ANDROID

Android Studio, App Basics,

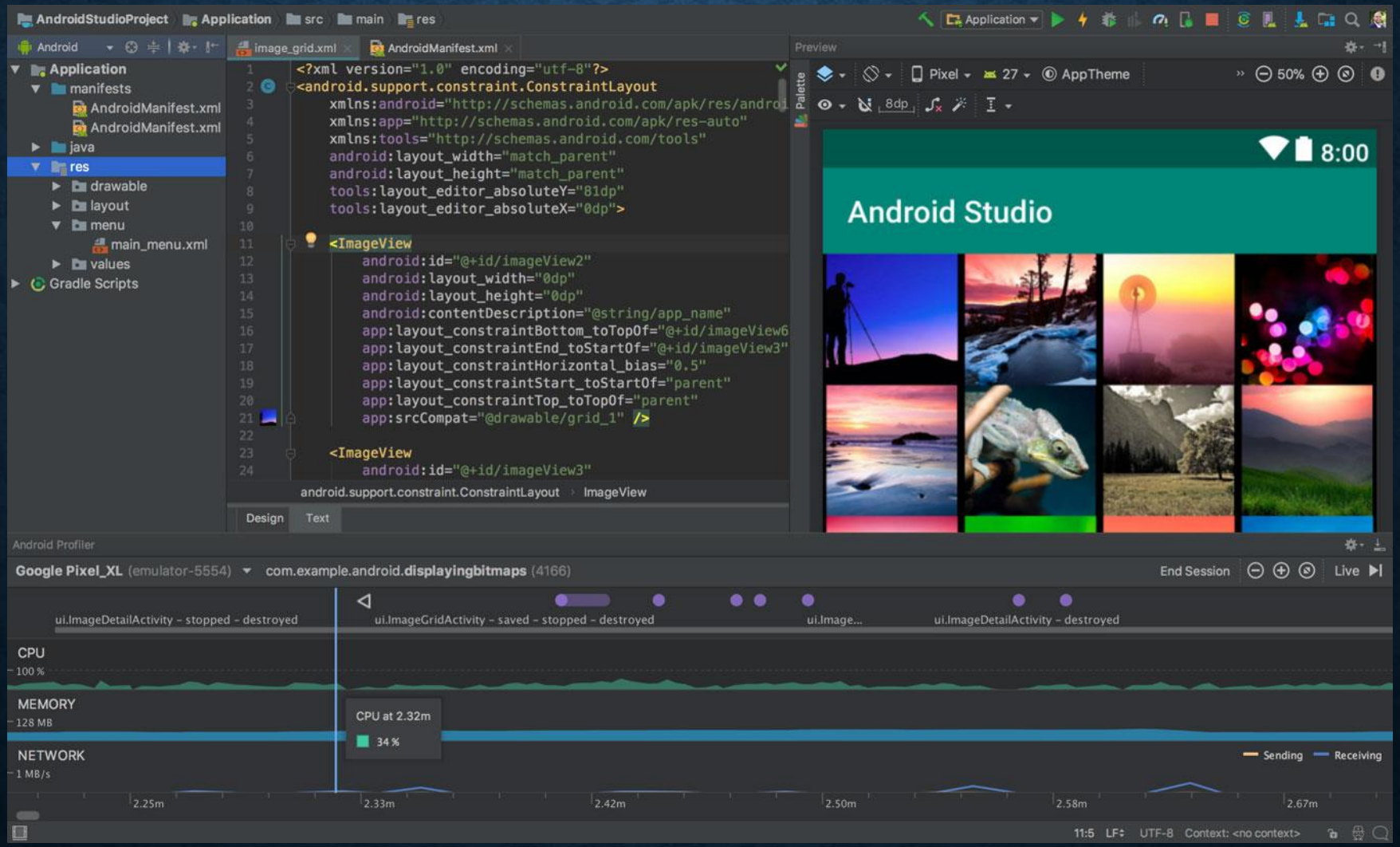
Events, Inputs, and Data Storage

COS216
AVINASH SINGH
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF PRETORIA

ANDROID STUDIO

- Android IDE with code editor, compiler, debugger, and emulator integration
- Based on IntelliJ originally designed by JetBrains
  - JetBrains has developed a new programming language: Kotlin
  - Runs on JVM, but can also be compiled into JavaScript
  - Fully supported by Android and Android Studio
- Android Studio written in Java and Kotlin
- Android apps written in Java with XML for the UI
  - Other languages are supported
  - Example: C/C++, Python, Kotlin

