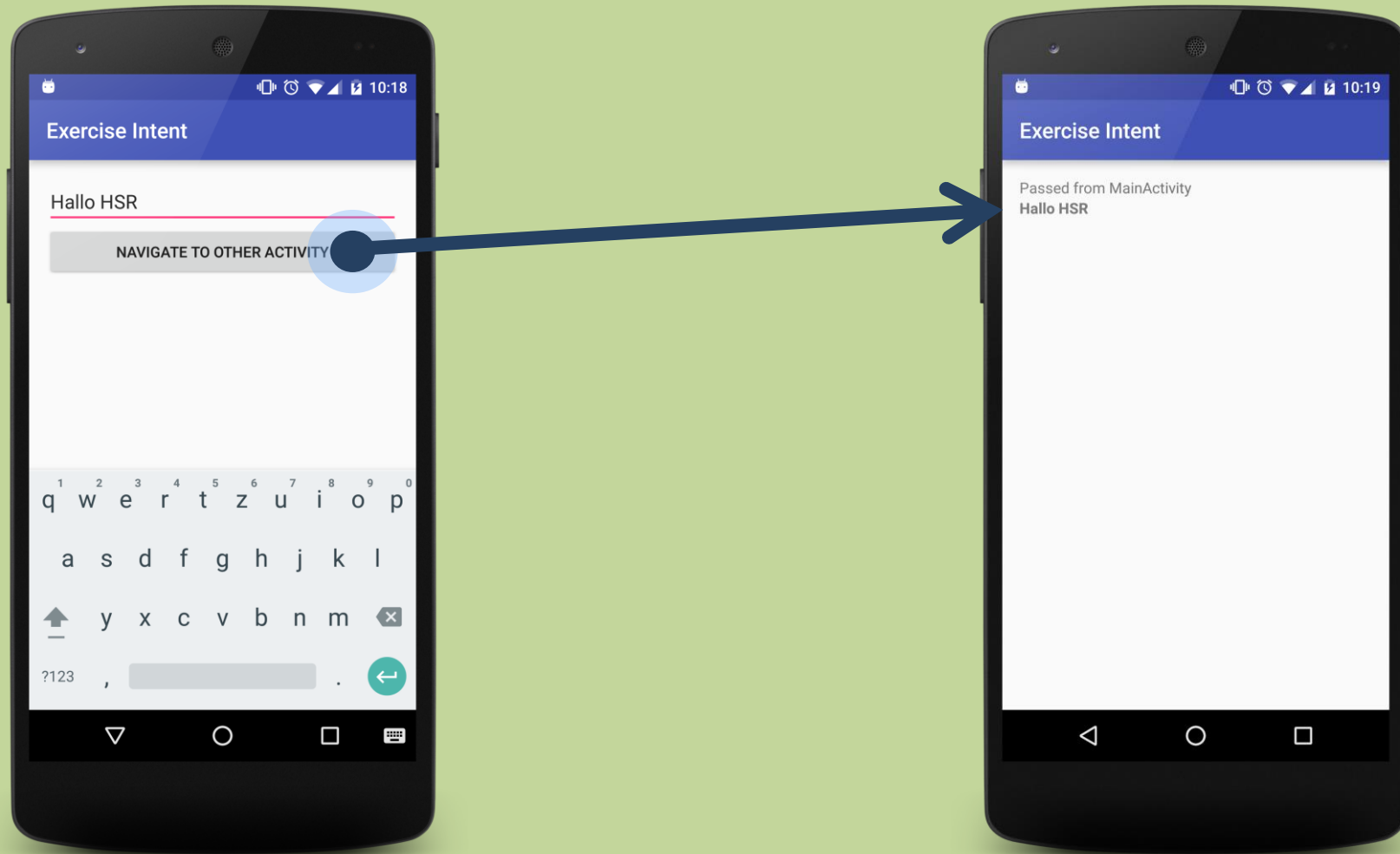
```
// Intent erstellen (this = Context)
Intent intent = new Intent(this, TargetActivity.class);

// Daten in den Intent packen für die neue Activity
// Es können nur Value Types (int, string, long, ... ) übergeben werden
intent.putExtra("MyKey", "Die Daten");

// Neue Activity durch Android starten
startActivity(intent);
```

```
// In der onCreate()-Methode der TargetActivity:
// Intent in TargetActivity abfragen
Intent intent = getIntent();

// Extra Daten abfragen
String data = intent.getStringExtra("MyKey");
```





## Aufgaben:

1. Verstehe das Konzept hinter den loose gekoppelten Komponenten
2. Verstehe das Konzept eines Intents
  - Wie wird ein Intent erstellt?
  - Wie kann ich einem Intent Daten mitgeben? (Extras)
3. Stelle sicher, dass wenn auf den “Navigate”-Button geklickt wird der Text aus dem EditText an eine neue Activity mitgegeben und dort angezeigt wird.

