Lab 13
Location-based Service

NetDB
CS, NTHU,
Fall, 2013

Outline

- Google Play Service Set up
- Making Your App Location-Aware
- Google Android Map API

Outline

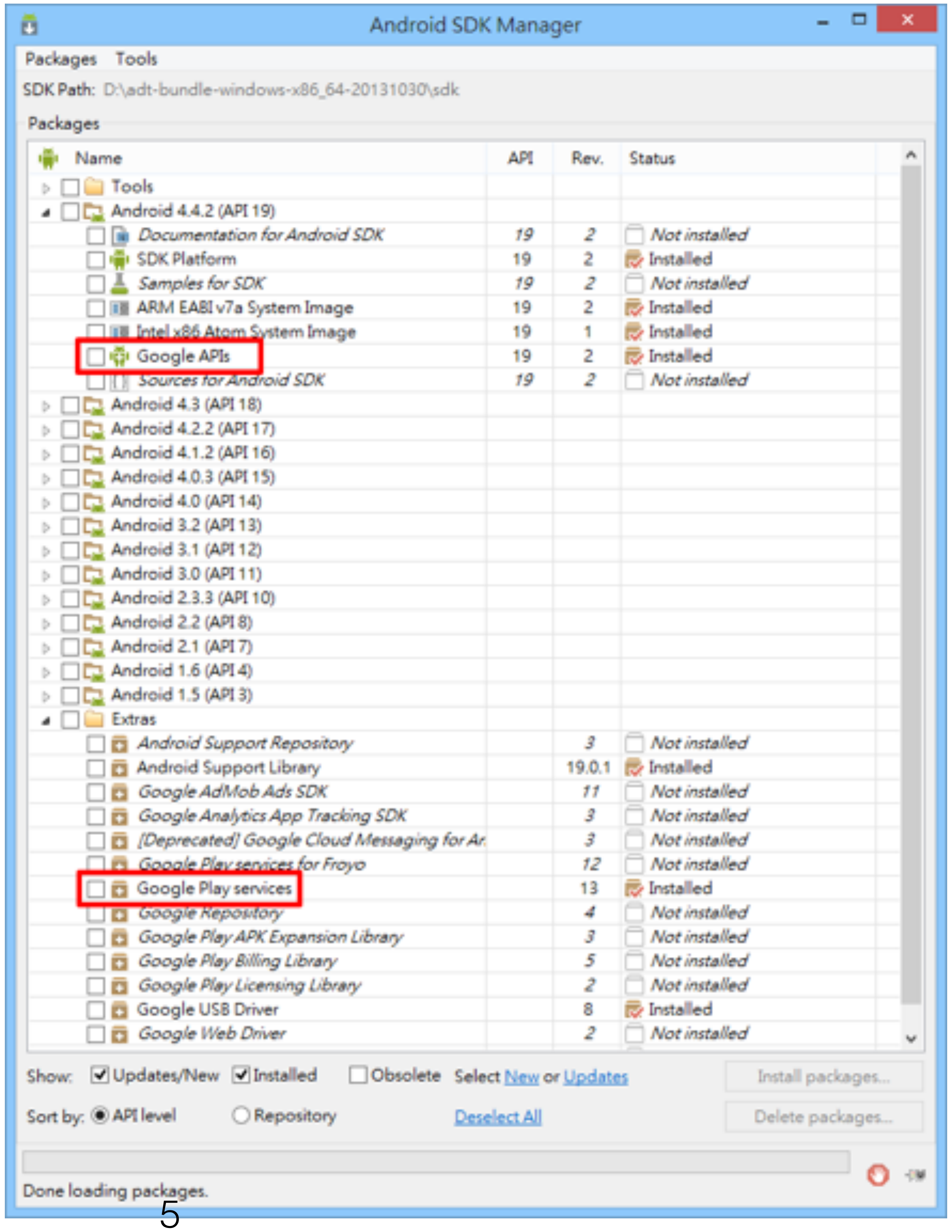
- Google Play Service Set up
- Making Your App Location-Aware
- Google Android Map API

Google Play Service

- Location Service
- Map
- Google+
- Mobile Ads
- ….