ANDROID STUDIO



ANDROID STUDIO

- Android Studio features
  - Code completion (text prediction and code completion)
  - Gradle-based build support (Apache Ant and Maven language replacing XML)
  - Lint tools (performance, usability, version compatibility, problem tracking)
  - ProGuard (sign your app with a certificate)
  - Template wizards (use existing templates for design and components)
  - UI design (layout editor, drag-and-drop UI, preview layouts)
  - Android Wear app support
  - Google support (Google Cloud Platform, Google App Engine, Firebase Cloud Messaging)
  - Android Virtual Device (emulator and debugging tools)

ANDROID STUDIO

- Project Build
  - Reuse most code for different devices
    - Phones, Tablets, TVs
    - Android Wear (smart watches)
    - Google Glasses (not sure, don't have a pair to test)
- Version Control
  - Integration with GitHub and other repositories
- Deploy to physical device
  - Requires a USB cable, Android Phone, and some tool, such as Android Debug Bridge (ADB)

ANDROID STUDIO

- Some additional info

- Android Studio

  https://developer.android.com/studio/features.html

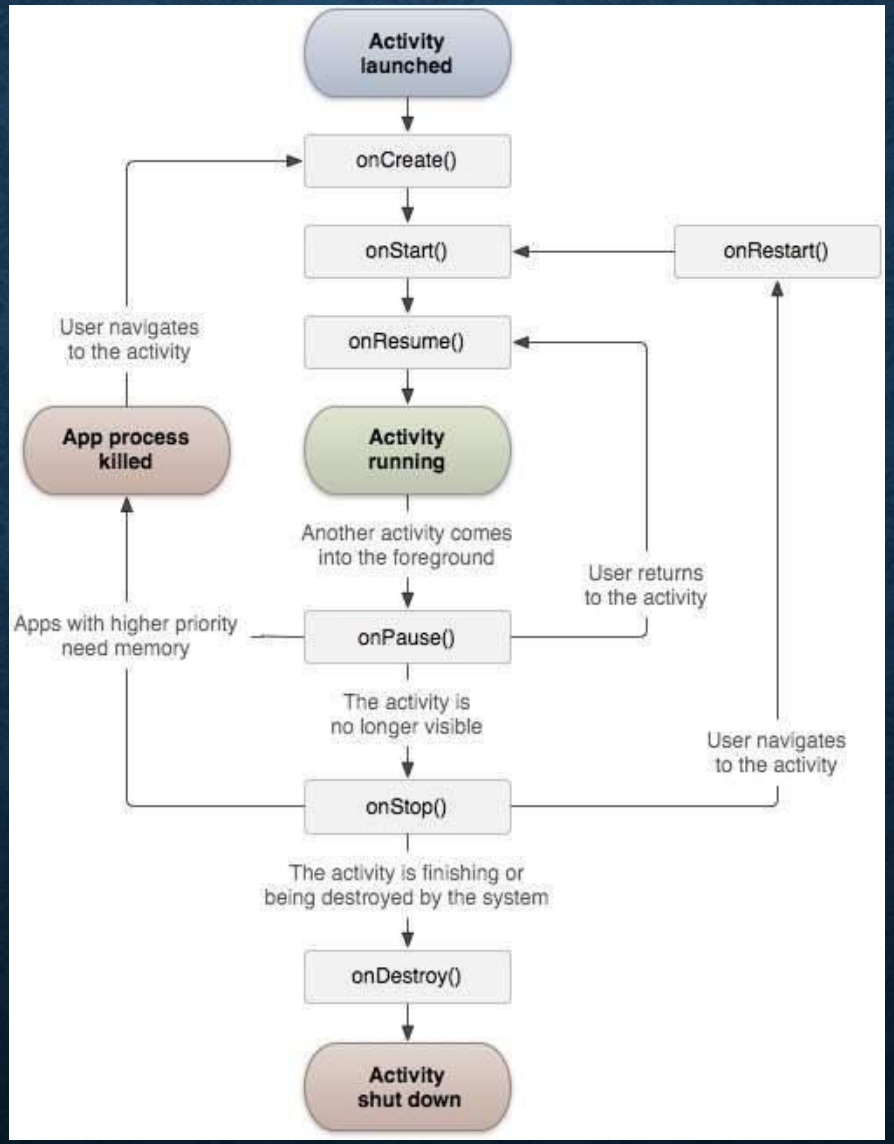
- Android Tutorials

  https://developer.android.com/training/index.html

ANDROID APP ID

- Choose an ID for your app
- Like with the standard Java convention
  - Choose a reversed domain name
  - You don't have to own the domain name, make one up, just make sure no one else uses it
  - Example: com.nordvpn.android or com.whatsapp
- Check your phone using a file manager for other IDs
  - <storage>/Android/data
  - Might differ between Android versions

