Hello Android

CS 236603
Dr. Oren Mishali
Based on the Official Android Development Guide

About the course(s)

• 236503, 3 points
• 236603, 2 points

Grade based on your work and project’s quality

Final exam, based on the lectures
About the lectures

• Android development (coding)
• Android design (screens & navigation)
• Agile software development
  • Scrum process
  • Relevant XP practices

Final exam will be based on these 3 topics

About Android

• A mobile operating system developed by Google
• The most popular mobile OS
• Based on the Linux kernel
• Designed primarily for smartphones and tablets
• Further developed for Android TV, Android Auto, and Android Wear
Android version history

- Android’s versions evolve rapidly
- Current version is Android 9.0 (“Pie”)
- Each version is associated with an API level

**Pie has API level 28**

*Release notes*

Why is the “versions” issue so important?
Since we want our apps to support as many devices as possible.

How Android SHOULD NOT Be Studied

- Do a first “Hello World” example
- Then ask fellows from **StackOverflow**...

This is a great site but you should first be familiar with the key official guides.
A better approach would be...

https://developer.android.com/docs/

Documentation for app developers

Whether you’re building for Android handsets, Wear OS by Google, Android TV, Android Auto, or Android Things, this section provides the guides and API reference you need.

And other Resources, e.g.,

- CODEPATH
- AndroidHive
- Vogella

Our development tools

- Android Studio
- GitHub
- Firebase
- Development Environment
- Version Control, Project Management
- Backend
Android Studio

- The official android development environment
- Replaces Eclipse
- Based on the IntelliJ IDEA
- Operates on top of Android SDK tools

Java JDK should be installed before installing Android Studio

Make sure you install Android Studio 3

Staying up-to-date

- Update Android Studio and SDK tools regularly
- Open the SDK Manager to update the Android SDK and get new packages
Kotlin

• “Kotlin is expressive, concise, extensible, powerful, and a joy to read and write”
• It's easy to get started because it works side by side with Java and on Android
• Starting with Android Studio 3.0, tooling support for Kotlin is bundled directly into Android Studio

We will continue to use Java as the official course language

An Android Project

• Contains all files and resources that comprise an Android app
• A project is built into a single apk file that is installed on a device
Running your app on a real device

- Enable **USB debugging** and connect your device
- “No Connected Device”? might need to **install a driver**
- For Samsung devices, installing **KIES** may do the job
- Select the project and click **Run**
Android Emulator

• The Android Emulator **simulates a device** and displays it on your development computer

• The emulator supports various devices such as Android phone, tablet, Android Wear, and Android TV

  - It may take time for the emulator to initially show up
  - An alternative (non-official) emulator is [GenyMotion](https://developer.android.com/studio/run/emulator.html)


Android emulator: change BIOS settings

• We got this error while trying to initially operate the Android Emulator

• The error means that we should modify some BIOS settings

• So we entered BIOS => Security, and enabled both VT-d feature, and Intel Virtualization Technology
An Android Screen (Activity)

- An application component with which users can interact in order to do something
- E.g., dial the phone, take a photo, send an email, view a map
- In Android, a screen is mapped to an Activity component that has
  - A corresponding layout XML file
  - A corresponding Java file

AndroidManifest.xml

We’ll now review key project’s components
**build.gradle**

- **Gradle** is used to compile and build the app
- A **build.gradle** for each module and for the entire project
- Usually we’ll use the **module’s** build.gradle

A **module** is a container for your app’s source code. Usually we’ll use a project with one module however additional modules may be added, e.g., to support new devices such as Android Wear or Android TV. Read more.

**Resources (res) folder**

- drawable-<density>/
- layout/
- menu/
-ipmap/
- values/

**res/** is a commonly used directory that we’ll explore later on
App color customization

• App colors can be customized in *styles.xml* and *colors.xml* files
• There are additional color attributes we’ll cover later on
• Use e.g. [this site](http://developer.android.com/training/basics/firstapp/) to choose a color palette that you like

Building a Simple User Interface

• The task
  • Main activity with a text field and a button
  • Pressing the button displays the text in a **second** activity

The tutorial has changed and now uses the GUI builder. We will learn the classical “text-based” approach

http://developer.android.com/training/basics/firstapp/
Graphical User Interface of an Activity

• Hierarchy of View and ViewGroup objects
  • View – e.g., buttons, text fields
  • ViewGroup – containers defining the layout, e.g., LinearLayout
• Android UI is mostly defined in XML

Linear Layout

• LinearLayout is a subclass of ViewGroup
• Lays out child views in either a vertical or horizontal orientation

```xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:orientation="horizontal"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  app:layout_behavior="@string/appbar_scrolling_view_behavior"
  tools:showIn="@layout/activity_my">
  
</LinearLayout>
```
Add a text field (EditText)

```xml
<EditText android:id="@+id/edit_message"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:hint="@string/edit_message" />
```

- What is the meaning of `@string/edit_message`?
- How to resolve the compilation error?