Set Up
Google Play Service SDK

- Open SDK Manager, install Google APIs and Google Play services



Android SDK Manager

Packages Tools

SDK Path: D:\adt-bundle-windows-x86_64-20131030\sdk

Packages

Name	API	Rev.	Status
Tools			
Android 4.4.2 (API 19)			
Documentation for Android SDK	*19*	*2*	*Not installed*
SDK Platform	19	2	Installed
Samples for SDK	*19*	*2*	*Not installed*
ARM EABI v7a System Image	19	2	Installed
Intel x86 Atom System Image	19	1	Installed
Google APIs	19	2	Installed
Sources for Android SDK	*19*	*2*	*Not installed*
Android 4.3 (API 18)			
Android 4.2.2 (API 17)			
Android 4.1.2 (API 16)			
Android 4.0.3 (API 15)			
Android 4.0 (API 14)			
Android 3.2 (API 13)			
Android 3.1 (API 12)			
Android 3.0 (API 11)			
Android 2.3.3 (API 10)			
Android 2.2 (API 8)			
Android 2.1 (API 7)			
Android 1.6 (API 4)			
Android 1.5 (API 3)			
Extras			
Android Support Repository		*3*	*Not installed*
Android Support Library		19.0.1	Installed
Google AdMob Ads SDK		*11*	*Not installed*
Google Analytics App Tracking SDK		*3*	*Not installed*
[Deprecated] Google Cloud Messaging for Ar		*3*	*Not installed*
Google Play services for Froyo		*12*	*Not installed*
Google Play services		13	Installed
Google Repository		*4*	*Not installed*
Google Play APK Expansion Library		*3*	*Not installed*
Google Play Billing Library		*5*	*Not installed*
Google Play Licensing Library		*2*	*Not installed*
Google USB Driver		8	Installed
Google Web Driver		*2*	*Not installed*

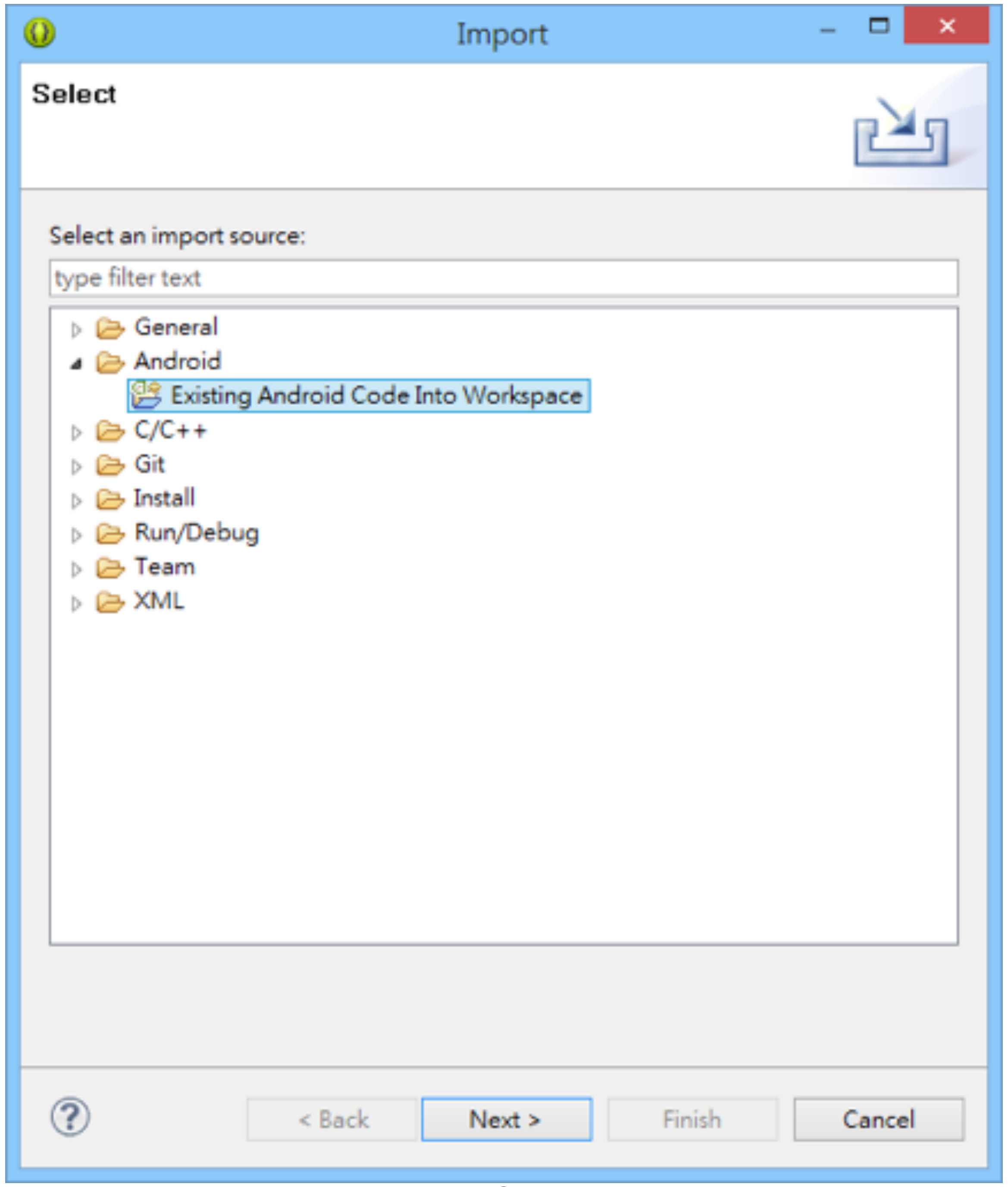
Show: [✓] Updates/New [✓] Installed [ ] Obsolete Select New or Updates