ANDROID ACTIVITIES



ANDROID ACTIVITIES

- Activities are a “screen” seen by the user
- Apps can have multiple activities
- Apps can switch between activities, similar to different windows in a normal desktop program
- Main activity is the entry point for your app
  - Your main logic should be added here
  - app > java > com.example.myfirstapp > MainActivity.java

ANDROID LAYOUTS

- Activities can have a layout
- Defines how the activity looks like to the user
- Layouts consist of a hierarchy of View and ViewGroup
  - Written in XML
- Each activity has a separate layout
  - app > res > layout > activity_main.xml

ANDROID MANIFEST

- Defines the key characteristics of your app
  - App ID and name
  - Icons and labels
  - Intent filters
  - Permissions
  - Device compatibility
  - App components with activities, services, receivers, and providers
- Written in XML
- Edit the manifest at
  - app > manifests > AndroidManifest.xml

ANDROID MANIFEST

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

          <uses-sdk android:minSdkVersion="15" android:targetSdkVersion="26" />
          <application android:allowBackup="true" android:icon="@mipmap/ic_launcher“
                    android:roundIcon="@mipmap/ic_launcher_round" android:label="@string/app_name“
                    android:supportsRtl="true" android:theme="@style/AppTheme">

          <activity android:name=".MainActivity">
                    <intent-filter>
                              <action android:name="android.intent.action.MAIN" />
                              <category android:name="android.intent.category.LAUNCHER" />
                    </intent-filter>
          </activity>
          <activity android:name=".DisplayMessageActivity" android:parentActivityName=".MainActivity" />
          </application>
</manifest>
```

ANDROID STRINGS

- Ability to reuse strings and labels
  - Especially in the UI
- Typically used to abstract the labels from the design