## Projekt: Exercise\_Intent

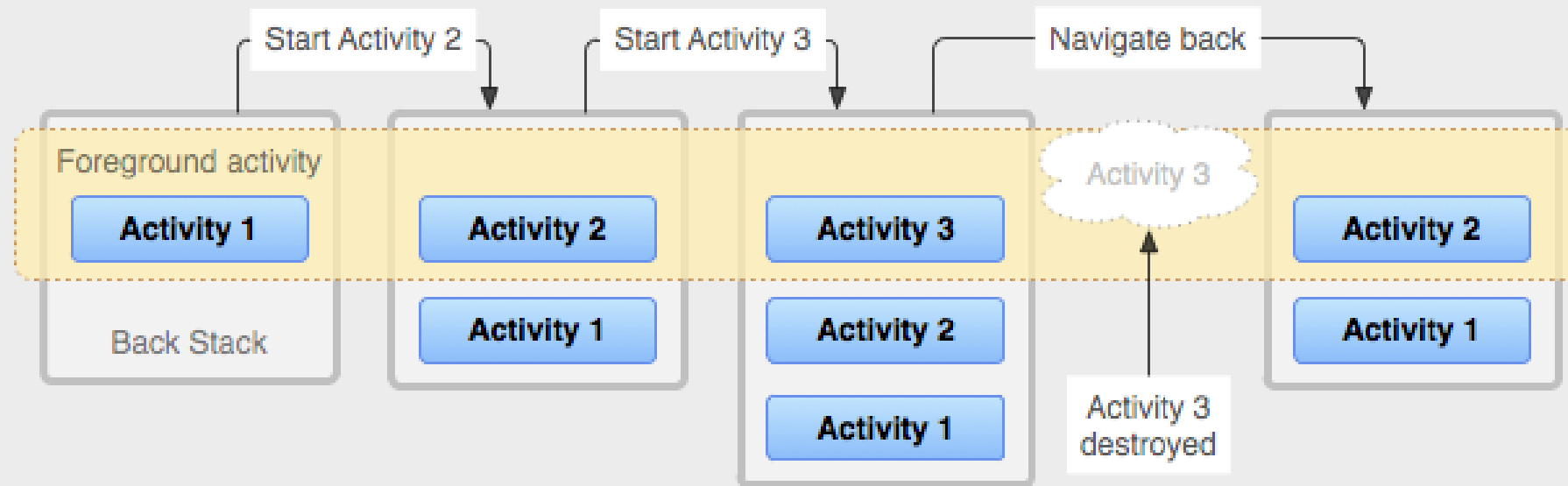
## Task and Back Stack

### **Task**

A task is a collection of activities that users interact with when performing a certain job.

### **Back Stack**

The activities are arranged in a stack (the "back stack"), in the order in which each activity is opened.



## Resources

- Split resources from code
  - Layout/strings/images/...
- Resources are all kept in a “res”-folder
- Different resources for different languages or screen sizes
- Default/Fallback-Resources
  - When not found → App crash (runtime)

### **drawable**

Everything that has something todo with graphics (images, xml)

### **layout**

Layout files for your activities, list items, ...

### **menu**

Menu definitions

### **values**

Simple values like strings for translations

### **mipmap**

Launcher Icon

## Resources – Providing alternative Resources

Same folder structure as default resources

Additional qualifiers

```
<resourcesfolder-name>-<qualifier>[-<qualifier>]...
```

Same filename as default resource

```
res/  
  drawable/  
    icon.png  
    background.png  
  drawable-hdpi/  
    icon.png  
    background.png
```