Sort by: [● API level] ○ Repository Deselect All

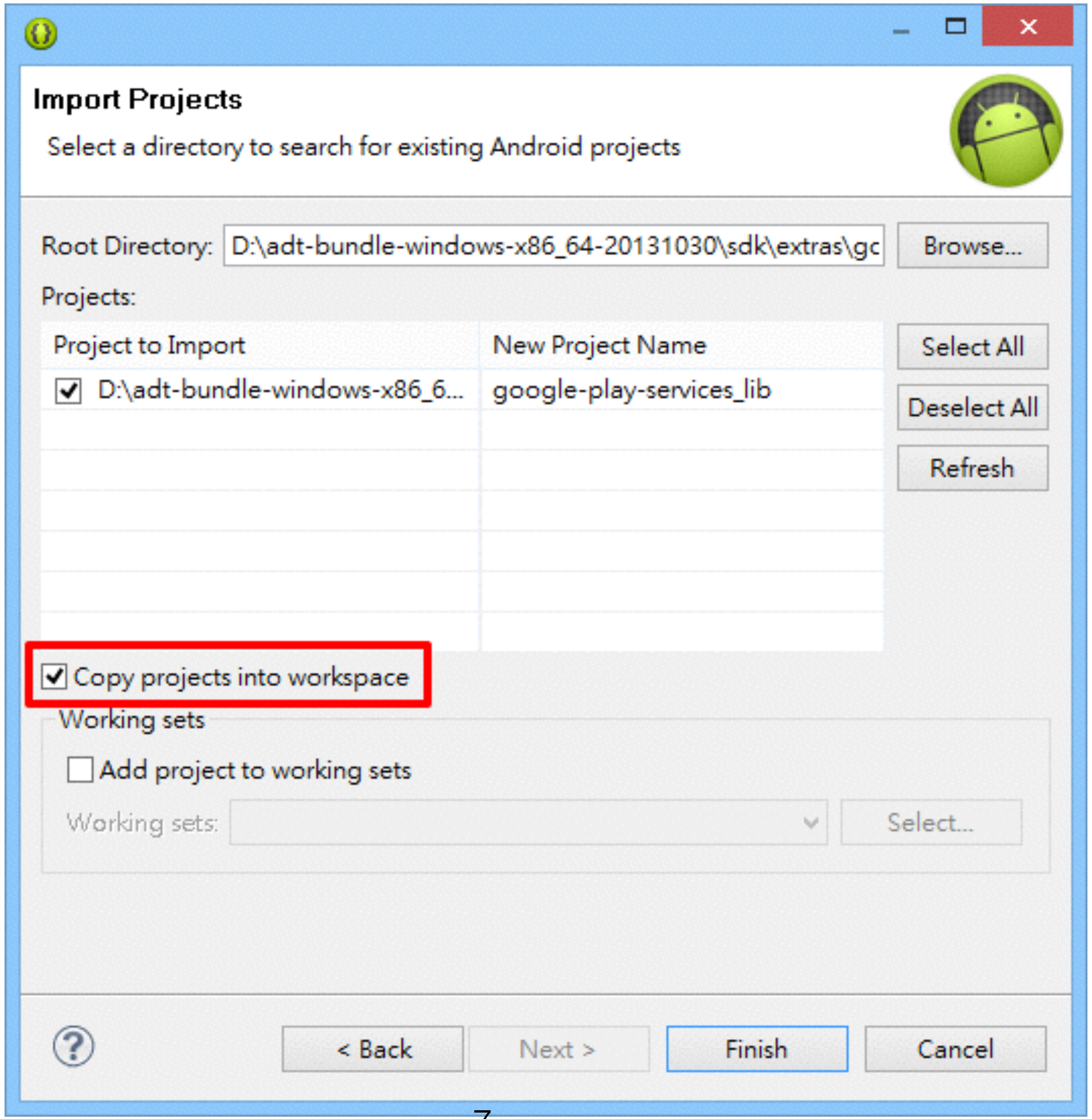
Install packages... Delete packages...

