IEMS5722 Mobile Network Programming and Distributed Server Architecture

Lecture 11

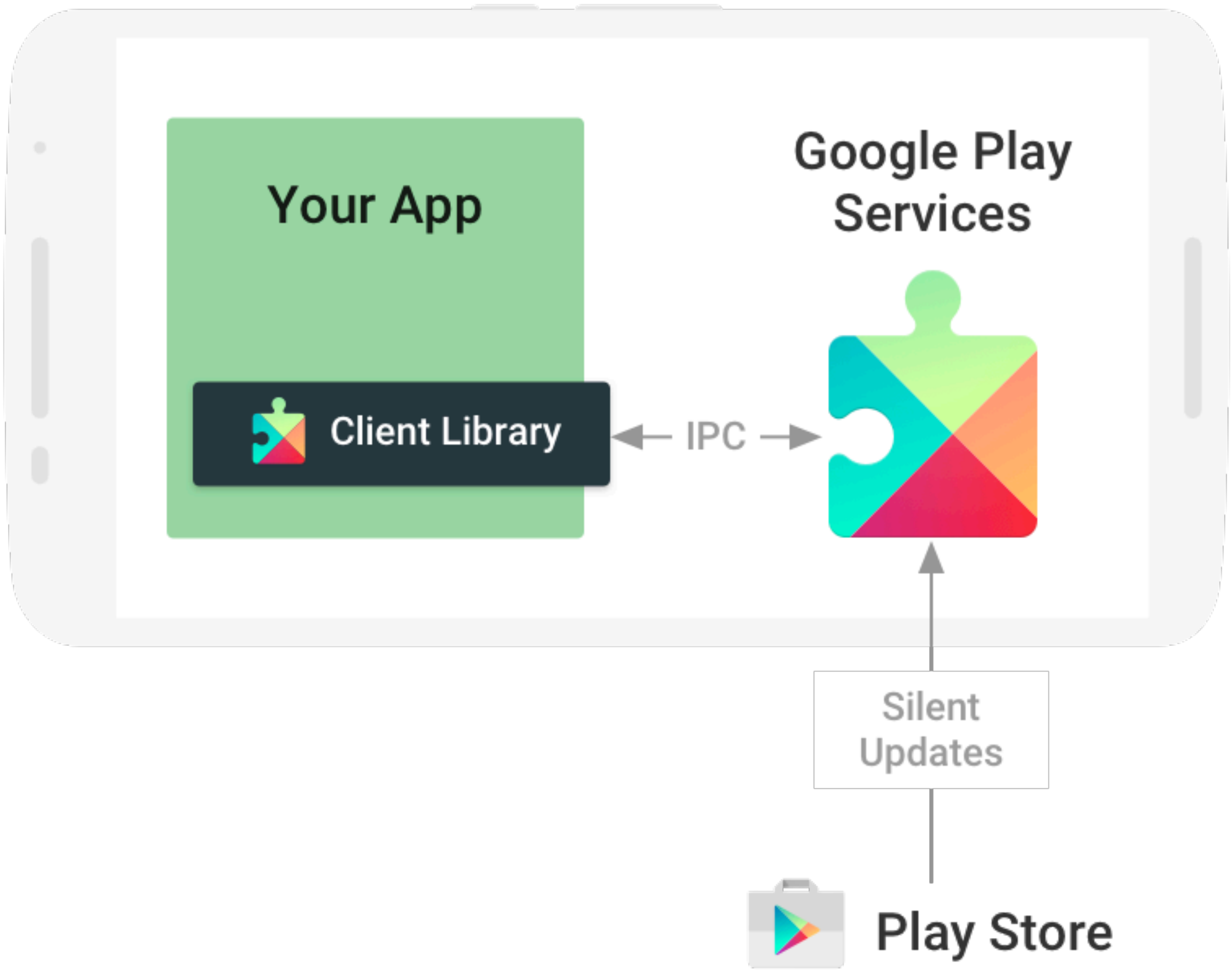
Advanced Android Programming

Google Play Services

Google Play Services

- Google Play Services is a set of cloud services and APIs for Android devices
- Most updated version: 20.06.16 (released Feb 20, 2020)
- Allows you to integrate various services from Google into the mobile app
- Some of Google's services include
    - Location detection
    - Geocoding and reverse geo-coding
    - Maps API (e.g. navigation, street views)
    - Google Drive for cloud storage
    - Google Wallet for online payment

Google Play Services



Google Play Services

- Setting up Google Play Services
  - Download latest Google Play services SDK in Android Studio
  - Update your app-level **build.gradle** file. E.g., to use **Google Location API**:

```
dependencies {
    ...
    implementation 'com.google.android.gms:play-services-location:17.0.0'
    ...
}
```

Reference: https://developers.google.com/android/guides/setup

Accessing Google APIs

- To access various Google APIs, you need to create an instance of a subclass of **GoogleApi**.
  - They automatically manage the connection to Google Play services.
  - Queues request when offline. Executes them in order when a connection is available.
  - GoogleApi objects are also cheap to create, so you can instantiate them as needed to access Google services.

