Software Engineering Large Practical: Android concepts and programming

Stephen Gilmore
School of Informatics
September 26, 2017
1. Android platform
2. Android concepts
3. Android projects
4. Android Studio
Android platform
Android is an open source, Linux-based software stack.

The Android Runtime (ART) relies on the Linux kernel for functionalities such as threading and memory management.

ART is written to run multiple virtual machines on low-memory devices by executing DEX files, a bytecode format designed specially for Android.

The hardware abstraction layer (HAL) provides interfaces that expose device capabilities to the higher-level Java API framework.

Many core Android system components and services, such as ART and HAL, are built from native code that require native libraries written in C and C++. 
Android developer concerns

• For an Android developer trying to produce a commercial app, producing an app which runs on older versions of Android would be a significant concern, because every user on an older version of Android who cannot use the app represents lost income.

• Although it is an important concern for commercial developers, in this practical *backwards compatibility has not been identified as an issue*. We do not care if your app does not work on older versions of Android.
Android concepts
Activities and contexts

- An Android app is split up into a number of different activities, which are subclasses of `android.app.Activity`, or subclasses of that class, such as `android.support.v7.app.AppCompatActivity`.

```
java.lang.Object
  ↓ android.content.Context (abstract class)
  ↓ android.content.ContextWrapper
  ↓ android.view.ContextThemeWrapper
  ↓ android.app.Activity
```

- An activity represents a single screen with a user interface.
- One activity can invoke another. Every `Activity` is a `Context`. 
Android activities

- Activities differ in nature from the main class of a Java application, in that it must be possible to pause, suspend, and resume them and have the app take action depending on which of these events happens.

- The allowable calls to methods such as
  - `onCreate()`,
  - `onStart()`,
  - `onResume()`,
  - `onPause()`,
  - `onStop()`,
  - `onRestart()`, and
  - `onDestroy()`.

make up the Android activity lifecycle.
Sample `onCreate` method — create UI components

```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main); // load res/layout/activity_main.xml

    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
    fab.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                .setAction("Action", null).show();
        }
    });
}
```
• Android projects separate application logic (coded in Java) from the user interface presentation layer (coded in XML).

• This separation of concepts means that the application logic does not get cluttered with presentation layer details about fonts, colours and positions of buttons in the user interface.
Sample button definition in Java and XML

**MainActivity.java**

```java
FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
```

**res/layout/activity_main.xml**

```xml
<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="@dimen/fab_margin"
    android:tint="@android:color/white"
    app:srcCompat="@android:drawable/ic_input_add" />
```
Android Activity lifecycle

From https://developer.android.com/training/basics/activity-lifecycle/starting.html
Android Activity lifecycle (create)

1. onCreate()
2. onStart()
3. onResume()

From https://developer.android.com/training/basics/activity-lifecycle/starting.html
Android Activity lifecycle (paused)

From https://developer.android.com/training/basics/activity-lifecycle/pausing.html
Android Activity lifecycle (stopping)

From https://developer.android.com/training/basics/activity-lifecycle/stoping.html
Android Activity lifecycle (saving state)

From https://developer.android.com/training/basics/activity-lifecycle/recreating.html
Adding a new Activity

• Most apps have more than one Activity.
• Adding a new Activity (with File → New → Activity):
  • adds a new Java class file,
  • adds a new XML layout file,
  • add the required <activity> element in AndroidManifest.xml,
and may add other files as needed for specific types of activity.
Using Intents

- An intent of `android.content.Intent` is a messaging object which can be used to communicate with another app component such as another Activity.

Image from http://www.vogella.com/tutorials/AndroidIntent/article.html
Using Intents

- You can start a new instance of an Activity by passing an Intent to startActivity().
- The Intent describes the activity to start and carries any necessary data.
- If a result is expected then startActivityForResult() is called instead.
- An Intent can also be used to start a Service of class android.app.Service.
Simple switch to another activity

private void switchToMap() {
    Intent intent = new Intent(this, MapsActivity.class);
    startActivity(intent);
}
One mechanism of activity starting another

public static final String EXTRA_MESSAGE = "com.example.myapp.MESSAGE";

public void sendMessage(View view) {
    Intent intent = new Intent(this, DisplayMessageActivity.class);
    EditText editText = (EditText) findViewById(R.id.editText);
    String message = editText.getText().toString();
    intent.putExtra(EXTRA_MESSAGE, message);
    startActivity(intent);
}

---

Credit: https://developer.android.com/training/basics/firstapp/starting-activity.html
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_display_message);

  // Get the Intent that started this activity and extract the string
  Intent intent = getIntent();
  String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);

  // Capture the layout's TextView and set the string as its text
  TextView textView = (TextView) findViewById(R.id.textView);
  textView.setText(message);
}

Credit: https://developer.android.com/training/basics/firstapp/starting-activity.html
Android projects
Android projects

- Android projects contain a mix of Java and XML code in a structured project which contains
  - **manifests**: Contains the `AndroidManifest.xml`, file which provides essential information about your app to the Android system, to allow it to run your code.
  - **java**: Contains the *Java source code files*, separated by package names, including *JUnit* test code.
  - **res**: Contains *all non-code resources*, such as XML layouts, UI strings, and bitmap images.
- Java code describing resources is automatically generated from XML source code by Android Studio.
Android build files

- In addition, Android projects also contain **build files** for compiling the project source code into an executable.
- Android uses the **Gradle** build system which specifies Android version requirements and app dependencies.

```java
... dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support:appcompat-v7:26.0+'
    compile 'com.android.support.constraint:constraint-layout:1.0.0-alpha9'
    compile 'com.android.support:design:26.0+'
    compile 'com.google.android.gms:play-services:11.0.4'
    testCompile 'junit:junit:4.12'
}
```
Android Studio
Android Studio

- Android Studio is the official Integrated Development Environment (IDE) for Android app development. It is based on JetBrains’s IntelliJ IDEA.
- Because it is an Android-specific development environment, Android Studio can make suggestions regarding issues such as missing import statements.

```
private void switchToMap() {
    Intent intent = new Intent(this, MapsActivity.class);
    startActivity(intent);
}
```

- A helpful introduction to Android Studio is available at [https://developer.android.com/studio/intro](https://developer.android.com/studio/intro)
Platform updates

• Android Studio and the Android APIs and device emulators are *active, current software projects*. It is quite usual when starting up Android Studio to see that updates are available for some of the components that you use.

![Platform and Plugin Updates](image)

• We recommend applying these as they become available.
Links

- http://www.oracle.com/technetwork/java/javase/downloads/ — to download Java 8 or Java 9
- YouTube Tutorial: Android Studio, from zero knowledge to something basic, Jonathan Warner, https://www.youtube.com/watch?v=-igAiudpBng