Done loading packages.

Make a copy of the Google Play services library project



- Copy the library project at <android-sdk>/extras/google/google_play_services/libproject/google-play-services_lib/



Import Projects

Select a directory to search for existing Android projects

Root Directory: D:\adt-bundle-windows-x86_64-20131030\sdk\extras\gc

Projects:

Project to Import	New Project Name
✓ D:\adt-bundle-windows-x86_6...	google-play-services_lib

✓ Copy projects into workspace

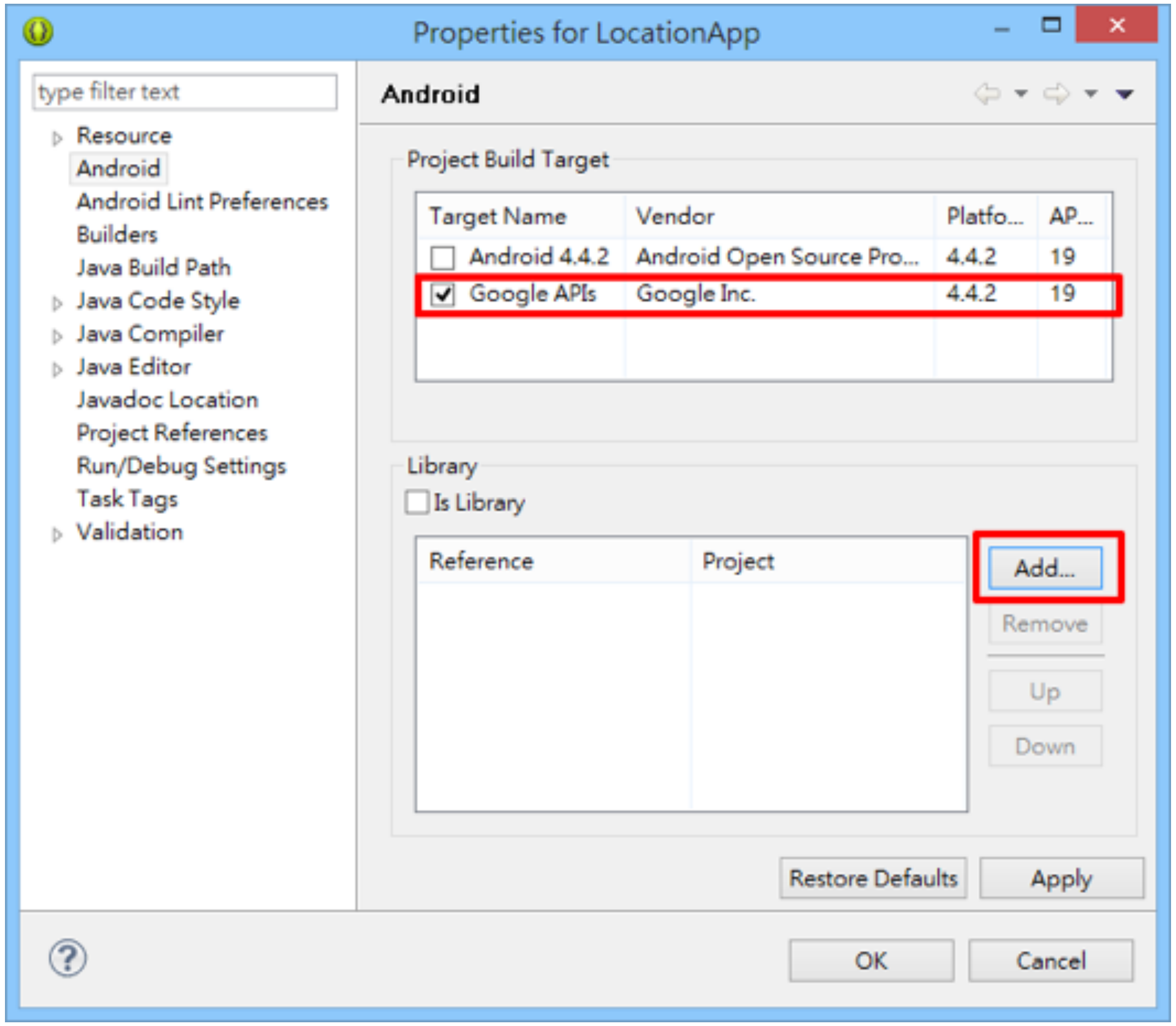
Working sets

☐ Add project to working sets

Working sets:

Referencing a Library Project for Existing Project

- Go to the Properties of the existing project (right click on it)



Properties for LocationApp

type filter text

- Resource
- Android
- Android Lint Preferences
- Builders
- Java Build Path
- Java Code Style
- Java Compiler
- Java Editor
- Javadoc Location
- Project References
- Run/Debug Settings
- Task Tags
- Validation

Android

Project Build Target

	Target Name	Vendor	Platfo...	AP...
☐	Android 4.4.2	Android Open Source Pro...	4.4.2	19
✓	Google APIs	Google Inc.	4.4.2	19