  - Can have different labels set for different countries/languages
  - Can easily switch between these strings
  - Allows you app to be multi-language
- Written in XML
  - Can be changed in Android Studio or manually in XML file
- Edit the strings at
  - app > res > values > strings.xml

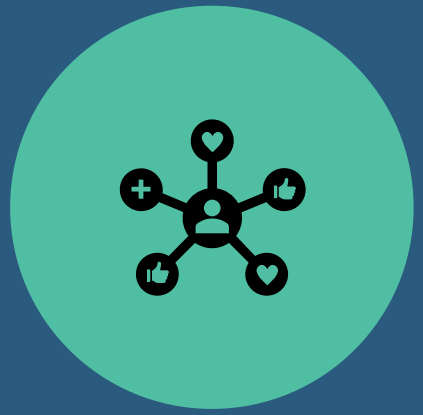
ANDROID STYLES

- Ability to change colours and styles
  - Especially in the UI
- Can have different themes for your app
  - Or different colours/styles for different devices
  - Similar to the idea of CSS for websites
- Written in XML
- Colours and styles are in two different files
  - app > res > values > colours.xml
  - app > res > values > styles.xml

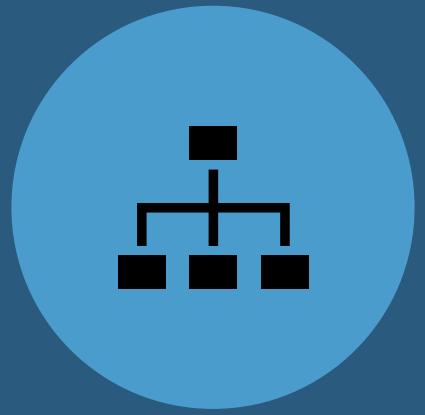
ANDROID GRADLE

- Domain Specific Language (DSL) similar to XML
- Uses Direct Acyclic Graphs (DAGs)
  - Just like some blockchains, eg Ethereum
- Used for building and packaging
  - Especially for very large projects with sub-parts
  - Allows for incremental building (eg: build lib1.so, then lib2.so, then executable1, then executable2, then link lib1.so and lib2.so to both executables)
- Scripts are located in the root directory
  - Might contain additional scripts for other parts or sub-projects
  - build.gradle and settings.grandle

ANDROID VIEWS



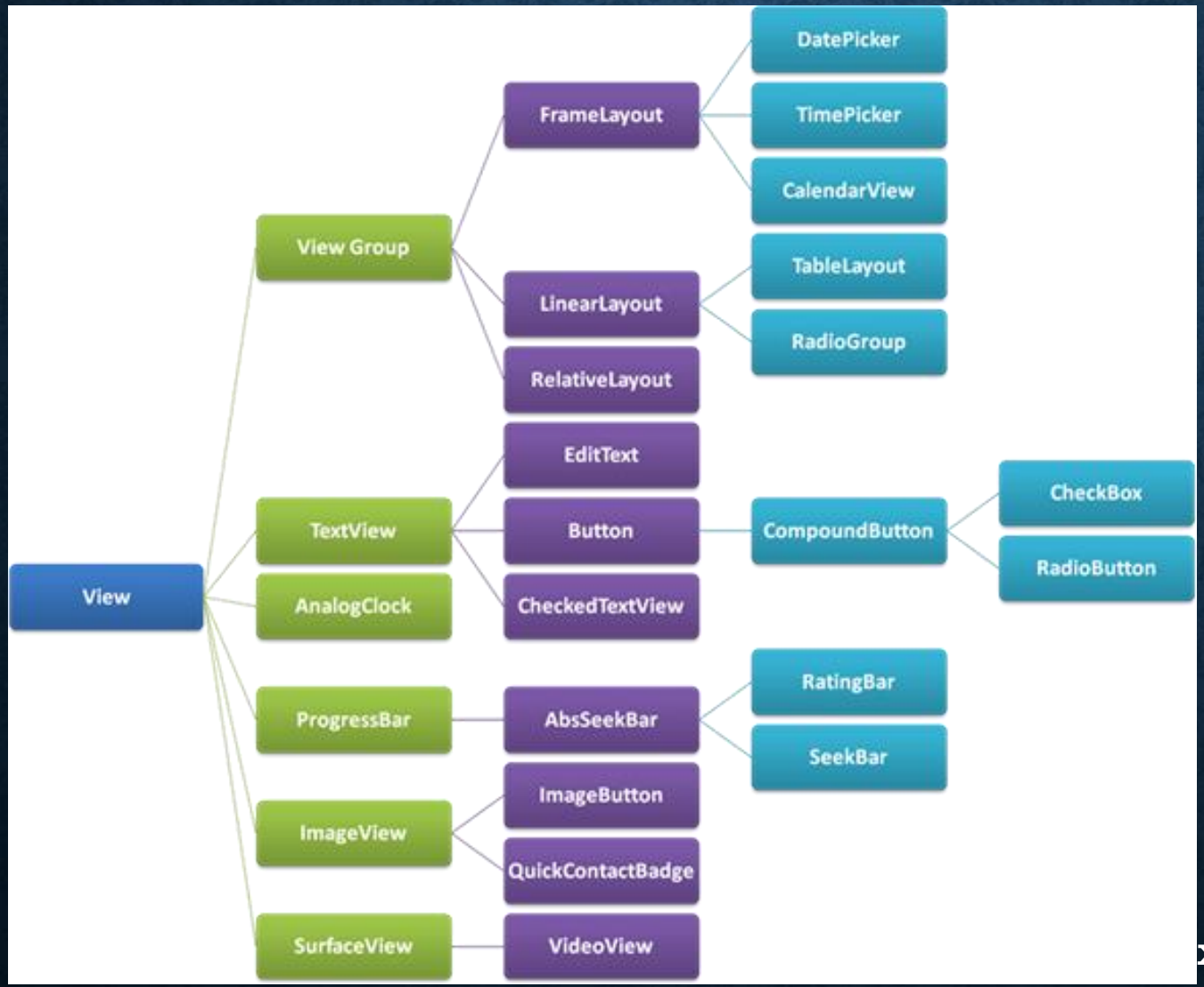
VIEWS ARE INDIVIDUAL UI COMPONENTS THAT ARE ADDED TO THE ACTIVITY/LAYOUT



A LARGE HIERARCHY OF DIFFERENT (INHERITED) VIEWS ARE AVAILABLE

WITH SOME EFFORT YOU CAN CREATE CUSTOM VIEWS

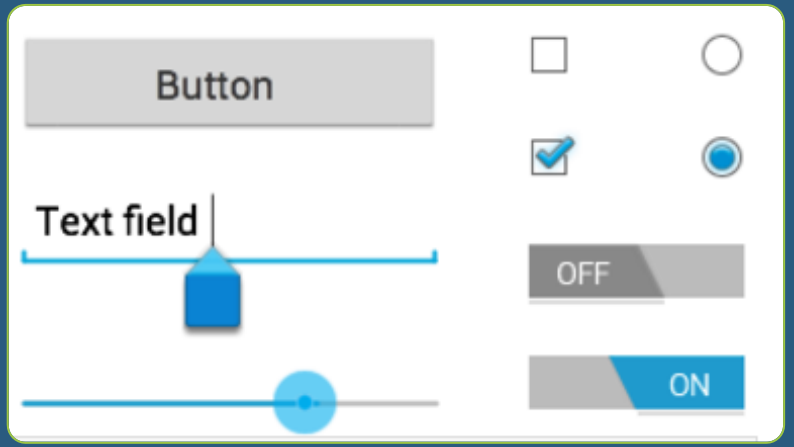
ANDROID VIEWS



xist than the ones shown here

ANDROID VIEWS

- Views will look differently between Android versions and themes
  - The will look different, but should function the same
  - Older Android version might not have certain views, check the documentation
- Some examples



ANDROID BUTTONS

- Buttons can be configured
  - Text only
  - Icon only
  - Text and icon