Reference: https://developers.google.com/android/guides/api-client

Accessing Google Services

- To access a service that does not require API authorization, get an instance of the service's client object, pass it to the current **Context** or **Activity**.
- E.g., to get the device's last known location:

```
FusedLocationProviderClient client =
    LocationServices.getFusedLocationProviderClient(this /* Current Activity */);

// Get the last known location
client.getLastLocation()
    .addOnCompleteListener(this, new OnCompleteListener<Location>() {
        @Override
        public void onComplete(@NonNull Task<Location> task) {
            // ...
        }
    });
```

Accessing Google Services with Authorization

- To access a service that **requires** API authorization, first sign the user in and request permissions. Then get an instance of the service's client object, pass it to the **GoogleSignInAccount** and the current **Context** or **Activity**.
- E.g., to access app file in user's Google Drive:

```
// This account must have the necessary scopes to make the API call
// See https://developers.google.com/identity/sign-in/android/
GoogleSignInAccount account = GoogleSignIn.getLastSignedInAccount(this);

// Get the app's Drive folder
DriveResourceClient client = Drive.getDriveResourceClient(this, account);
client.getAppFolder().
    .addOnCompleteListener(this, new OnCompleteListener<DriveFolder>() {
        @Override
        public void onComplete(@NonNull Task<DriveFolder>() {
            // ...
        }
    });
```

Google APIs

- For a list of APIs to be described in the build.gradle file, see:
  - https://developers.google.com/android/guides/setup
- For a reference list of Google Play services APIs, see:
  - https://developers.google.com/android/reference/packages

Location Detection

Location Detection

- Location detection is one of the most commonly used features of smartphones
- Many apps offers location-based services, e.g.:
  - Maps, transportation and navigation
  - Location-based social networking
  - Tracking
  - News
  - Weather forecast
  - …

Location Detection

- Mobile phones can detect locations by using a variety of data sources
  1. GPS (Global Positioning System)
  2. Information of base stations (Cell ID) (e.g. https://opencellid.org/)
  3. Wi-Fi + IP Address
  4. Others

Location Detection

- Android provides simple APIs that return user’s current location by combining the input from different sources

Android Framework's Location API (android.location)

Not recommended for access location

Google Location Services API (com.google.android.gms.location)