Library

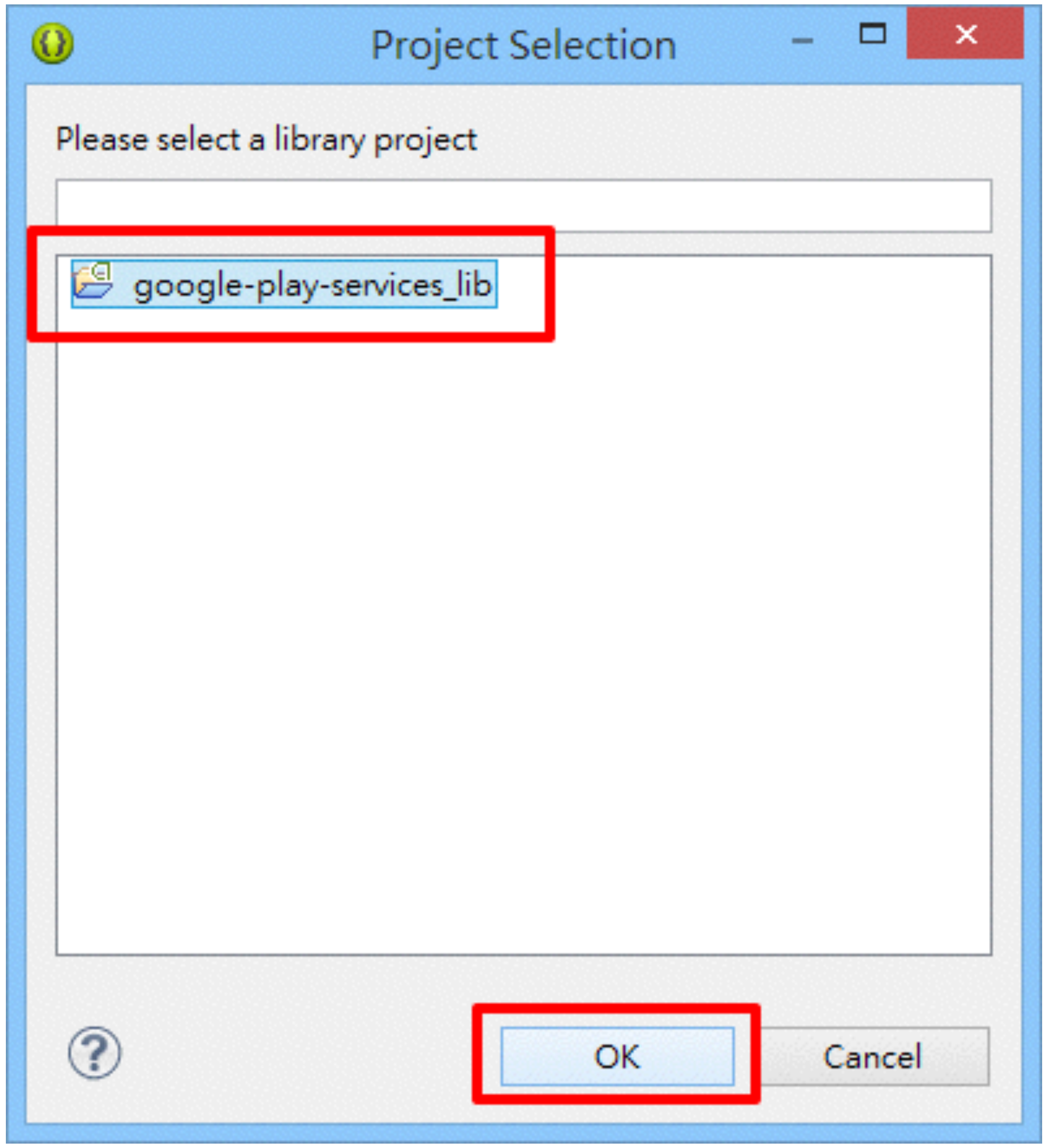
☐ Is Library

Reference	Project

Add... Remove Up Down

Restore Defaults Apply

OK Cancel

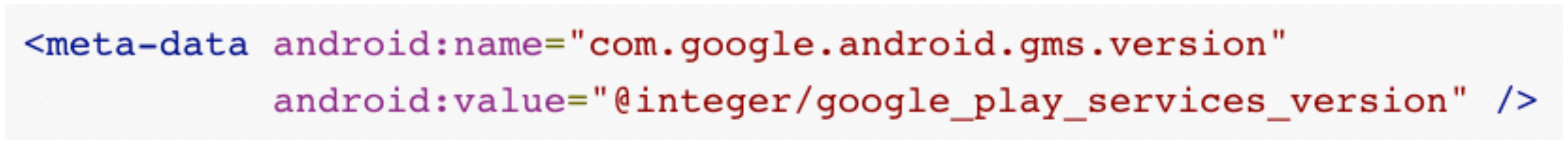

Project Selection

Please select a library project

google-play-services_lib

Configuration

- After you've added the Google Play services library as a dependency for your app project, open your app's manifest file and add the following tag as a child of the <application> element:



```
<meta-data android:name="com.google.android.gms.version"
           android:value="@integer/google_play_services_version" />
```

Reference

- If you encounter any problems during setting up, please look the setting up document from Google for help

Outline

- Google Play Service Setup
- Making Your App Location-Aware
  - Retrieving the Current Location
  - Receiving Location Updates
- Google Android Map API

Outline

- Google Play Service Setup
- Making Your App Location-Aware
    - Retrieving the Current Location
    - Receiving Location Updates
- Google Android Map API