```
<Button
        android:id="@+id/button_id"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:text="@string/login_label"
        android:drawableLeft="@drawable/login_icon"
/>
```

ANDROID BUTTONS

- Add a click event to the button (just like in HTML and JS)

```
<Button
        android:id="@+id/button_id"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:text="@string/login_label"
        android:drawableLeft="@drawable/login_icon"
        android:onClick="loginExecute"
/>
```

- And implement a function in Java activity

```
public void loginExecute(View view)
{
        // Some login functionality here
}
```

ANDROID BUTTONS

- Buttons can be added dynamically
  - Like most other views
- Buttons can be styled and coloured dynamically
- Icons can be dynamically added and removed from buttons
- Events can be dynamically added and removed from buttons
  - Using OnClickListener

https://developer.android.com/guide/topics/ui/controls/button.html

ANDROID CHECKBOXES

- OnClick events similar to button clicks
- Use the isChecked() function to determine which box is checked
- R is an Android class containing constants of the resources
  - Access to various values defined in the res files

ANDROID CHECKBOXES

```
public void onCheckboxClicked(View view)
{
        // Is the view now checked?
        boolean checked = ((CheckBox) view).isChecked();

        // Check which checkbox was clicked
        switch(view.getId())
        {
                case R.id.bitcoin:
                        if(checked) // Allow Bitcoin payments
                        else // Don't allow Bitcoin payments
                        break;
                case R.id.ethereum:
                        if(checked) // Allow Ethereum payments
                        else // Don't allow Ethereum payments
                        break;
        }
}
```

ANDROID TEXTFIELDS

- Input textbox: EditText
- Output textbox: TextView
- Setting the value can be done by
  - findViewById(R.id.myTextBoxName)
  - Returns the object to be modified

ANDROID INTENTS

- An Intent is an object that provides runtime binding between separate components, such as two activities
- The intent represents an app's "intent to do something"
- To switch activities
  - Set a new Intent object = the class instance of that activity

ANDROID BACK BUTTON

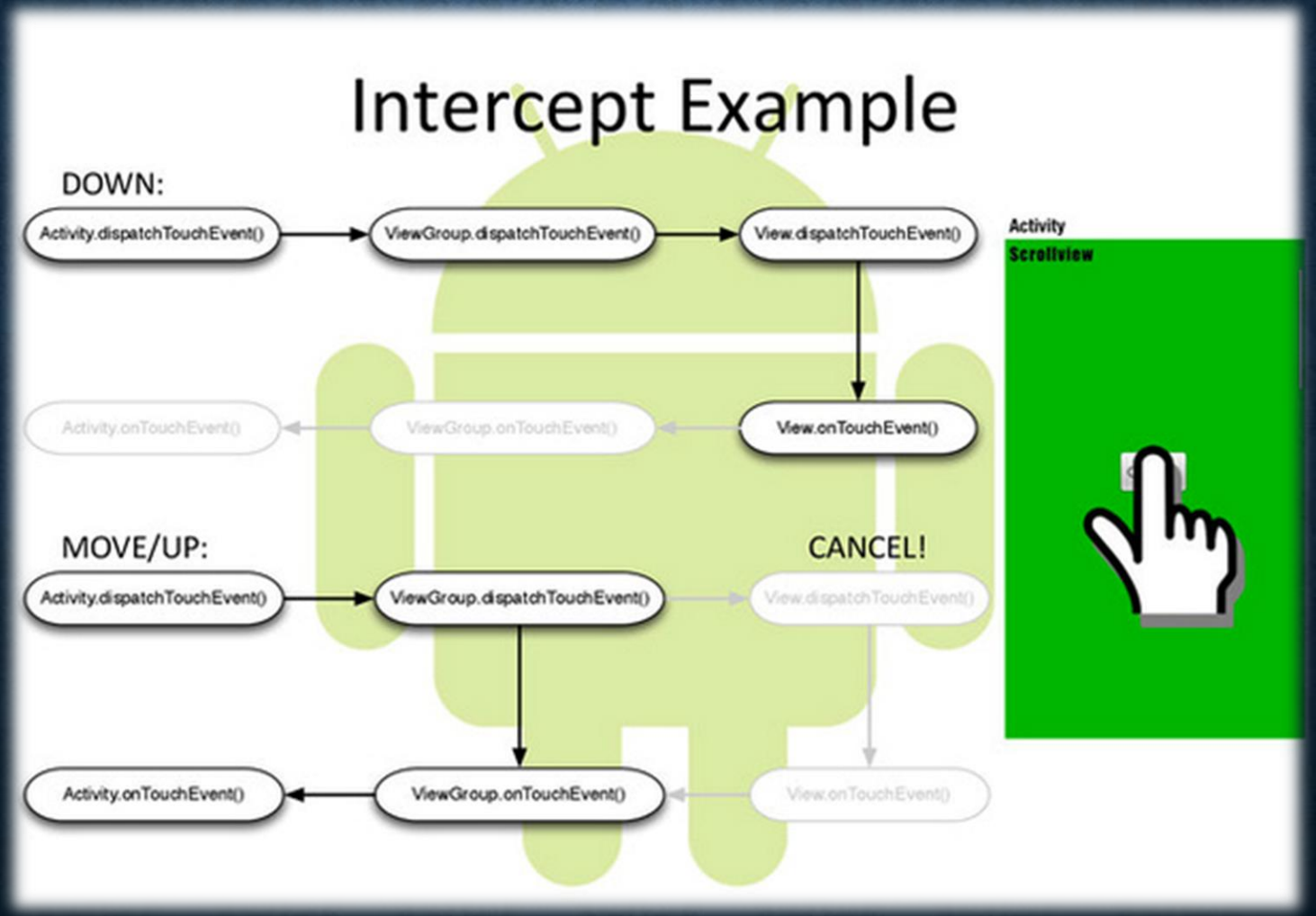
- Special accommodate the phones back button (the hardware/default back button)