Preferred way to add location-awareness

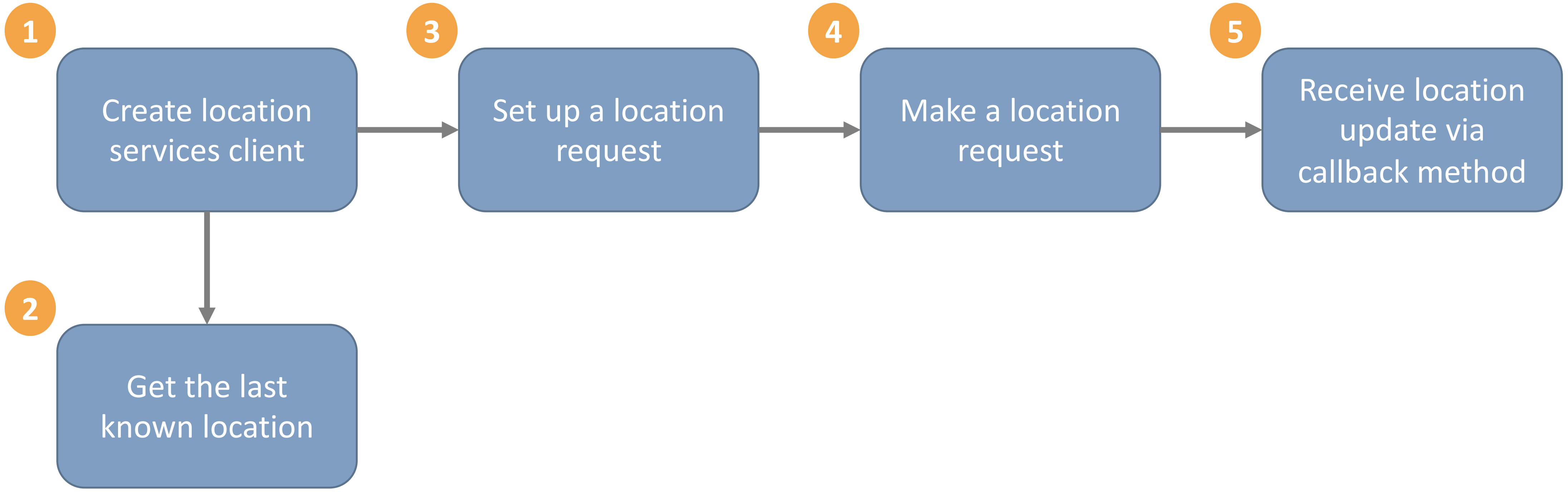
Google Play Services Location APIs

- If your app needs to use location services, you need to request permission from the user
- Location permissions:
  - **ACCESS_COARSE_LOCATION**
    (use Wi-Fi and/or mobile cell data, accurate to the level of a city block)
  - **ACCESS_FINE_LOCATION**
    (use GPS, Wi-Fi and mobile cell data, accurate to a few metres)
  - ACCESS_BACKGROUND_LOCATION
    (for background app with API level 29 and above)

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

Google Play Services Location APIs

- The flow of setting up the app to use the location APIs


1
Create location services client
2
Get the last known location
3
Set up a location request
4
Make a location request
5
Receive location update via callback method

Create Location Services Client

- In your activity’s **onCreate()** method, create an instance of the **Fused Location Provider Client**.

```
private FusedLocationProviderClient fusedLocationClient;

// ...

@Override
protected void onCreate(Bundle savedInstanceState) {
    // ...
    fusedLocationClient = LocationServices.getFusedLocationProviderClient(this);
}
```

Get the Last Known Location

- Once you have created the Location Services client you can get the last known location of a user's device, using the fused location provider's **getLastLocation()**.

```
fusedLocationClient.getLastLocation()
    .addOnSuccessListener(this, new OnSuccessListener<Location>() {
        @Override
        public void onSuccess(Location location) {
            // Got last known location. In some rare situations this can be null.
            if (location != null) {
                // Logic to handle location object
            }
        }
    });
```

Set Up a Location Request

- If your app needs to request location updates, the device needs to enable the appropriate system components (e.g. GPS or Wi-Fi scanning).
- Instead of enabling the services directly, your app specifies the parameters to be requested.

Set Up a Location Request

- Creating a location request:

```
protected void createLocationRequest() {
    LocationRequest locationRequest = LocationRequest.create();
    locationRequest.setInterval(10000);
    locationRequest.setFastestInterval(5000);
    locationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
}
```

- **Interval** (in ms):
  The rate your app prefers receive location update.
- **Fastest Interval** (in ms):
  The fastest rate your app can handle location updates. (Optional)
- **Priority**:
  Request priority. Affects power consumption.

Reference: https://developer.android.com/training/location/change-location-settings

Make a Location Request

- Submit a location request with callback method.

```
@Override
protected void onResume() {
    super.onResume();
    fusedLocationClient.requestLocationUpdates(locationRequest,
        locationCallback, /* Callback method when location update is available */
        Looper.getMainLooper());
}
```

- Stop location updates when the activity is no longer in focus.

```
@Override
protected void onPause() {
    super.onPause();
    fusedLocationClient.removeLocationUpdates(locationCallback);
}
```

Define the Location Update Callback

```
private LocationCallback locationCallback;

// ...

@Override
protected void onCreate(Bundle savedInstanceState) {
    // ...
    locationCallback = new LocationCallback() {
        @Override
        public void onLocationResult(LocationResult locationResult) {
            if (locationResult == null) {
                return;
            }
            for (Location location : locationResult.getLocations()) {
                // Update UI with location data
                // ...
            }
        }
    };
}
```

Using Google Maps

Using Google Maps in Your App

- You can embed an interactive map in your app by using **Google Maps Android API**
- Google Maps functions include:
    - Provide interactive maps with 3D maps, satellite view, terrain view, road maps, etc.
    - Allow overlaying of different components, such as markers, polygons, etc.
    - Control user's view such as rotation, zoom, pan
    - Street view

Getting Started

- To embed Google Maps in your app, you need to set up your app and obtain an API key from Google
- Follow the instructions at: https://developers.google.com/maps/documentation/android-sdk/start
- The fast and easy way is to create a Google Maps Activity in your project, following the link, and copy the API to the **google_maps_api.xml** file.
- The manifest will make use an automatically created string resource.

```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="@string/google_maps_key" />
```

- Or you can manually create an API key on Google Cloud Platform Console, and copy it back to the manifest file.

Add a Map to Your App

- You can create a separate Google Maps Activity.
- Or you can add a map by adding a fragment to your activity.
  - Add a **<fragment>** element to the activity's layout file:

```
<?xml version="1.0" encoding="utf-8"?>
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    android:name="com.google.android.gms.maps.MapFragment"
    android:id="@+id/map"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

Add a Map to Your App

- To work with the map, you need to obtain the map object.