Location Service

- Location Services automatically maintains the user's current location, so all your app has to do is retrieve it as needed

- Location Services sends the current location to your app through a location client, which is an instance of the Location Services class LocationClient. All requests for location information go through this client

Specify App Permissions

- Apps that use Location Services must request location permissions

- Android has two location permissions: ACCESS_COARSE_LOCATION and ACCESS_FINE_LOCATION. The permission you choose controls the accuracy of the current location

- For example, to add ACCESS_COARSE_LOCATION, insert the following as a child element of the <manifest> element:

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

Get the Current Location

- To get the current location, create a location client, connect it to Location Services, and then call its getLastLocation() method
- The return value is the best, most recent location, based on the permissions your app requested and the currently-enabled location sensors

Location Services Callbacks

- Before you create the location client, implement the interfaces that Location Services uses to communicate with your app:
  - ConnectionCallbacks
    - Specifies methods that Location Services calls when a location client is connected or disconnected
      - onConnected()
      - onDisconnected()

Location Services Callbacks

- OnConnectionFailedListener
  - Specifies a method that Location Services calls if an error occurs while attempting to connect the location client
    - onConnectionFailed(ConnectionResult connectionResult)

Connect the Location Client

- Now that the callback methods are in place, create the location client and connect it to Location Services
- You should
  - create the location client in onCreate()
  - connect it in onStart()
  - Disconnect the client in onStop()
- Following this pattern of connection and disconnection helps save battery power

Connect the Location Client

- Create

```
mLocationClient = new LocationClient(this, this, this);
```

- Connect

```
mLocationClient.connect();
```

- Disconnect

```
mLocationClient.disconnect();
```

Get the Current Location

- To get the current location, call getLastLocation()

```
Location loc = mLocationClient.getLastLocation();
```

Outline

- Google Play Service Setup
- Making Your App Location-Aware
    - Retrieving the Current Location
    - Receiving Location Updates
- Google Android Map API

Receiving Location Updates

- To listen to location updates, you also need to go through Specify App Permission ands and Define Location Service Callbacks mentioned previously

Start Location Updates

- To send the request for location updates, create a location client in onCreate(), then make the request by calling requestLocationUpdates() in onConnected() callback of ConnectionCallbacks interface

```
mLocationClient.requestLocationUpdates(mLocationRequest, this);
```

Location Update Callback

- Location Services sends location updates to your app either as an Intent or as an argument passed to a callback method you define
- The callback method that Location Services invokes to send a location update to your app is specified in the LocationListener interface, in the method onLocationChanged(Location)

Location Update Callback

```
@Override
    public void onLocationChanged(Location location) {
        // Report to the UI that the location was updated
        String msg = "Updated Location: " +
                Double.toString(location.getLatitude()) + "," +
                Double.toString(location.getLongitude());
        Toast.makeText(this, msg, Toast.LENGTH_SHORT).show();
    }
```

Reference

- Make your app location-aware
- Location Updates sample code

Outline

- Google Play Service Setup
- Making Your App Location-Aware
- Google Android Map API
    - Set up
    - Map
    - Markers
    - Info Windows

Outline

- Google Play Service Setup
- Making Your App Location-Aware
- Google Android Map API
    - Set up
    - Map
    - Markers
    - Info Windows

Google Android Map API Set Up

- Getting started

Outline

- Google Play Service Setup
- Making Your App Location-Aware
- Google Android Map API
  - Set up
  - Map
  - Markers
  - Info Windows

The Map Object

- The key class when working with a Map object is the GoogleMap class. GoogleMap models the map object within your application

- Within your UI, a map will be represented by either a MapFragment or MapView object

The Map Object

- GoogleMap handles the following operations automatically:
  - Connecting to the Google Maps service
  - Downloading map tiles
  - Displaying tiles on the device screen
  - Displaying various controls such as pan and zoom
  - Responding to pan and zoom gestures by moving the map and zooming in or out

MapFragment

- MapFragment allows you to place a map in an Android Fragment. MapFragment objects act as containers for the map, and provide access to the GoogleMap object