  - Called an "Up Action"
- Depending on your Android settings and your apps, there might be default actions for the back button
  - Single Tap: Go to previous activity
  - Tap And Hold: Kill the current app (kills it, does not suspend it)

https://developer.android.com/training/appbar/up-action

ANDROID EVENTS

- A range of events can be used
  - onClick: This is called when the user touches the item
  - onLongClick: Touches and holds the item
  - onKey: Presses hardware key (eg the volume up or lock button)
  - onTouch: Touch event, including press, release, or any movement gesture on the screen
  - onFocusChange: When the user navigates onto or away from the item (navigation key or trackball)
  - onCreateContextMenu: When a context menu is build/rendered
  - onScroll: When the activity is scrolled (eg dragging and pulling)

ANDROID EVENTS



ANDROID EVENTS

- Dynamically add an event listener to a button

```
public class CryptoActivity extends Activity implements OnClickListener
{
        protected void onCreate(Bundle values)
        {
                Button button = (Button) findViewById(R.id.crypto_button);
                button.setOnClickListener(this);
        }

        public void onClick(View view)
        {
                // Do something on the crypto button click
        }
}
```

ANDROID EVENTS

- Handle predefined touch/motion event combinations

```
@Override
public boolean onTouchEvent(MotionEvent event)
{
        int action = MotionEventCompat.getActionMasked(event);
        switch(action)
        {
                case MotionEvent.ACTION_DOWN:
                        Log.d(DEBUG_TAG, "Action was down");
                        return true;
                case MotionEvent.ACTION_UP:
                        Log.d(DEBUG_TAG, "Action up");
                        return true;
                // More options on next slide
        }
}
```

ANDROID EVENTS

```
switch(action)
{
        // More options on previous slide
        case MotionEvent.ACTION_MOVE:
                Log.d(DEBUG_TAG, "Action move");
                return true;
        case MotionEvent.ACTION_CANCEL:
                Log.d(DEBUG_TAG, "Action cancel");
                return true;
        case MotionEvent.ACTION_OUTSIDE:
                Log.d(DEBUG_TAG, "Movement occurred outside bounds");
                return true;
        default:
                return super.onTouchEvent(event);
}
```

ANDROID EVENTS

- Handle press-and-hold events