```
public class MainActivity extends FragmentActivity implements OnMapReadyCallback {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        MapFragment mapFragment = (MapFragment) getFragmentManager()
            .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }
    @Override
    public void onMapReady(GoogleMap map) {
        map.addMarker(new MarkerOptions()
            .position(new LatLng(0, 0))
            .title("Marker"));
    }
}
```

Other Maps SDKs

- HERE Technologies
  https://developer.here.com/
- OpenStreetMap
  https://wiki.openstreetmap.org/wiki/Develop
- Baidu Map
  http://lbsyun.baidu.com/index.php?title=androidsdk

Geocoding & Reverse Geocoding

Geocoding & Reverse Geocoding

- **Geocoding**
  - Converting an address to a geographic location (latitude/longitude)
- **Reverse Geocoding**
  - Converting a geographic location (latitude/longitude) to an address

- Both of these functions can be performed by using the **Geocoder** class in Android

Reference: https://developer.android.com/reference/android/location/Geocoder

Geocoding

- In a mobile app, you might want to let the user **enter** a location on the map instead of getting the current location.
- Consider an app for getting a driver pick-up or a food delivery drop-off.
- Then you will need to translate the location name to a geographical location (**geocoding**).

Geocoding

- E.g., setting a marker to a map object.

```
Geocoder geocoder = new Geocoder(this /* context */);
String cuhkLocationName = "The Chinese University of Hong Kong, Shatin, Hong Kong";

try {
    List<Address> addresses = geocoder.getFromLocationName(cuhkLocationName, 1);

    if ((addresses != null) && (addresses.size() > 0)) {
        map.addMarker(new MarkerOptions()
            .position(new LatLng(addresses.get(0).getLatitude(),
                                 addresses.get(0).getLongitude()))
            .title("CUHK"));
    }
} catch (IOException e) {
    e.printStackTrace();
}
```

Reverse Geocoding

- In a mobile app, you might want to convert a geographical location to the street address (**reverse geocoding**).
- E.g., getting the name of the current location, then searching the Web for nearby restaurants.
- To use reverse geocoding, you need a precise geolocation, therefore you should ask for the **ACCESS_FINE_LOCATION** permission.
- You can also use the **Geocoder** class to perform reverse geocoding.

Reverse Geocoding

- The function **getFromLocation()** is a blocking and may take several seconds to respond.
- You should not execute this on the UI thread.
- Instead, you should use one of the asynchronous access methods, e.g.:
  - AsyncTask
  - Intent service (IntentService) with result receiver (ResultReceiver) or local broadcasting (LocalBroadcastManager)

Reverse Geocoding

- Assume last known location is stored in **location**.

```
Geocoder geocoder = new Geocoder(this /* context */);
String addressLine;

try {
    List<Address> addresses = geocoder.getFromLocation(location.getLatitude(),
                                                       location.getLongitude(), 1);

    if ((addresses != null) && (addresses.size() > 0)) {
         addressLine = addresses.get(0).getAddressLine(0);
    }
} catch (IOException e) {
    e.printStackTrace();
}
```

Other Cloud Services

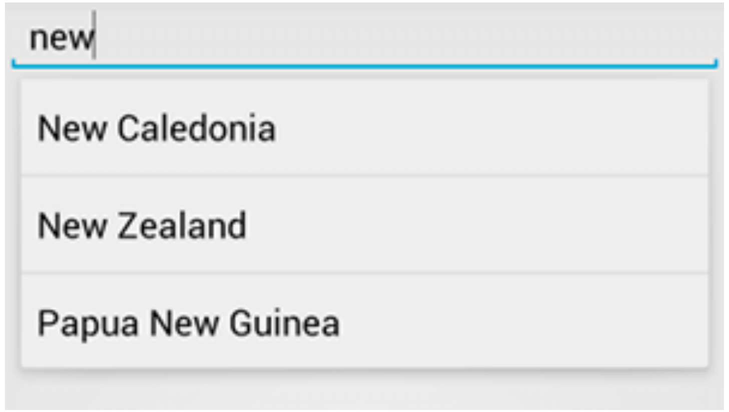
Cloud Services & APIs

- Facebook
  https://developers.facebook.com/
- Instagram
  https://developers.facebook.com/docs/instagram
- WeChat
  https://open.weixin.qq.com/
- Baidu
  https://developer.baidu.com/
- YouTube
  https://developers.google.com/youtube
- IBM Cloud
  https://www.ibm.com/cloud

Auto-complete Text Views

Auto-complete Text Views

- A text view that automatically suggests words or phrases for input while the user is typing
- Useful when you allow the user to perform searching and input specific information (e.g. country names, addresses, etc.)
- You can use one of the following classes depending on your requirements
  - AutoCompleteTextView
  - MultiAutoCompleteTextView



Reference: https://developer.android.com/training/keyboard-input/style#AutoComplete

Auto-complete Text Views

- Add the **AutoCompleteTextView** to your layout. E.g.,

```xml
<AutoCompleteTextView
    android:id="@+id/autocomplete_country"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
```

Auto-complete Text Views

- Define the array that contains the suggestions. You can define it in the XML resource file (e.g., res/values/strings.xml):

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="countries_array">
        <item>Afghanistan</item>
        <item>Albania</item>
        <item>Algeria</item>
        <item>American Samoa</item>
        <item>Andorra</item>
        <item>Angola</item>
        <item>Anguilla</item>
        <item>Antarctica</item>
        ...
    </string-array>
</resources>
```

Auto-complete Text Views

- In your activity, set up an adapter to provide the suggestions, and assign the adapter to the text view (similar what you have done with list view).