Default

HDPI Devices



### Qualifier Examples

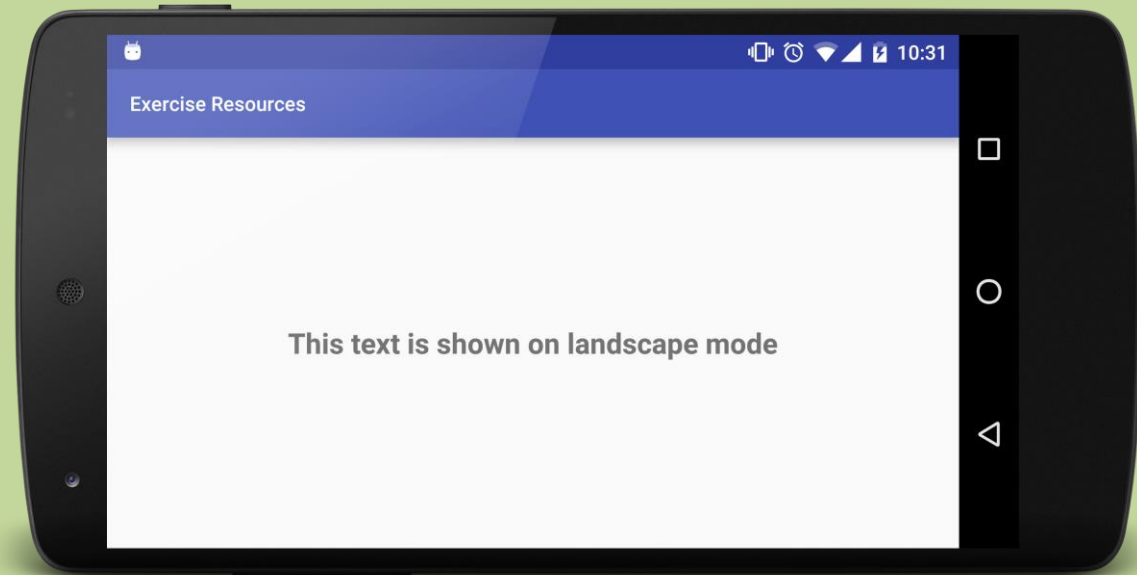
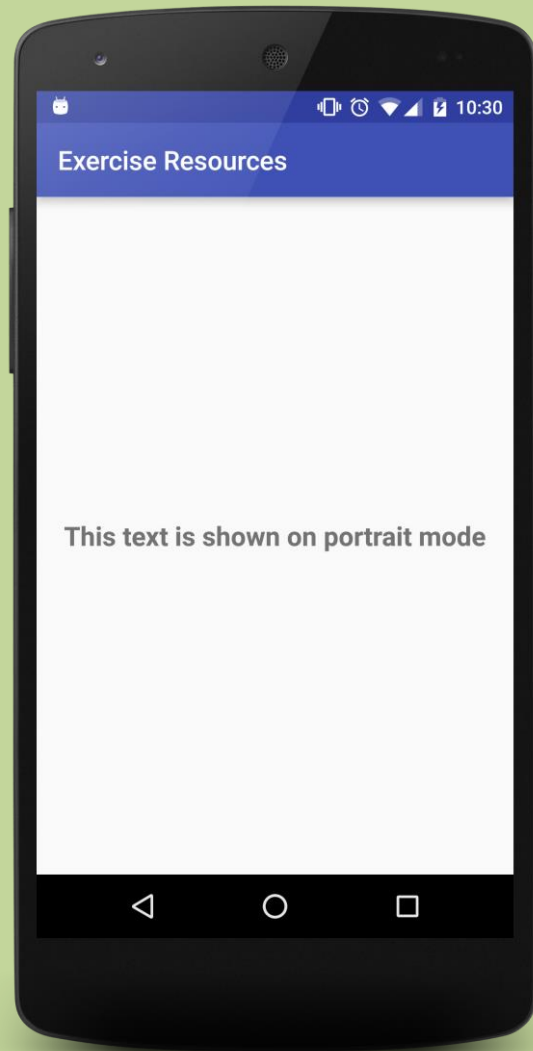
Language / Region	values-en ; values-de-CH
Screen Size	layout-small ; layout-large
Screen Orientation	layout-land ; layout-port
Screen pixel density (dpi)	layout-mdpi ; layout-hdpi ; drawable-hdpi
Platform version (API Level)	values-v10 ; layout-v8

### Other qualifiers

Mobile Country Code (MCC); Layout Direction; Smallest Width; Available Width; Available Height; Screen aspect; UI Mode; Night Mode; Touchscreen type; Keyboard availability; Primary text input method; Navigation key availability; Primary non touch navigation method

### Qualifier Rules

- Multiple qualifiers for single resource possible  
For example: layout-de-CH-hdpi
- Order of chained qualifiers is important  
*<http://developer.android.com/guide/topics/resources/providing-resources.html#AlternativeResources>*
- Case insensitive
- Multiple qualifier of same type not supported  
For example: “layout-hdpi-mdpi” is not allowed



**Aufgaben:**

1. Verstehe wie Android die korrekte Resource lädt
2. Verstehe warum und wann du Ressourcen einsetzen musst
3. Stelle sicher dass die Demo-App in der Landscape-Ansicht einen anderen Text darstellt als im Portrait-Mode

**Projekt: Exercise\_Resources**