```
@Override
public void onLongPress(MotionEvent event)
{
        Log.d(DEBUG_TAG, "Long press occurred.
                More details: " + event.toString());
}
```

ANDROID EVENTS

- Handle scroll events

```
@Override
public boolean onScroll(MotionEvent event1, MotionEvent event2,
                        float distanceX, float distanceY)
{
        Log.d(DEBUG_TAG, "Scroll from: " + event1.toString());
        Log.d(DEBUG_TAG, "Scroll to: " + event2.toString());
        Log.d(DEBUG_TAG, "X-axis distance: " + Float.toString(distanceX));
        Log.d(DEBUG_TAG, "Y-axis distance: " + Float.toString(distanceY));
}
```

ANDROID INPUTS

- Ask for user input for a phone number
- Only characters allowed in a phone number will show on the keyboard

```
<EditText
        android:id="@+id/phone"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="@string/phone_hint"
        android:inputType="phone"
/>
```

ANDROID INPUTS

- Ask user input for a password

```
<EditText
        android:id="@+id/txtPassword"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/password_hint"
        android:inputType="textPassword"
/>
```

ANDROID INPUTS

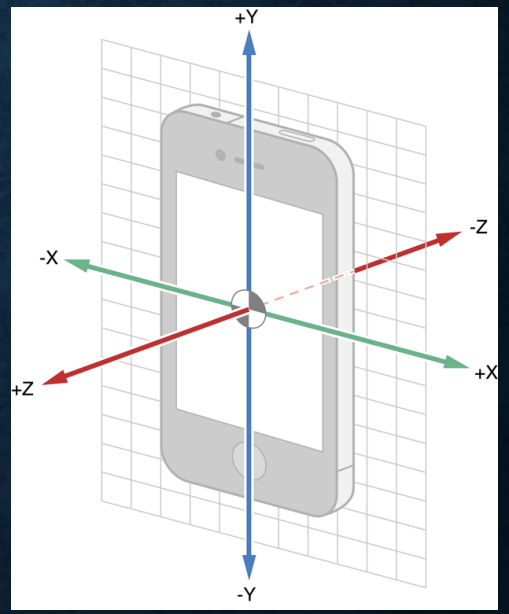
- Many input types exist to customize the keypad presented to the user
  - Standard text and case-sensitive text
  - Passwords
  - Phone numbers
  - Email address, subjects
  - Date, time, datetime
  - Numbers, signed numbers, decimal numbers, password numbers (pins)
  - Person name, phonetic
  - Postal addresses
  - URLs
  - Many more …

ANDROID SENSORS

- Android supports many different sensors
- Sensors are depended on the phone hardware
- Android has to detect the sensors before being able to use them
- onChange events but only if that event is significant for the given device
- Divided into four main categories
  - Motion Sensors
  - Position Sensors
  - Environment Sensors
  - Geolocation Sensors

ANDROID SENSORS

- Motion Sensors
  - Accelerometer: acceleration of movement
  - Gravity: force of gravity
  - Gyroscope: rate of rotation
  - Step Counter: counting the number of steps taken by the user
- These sensors have different variants with/without calibrations



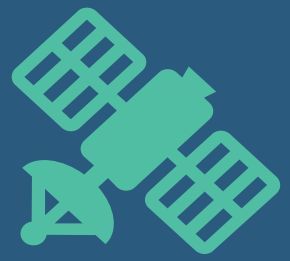
ANDROID SENSORS

- Position Sensors
  - Game Rotation: similar to motion sensor rotation without magnetometer
  - Geomagnetic Rotation: rotation with a magnetometer
  - Orientation: azimuth, pitch, and roll
  - Proximity: distance from object

ANDROID SENSORS

- Environment Sensors
  - Temperature: temperature of device
  - Ambient Temperature : ambient air temperature
  - Pressure: ambient air pressure
  - Humidity: ambient relative humidity
  - Light: illuminance

ANDROID SENSORS



Geolocation Sensors known as Global Navigation Satellite System (GNSS)

Depends on the device manufacturer and the built-in positing system

GPS (US global military positioning system)
GALILEO (EU global positioning system)
GLONASS (Russian global positioning system)
QZSS (Japanese global positioning system)
BeiDou (Chinese global positioning system)

ANDROID STORAGE

- Many apps will need to store some data permanently
- Data might be stored because
  - App state to continue where you left of after restarting the app/phone
  - User settings and preferences
  - User's information, either on the system or a database
  - Storing downloaded files, recorded videos, photos, or sounds
  - Storing credentials, such as cookies and API keys

ANDROID STORAGE

- Android does not natively support cookies
- Many third-party Android libraries exists that have a cookie jar
  - Example: OkHTTP
- You should avoid cookies in any case
- Rather use something more customizable, robust, and portable
  - For instance, a token system or API key

ANDROID STORAGE

- Three main modes to save files
  - Key-value pairs (configuration and registries)
  - Files (standard binary and text files)
  - Databases (mainly SQLite)

ANDROID KEY-VALUE STORAGE

- Store small values, such as configs and settings
  - Similar to Windows registry
  - Similar to INI files
  - Similar to DOM storage used by websites
- Uses the SharedPreferences API
  - Each SharedPreferences file is managed by the framework
  - Data can be private to the app or shared with other apps

ANDROID KEY-VALUE STORAGE

- Write a key-value pair

