1. App Fundamentals

Understanding Android Applications

Library vs Framework

What is the difference between a library and a framework?

Libraries are called by your code.
Frameworks calls your code.

Android is a framework for building mobile applications.

Android Source

An android app has a very specific folder structure, with all code organized into a particular pattern:

- *src* - This is where all Java source files are located
- res/layout - XML defining view layouts
- res/drawable - Place to store images
- res/values - Strings, colors, etc
- assets - Place to store misc. static files (i.e text files)
- AndroidManifest.xml - Application-wide settings
- build.gradle - Build file (declare dependencies here)

Android Manifest

AndroidManifest.xml is in every android application and contains application-wide settings.

The manifest specifies:

- Package and application name
- What the application launches on startup
- The components and views of the application
- Permissions the app requires

build.gradle

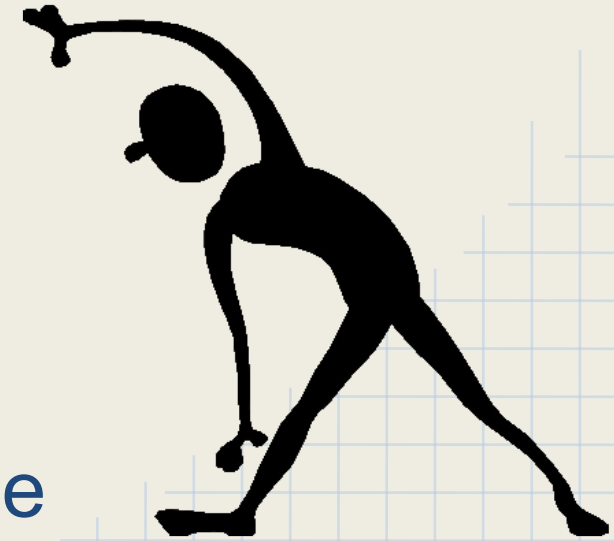
Gradle is the build system that comes with Android Studio. It's build settings are contained in a build.gradle file.

The build file specifies:

- Android specific build options (targetSdkVersion, etc)
- Remote library dependencies
- Version information for the app
- The version of android the app targets

Fundamentals Exercises, 1

- Open **HelloWorldDemo** in Android Studio
- Open the **app** build.gradle
  - Change the version code and name to 2 and 2.0
- Open the AndroidManifest.xml
  - Change the application icon
  - Change the application name



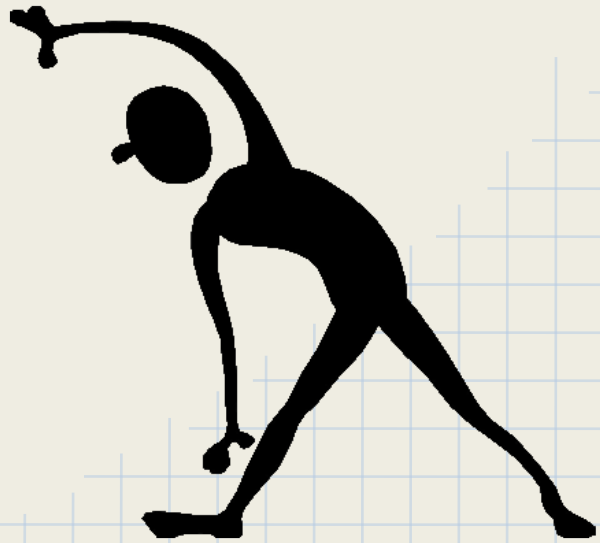
Activity

In Android, each full-screen within an application is called an **Activity**.

- An application can have one or more activities that make up the interaction flow.
- Most of the time in an android application is spent constructing the look and content of activities.
- Activities have at least two parts:
  - The Java source file in src/package/FooActivity.java
  - The XML layout in res/layout/foo_activity.xml
- Activities are each independent and do not **directly** communicate with each other.

Fundamentals Exercises, 2

- Open **HelloWorldDemo** in Android Studio
- Open res/layout/activity_hello_world.xml
- Copy and paste new TextView into XML
- Change `android:text` value



Resources

In Android, **Resources** include strings, images, colors, xml-based activity layouts, menu items, etc.