---

String Resources

- **Always** specify **UI strings** as resources
  - Allow to manage all UI text in a single location
  - Straightforward **localization**
Add a Button

• Why no `android:id` for the button?
• Should the resulting layout be improved?

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_behavior="android.support.design.widget.TintCubeBehavior"
    tools:showIn="@layout/activity_my">

    <EditText android:id="@+id/edit_message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="Enter a message" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_send" />

</LinearLayout>
```

Improved Layout

• Note the use of `0dp` in `layout_width`
• Improved performance since calculation of `wrap_content` is not needed

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res-auto"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:hint="Enter a message" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_send" />

</LinearLayout>
```
Starting Another Activity – *onClick* event

• **The task:** start a new activity when the user presses the button

```xml
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_send"
    android:onClick="sendMessage" />
</LinearLayout>
```

Creating a Java Method Handler

• The method’s signature must be
  • public
  • void
  • single **View** Parameter

```java
/** Called when the user clicks the Send button */
public void sendMessage(View view) {
    // Do something in response to button
}
```
Building an Intent

```java
/** Called when the user clicks the Send button */
public void sendMessage(View view) {
    Intent intent = new Intent(this, DisplayMessageActivity.class);
    EditText editText = (EditText) findViewById(R.id.edit_message);
    String message = editText.getText().toString();
    intent.putExtra(EXTRA_MESSAGE, message);
    startActivity(intent);
}
```

Note the use of `this` as first `Context` parameter. Second parameter is the class of the target Activity.

An Intent can carry data types as key-value pairs called `extras`

```java
public static final String EXTRA_MESSAGE = "com.mycompany.myfirstapp.MESSAGE";
```

Creating the Second Activity

- **Hierarchical Parent** adds Up Navigation (API 16+)
- The `<meta-data>` element adds support for older versions
Java code of the second activity

- Every Activity is invoked by an Intent
- Note how the layout is created programmatically

```java
public class DisplayMessageActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_display_message);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        FloatingActionButton fab = findViewById(R.id.fab);
        fab.setOnClickListener(v -> {
            Snackbar.make(v, "Replace with your own action", Snackbar.LENGTH_SHORT)
                .setAction("Action", null).show();
        });
        fab.setSupportProgressBar indirect(true);
        Intent intent = getIntent();
        String message = intent.getStringExtra(MyActivity.EXTRA_MESSAGE);
        TextView textView = findViewById(android.R.id.textView);
        textView.setText(message);
        RelativeLayout layout = (RelativeLayout) findViewById(R.id.content);
        layout.addView(textView);
    }
}
```

Layout of the second activity

No sub-elements since the layout is built programmatically
Relative Layout

- A very powerful utility for designing a user interface
- By default, all child views are drawn at the top-left of the layout
- The desired position of each view is defined using the various layout properties
- Views are positioned relative to other view and to the parent view

See `RelativeLayout.LayoutParams` for all attributes available

Getting a Result from an Activity

- You can start another activity and receive a result back
- You can get result from your own activities or from other apps
- The basic flow
  - ActivityOne calls `startActivityForResult()`
  - ActivityTwo sends the result as another `Intent` object
  - ActivityOne receives it in the `onActivityResult()` callback

For example, your app can start a camera app and receive the captured photo as a result

https://developer.android.com/training/basics/intents/result.html
Example: ActivityOne starts ActivityTwo

```java
public void onClick(View view) {
    Intent i = new Intent(this, ActivityTwo.class);
    i.putExtra("Value1", "This value one for ActivityTwo");
    i.putExtra("Value2", "This value two ActivityTwo");
    // set the request code to any code you like,
    // you can identify the callback via this code
    startActivityForResult(i, REQUEST_CODE);
}
```

- You can specify a **request code** to determine which activity was started (if multiple activities may be started)

---

**ActivityTwo**

```java
@override
public void finish() {
    // Prepare data intent
    Intent data = new Intent();
    data.putExtra("returnKey1", "Swinging on a star. ");
    data.putExtra("returnKey2", "You could be better then you are. ");
    // Activity finished ok, return the data
    setResult(RESULT_OK, data);
    super.finish();
}
```

An activity can be closed via the back button of the phone. In this case the `finish()` method is performed.

RESULT_OK and RESULT_CANCELED are constants in Activity class.

Once the activity finishes, the `onActivityResult()` method in the calling activity is called.
ActivityOne handles the result

```java
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (resultCode == RESULT_OK && requestCode == REQUEST_CODE) {
        if (data.hasExtra("returnKey1")) {
            Toast.makeText(this,
                data.getExtras().getString("returnKey1"),
                Toast.LENGTH_SHORT).show();
        }
    }
}
```