```
SharedPreferences sharedPref =
        getActivity().getPreferences(Context.MODE_PRIVATE);
SharedPreferences.Editor editor = sharedPref.edit();
editor.putString(getString(R.string.api_key), myApiKey);
editor.commit();
```

ANDROID KEY-VALUE STORAGE

- Read a key-value pair

```
SharedPreferences sharedPref =
        getActivity().getPreferences(Context.MODE_PRIVATE);
String defaultApiKey = "";
String myApiKey = sharedPref.getString(
        getString(R.string.api_key), defaultApiKey);
```

ANDROID FILE STORAGE

- Read and write files like in normal Java
  - Have a look at Java IO and InputStream/OutputStream
- Permissions required to access the file system
- Internal permissions are granted by default
  - App can write to it's own directory (similar to iOS app islands)
  - App requires additional permissions to write to other locations (not possible in iOS)
  - Certain directories/files can not be written at all by normal permissions (eg OS files)
    - Requires super user privileges for reads/writes
    - Might require a rooted device for reads/writes

ANDROID FILE STORAGE

- Additional permissions might be necessary
- Update your AndroidManifest

```
<manifest>
        <uses-permission
                android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
        <uses-permission
                android:name="android.permission.READ_EXTERNAL_STORAGE" />
</manifest>
```

https://developer.android.com/training/basics/data-storage/files.html#WriteExternalStorage

ANDROID FILE STORAGE

- Write some text to a file

```
String filename = "crypto.txt";
String data = "Bitcoin < Ethereum";
FileOutputStream outputStream;
try
{
        outputStream = openFileOutput(filename, Context.MODE_PRIVATE);
        outputStream.write(data.getBytes());
        outputStream.close();
}
catch(Exception e)
{
        e.printStackTrace();
}
```

ANDROID DATABASE STORAGE

- Why SQLite and not other systems like MySQL or PostgreSQL?

ANDROID DATABASE STORAGE

- Why SQLite and not other systems like MySQL or PostgreSQL?
  - Very widely supported by many programming language, devices, operating systems, etc
  - SQLite is easy to use (lightweight SQL implementation)
  - SQLite databases are stored in a single file, can easily be copied, shared, and backed-up
  - Most important: SQLite functionality is built into programming languages and libraries
    - This means that no server has to run all the time, which consumes unnecessary processing power, memory, and battery power
    - SQLite is executed directly by the app (programming languages) and after the query is done, not more execution is required
    - With MySQL and other database systems, a server constantly runs in the background

ANDROID DATABASE STORAGE

- Store any data inside an SQLite database
- SQLite has a similar syntax to MySQL, but it is lightweight/primitive and has less advanced functionality
- Must include the android.database.sqlite package
- Does not require a server, and therefore not host, port, username, or password
  - Simply specify a file and execute SQL queries directly on the file

https://developer.android.com/training/basics/data-storage/databases.html

ANDROID DATABASE STORAGE

- Quickly create a new table

```
SQLiteDatabase database =
        openOrCreateDatabase("mydata.db",MODE_PRIVATE,null);

database.execSQL("CREATE TABLE IF NOT EXISTS
        users(username VARCHAR,password VARCHAR);");

database.close();
```

ANDROID DATABASE STORAGE

- A typical approach in Android is to subclass (inherit) the SQLite class
- Provides an object-oriented approach to accessing the database
  - Similar to the SQL database manager example we did in PHP
  - Encapsulate all SQL queries inside a single class (and maybe even make it a Singleton)