- In XML, resources are accessed by a special syntax
  - `@string/my_string` - references string in strings.xml
  - `@drawable/cool_image` - references image in drawable folder.
- In Java, at compile time the resources folders are inspected and a special class called R is **generated**.

```
R.string.my_string
```

String Resources

In Android, **Strings** are typically not hard-coded in your application but instead stored in **strings.xml**

- The strings.xml file is used to define a key "name" for the string and the value which is the text.
- You can access strings as "@string/some_name" (XML) or R.string.some_name (Java)

```
<resources>
    <string name="some_name">My String Text</string>
</resources>
```

Fundamentals Exercises, 3

- Open **HelloWorldDemo** in Android Studio
- Open res/layout/activity_hello_world.xml
- Replace `android:text` values with references to strings defined in strings.xml

Activity Lifecycle

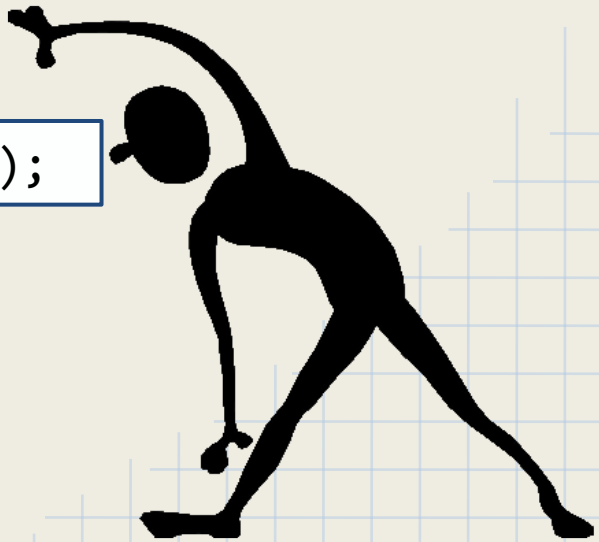
An Android activity transitions through various states as it is shown, hidden, and destroyed.

- Each state transition will call a method on the Activity.
- The three most important methods are:
  - onCreate - Called to create an activity. Usually sets the xml layout to use as the interface.
  - onPause - Called when leaving an activity. Usually where any needed data is stored for later.
  - onResume - Called when returning to an activity. Any stored data is restored here.

Fundamentals Exercises, 4

- Open **HelloWorldDemo** in Android Studio
- Open src/codepath/HelloWorldActivity.java
- Add a `Log.d` call to OnCreate, OnPause, and onResume

```
Log.d("TEST", "Activity has been started");
```



Size Units

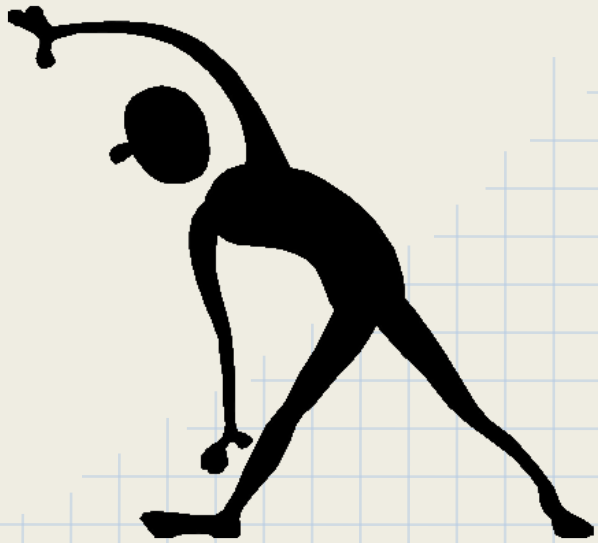
In order to support a variety of screen densities, you should use relative units instead of absolute units.

- The most common units within Android development are **dp** (density independent), and **sp** (scale independent).
- Rule of thumb: **sp** for text size, **dp** for everything else.
- Do NOT use px or pt.
- The sp units for fonts will adjust for **both** the **screen density** and user's system **font preference**.

Fundamentals Exercises, 5

- Open **HelloWorldDemo** in Android Studio
- Open res/layout/activity_hello_world.xml
- Switch textSize larger

```
<TextView
  android:textSize="50sp" />
```



Fundamentals Wrap-up

- Each Android project has several key folders including **res** and **src**.
- The **AndroidManifest**.xml is a file which stores all application settings and configuration.
- The **build.gradle** contains the build settings for the app.
- Every full-screen on the Android is called an **Activity**. An application typically has **multiple** activities.
- Android development involves access to many **resources** including strings, colors, layouts, et al.
- Activities have a **lifecycle** which can be managed using methods such as **onCreate**
- For position, use **dp** relative units. For text size, use **sp**.