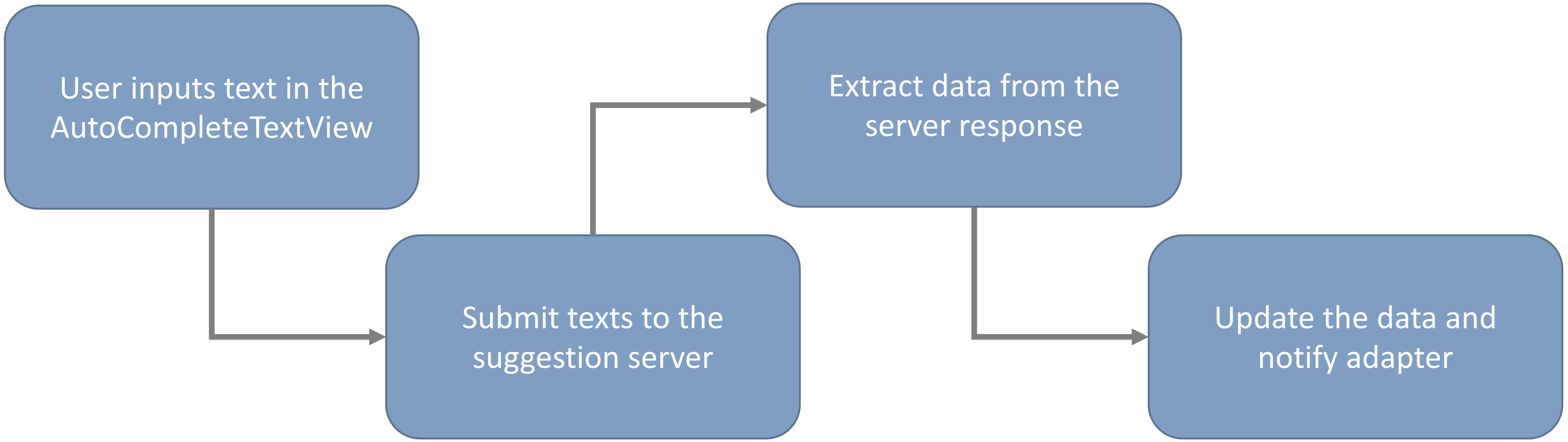
```
AutoCompleteTextView textView = (AutoCompleteTextView)
    findViewById(R.id.autocomplete_country);
String[] countries = getResources().getStringArray(R.array.countries_array);

// Create the adapter and set it to the AutoCompleteTextView
ArrayAdapter<String> adapter =
        new ArrayAdapter<>(this, android.R.layout.simple_list_item_1, countries);

// Assign the adapter to the AutoCompleteTextView
textView.setAdapter(adapter);
```

Auto-complete Text Views

- What if you don't want to hard-code the suggestion list in the app?
- You can send the intermediate input of the user to the suggestion server, and use the data returned by the server update the adapter data.


User inputs text in the AutoCompleteTextView
Submit texts to the suggestion server
Extract data from the server response
Update the data and notify adapter

Auto-complete Text Views

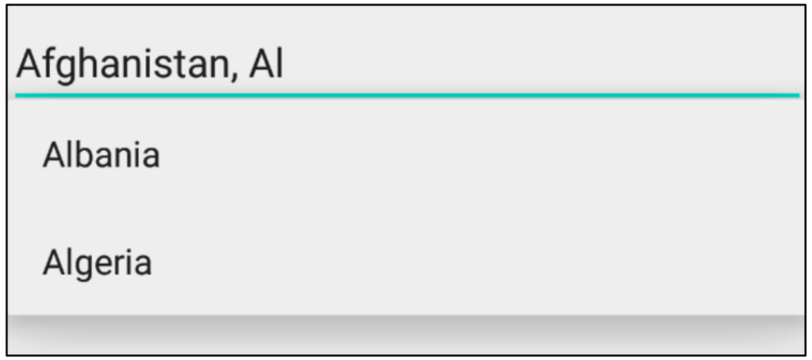
```
RequestQueue queue = Volley.newRequestQueue(MainActivity.this);

textView.addTextChangedListener(new TextWatcher() {
    @Override
    public void afterTextChanged(Editable s) {
        final String query = s.toString().trim();
        if (query.length() > 1) {
            // Combine the query into a URL => urlWithQuery
            StringRequest request = new StringRequest(urlWithQuery, new Response.Listener<String>() {
                @Override
                public void onResponse(String response) {
                    // Extract data... => suggestions
                    adapter.clear();
                    adapter.addAll(suggestions);
                    adapter.notifyDataSetChanged();
                }
            }, null);

            queue.add(request);
        }
    }
});
```

MultiAutoCompleteTextView

- MultiAutoCompleteTextView allows user to input multiple items separate by a separator (e.g. a comma)
- Suggestions will be provided for the latest item in the text view



```
MultiAutoCompleteTextView textView = (MultiAutoCompleteTextView)
    findViewById(R.id.autocomplete_country);
String[] countries = getResources().getStringArray(R.array.countries_array);

ArrayAdapter<String> adapter =
        new ArrayAdapter<>(this, android.R.layout.simple_list_item_1, countries);

textView.setAdapter(adapter);
// Need a tokenizer to distinguish the substrings
textView.setTokenizer(new MultiAutoCompleteTextView.CommaTokenizer());把逗号设置成分割
```

Gradle

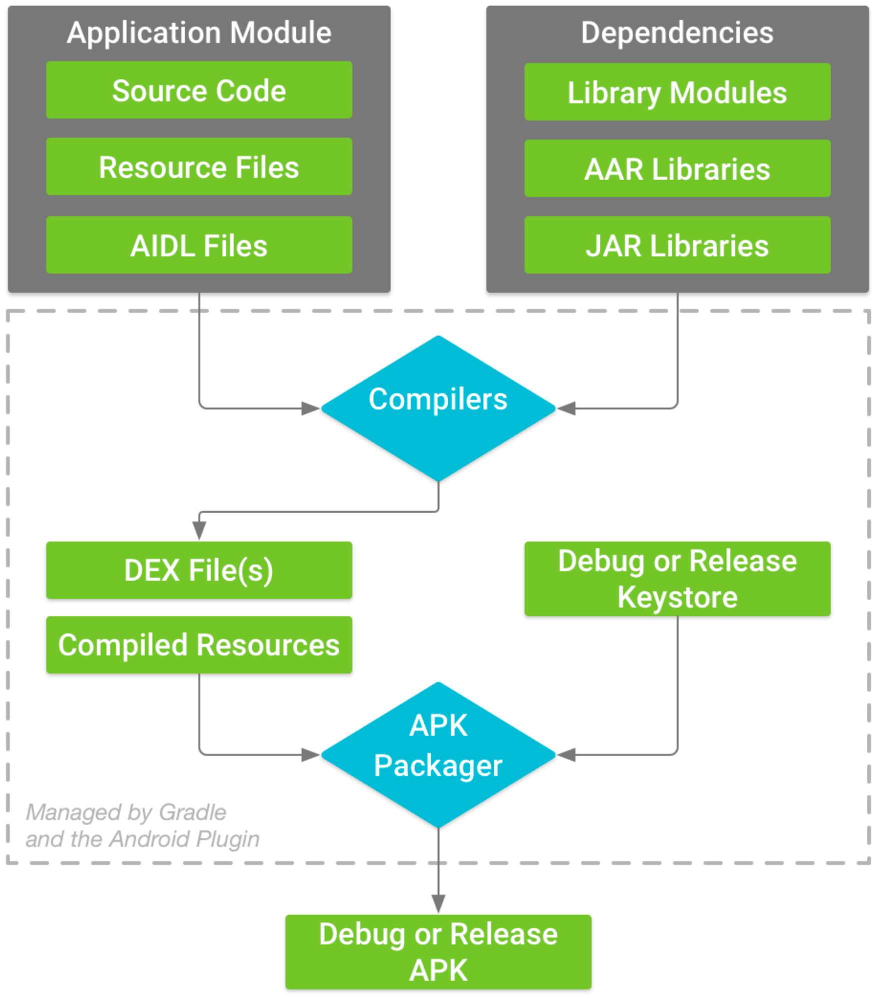
Gradle

- Gradle is the official build tool for Android applications
- It helps you build (compile), test, run and package your apps
- It is integrated into Android Studio through the Android Gradle Plugin
- It can also be executed independently from the command line

Reference: https://gradle.org/

Gradle

- Build process of an Android app

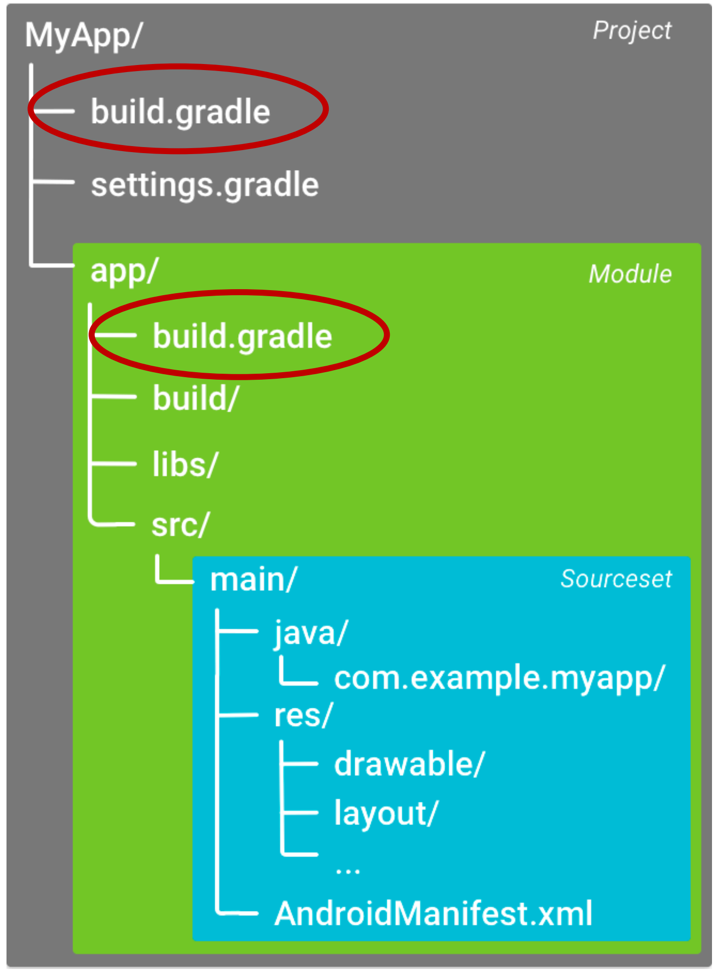


What are the benefits?

- Using Gradle allows you to
  - Configure and customize the build process
  - Create different versions of your app with different features, settings or parameters under the same project
  - Easily incorporate third-party modules into your application

Build Configuration

- An Android Studio project contain a top-level build file and a build file for each module ("app" is a module in the project)
- The build files are all named "**build.gradle**"
- In most cases, you only need to modify the build files at the module level



Reference: https://developer.android.com/studio/build

Build Configuration

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 29
    buildToolsVersion "29.0.3"

    defaultConfig {
        applicationId "hk.edu.cuhk.ie.iems5722"
        minSdkVersion 21
        targetSdkVersion 29
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }

}
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.1'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'
}
```

Apply the Android plugin for Gradle in this build

Configure the core settings and entries in AndroidManifest.xml

Specify different build types. The two default build types are “debug” and “release”.

Declare dependencies to libraries, jar files, remote packages

Product Flavors

- When publishing an app, you might want to publish different variants of the app, which features different functions
- You can specify different "**product flavors**" in the **build.gradle** file, and specify their individual parameters

```
flavorDimensions "tier"
productFlavors {
    free {
        dimension "tier"
        applicationId 'hk.edu.cuhk.ie.iems5722.free'
    }

    pro {
        dimension "tier"
        applicationId 'hk.edu.cuhk.ie.iems5722.pro'
    }
}
```

Reference: https://developer.android.com/studio/build/build-variants

Product Flavors

```
android {
  ...
  buildTypes {
    debug {...}
    release {...}
  }

  flavorDimensions "api", "mode"

  productFlavors {
    demo {
      dimension "mode"
      ...
    }
    full {
      dimension "mode"
      ...
    }

...
```

```
...
    minApi24 {
      dimension "api"
      minSdkVersion 24
      versionCode 30000 + android.defaultConfig.versionCode
      versionNameSuffix "-minApi24"
      ...
    }
    minApi23 {
      dimension "api"
      minSdkVersion 23
      versionCode 20000  + android.defaultConfig.versionCode
      versionNameSuffix "-minApi23"
      ...
    }
    minApi21 {
      dimension "api"
      minSdkVersion 21
      versionCode 10000  + android.defaultConfig.versionCode
      versionNameSuffix "-minApi21"
      ...
    }
  }
}
...
```

- Build variant: [minApi24, minApi23, minApi21][Demo, Full][Debug, Release]
- Corresponding APK: app-[minApi24, minApi23, minApi21]-[demo, full]-[debug, release].apk

Using BuildConfig

- You can specify constant values in the **build.gradle** file, and use them in your Java code
- When building, different build types or product flavors will use the corresponding values automatically
- Steps:
  1. Specify constant values in **build.gradle** using **buildConfigField**
  2. Use the values in your Java code using the **BuildConfig** class

Using BuildConfig

- build.gradle

```
android {
  buildTypes {
    release {
      buildConfigField("String", "APP_NAME", "App")
      buildConfigField("Boolean", "LOG", "false")
      resValue("string", "app_name", "App")
    }
    debug {
      buildConfigField("String", "APP_NAME", "App (Debug Version)")
      buildConfigField("String", "LOG", "true")
                        Boolean
      resValue("string", "app_name", "App")
    }
  }
}
```

- Java

```
String app_name_1 = BuildConfig.APP_NAME;
boolean isLogging = BuildConfig.LOG;
String app_name_2 = getString(R.string.app_name);
```

Using BuildConfig

- In developing client-server systems, one scenario in which you will use BuildConfig is to specify the URL to the APIs for the debug and release build types of your app

```
android {
  buildTypes {
    release {
      buildConfigField("String", "API_SERVER", "api.myserver.com")
    }
    debug {
      buildConfigField("String", "API_SERVER", "dev.myserver.com")
    }
  }
}
```

ProGuard

ProGuard

混淆

- ProGuard is a tool that helps you shrinks, optimizes, and obfuscates the codes of your app

使反向工程变难

- It does so by removing unused code, renaming classes, fields, and methods with semantically obscure names
- Why?
  - Creates an APK file with smaller size
  - Makes the app more difficult to reverse engineer

Reference:
https://developer.android.com/studio/build/shrink-code
https://www.guardsquare.com/en/products/proguard

ProGuard

- ProGuard will only be run when you build your application in release mode (and if you have enabled it)
- To enable ProGuard:

```
android {
    buildTypes {
        release {
            minifyEnabled true
            shrinkResources true
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'),
                'proguard-rules.pro'
        }
    }
    ...
}
```

Enable code shrinking and resource shrinking

Include the default ProGuard rules

Configuring ProGuard

- In some cases, ProGuard might not analyze your code correctly, and might remove codes that are actually required in your app
- You can specifically ask ProGuard to not modify some of the codes in your app when necessary
- If you see **ClassNotFoundException** when building your app for release, you can add lines in the following format to the **proguard-rules.pro** file in the app module

```
-keep public class hk.edu.cuhk.ie.iems5722.MyClass
```

Project Presentations

Project

- To facilitate testing and marking, please name the package of your app using the following format:
  - **hk.edu.cuhk.ie.iems5722.groupX**
    (Replace X with your group number)
- One member of your group should post the following to the **Discussion Board** on Blackboard (https://blackboard.cuhk.edu.hk/) on or before **Sun May 10, 2020**.
  - Your app's APK (debug version is okay) for everyone to try
  - Your presentation video (max. 6 minutes)
  - Your report (in PPT/PPTX)
- And also submit ALL the source codes (in a ZIP file) to submission link.

Project Presentations (Video)

- Each group should record a video no more than 6 minutes to present their app
- Your presentation is suggested to be formatted as follows:
  - **Introduction**: What is the app you have developed?
  - **Functions**: What are the major functions?
  - **Demonstration**: A trial run / an example usage / demonstrate most important features
  - **Challenges**: What is one of the most challenging parts? How did you solve it?
  - **Future Work**: What additional things will you add if you have more time?

Project Report (PPT/PPTX)

写上成员名字

- What should you include in your report:
  - **Problem Definition**: What problem does your app try to solve?
  - **Features / Functions**: What are the major (networking) functions of the app? (with screenshots)
  - **System Architecture**: Use diagrams and illustrations to describe how does the system (client/server) work.
  - **Future Work**: How would you further improve it?
  - **Reference**: A list of third-party libraries that you have used in your project

Project Source Codes

- Create a folder name "groupX" (Replace X with your group number)
  - Create two folders “server” and “client”
  - Put all your server source codes in the “server” folder. Create sub-folders for better organization (e.g., “nginx”, “gunicorn”, “flask”, “python”, etc.)
  - Put the project folder of your Android app in the “client” folder.
  - Compress the folder using ZIP, and submit the “groupX.zip” file to Blackboard submission link under Project (link will be opened later).
- Make sure your server is functional and connectable until **Mon May 18, 2020**.

Project

- You can build up on the assignments, or create a new app from scratch.
- The app should solve some problems (e.g., provides a map/file to a chatroom for all participants to access, provides one-to-one video streaming for conferencing call, provides audio broadcast like in a radio station, etc.)
- The app should make use of networking functions (e.g. send and receive data, files, broadcasting or streaming, multi-player games, local area connectivity, etc.)

Project Assessment

- **Assessment Criteria**
  - Networking functionality (30%)
  - Client-side system design and complexity (20%)
  - Serve- side system design and complexity (20%)
  - Error handling and robustness (10%)
  - User-friendliness & UI/UX design (10%)
  - Presentation & report (10%)

Class Arrangements

- No class next week (Apr 13) due to holiday.
- Next class will be on Apr 20. Final Q&A for the project.

End of Lecture 11