Add a Map

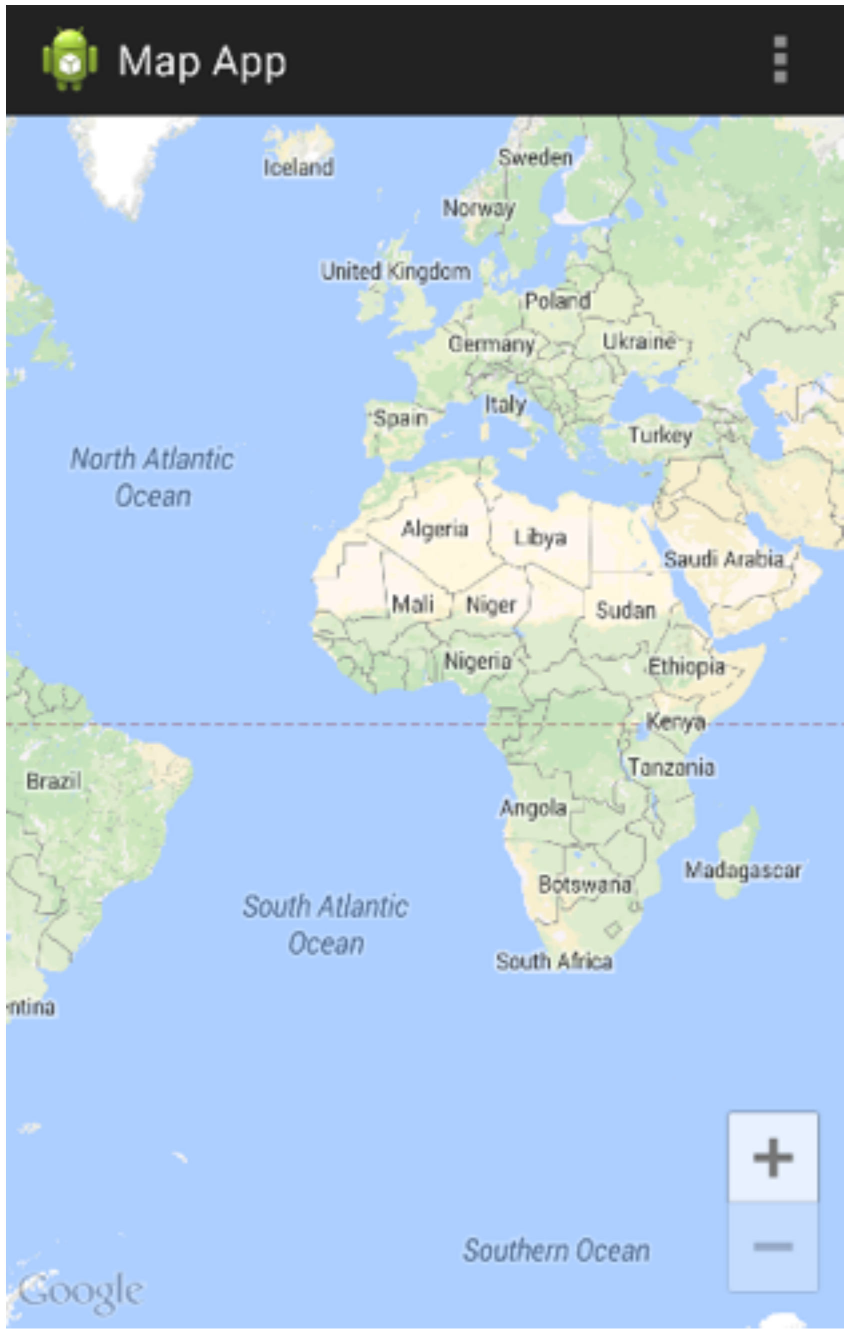
- Add a <fragment> element to the Activity's layout file to define a Fragment object. In this element, set the android:name attribute to com.google.android.gms.maps.MapFragment. This automatically attaches a MapFragment to the Activity

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

Add a Map

- Setting the layout file as the content view for the Activity

- Run the app, you should see the map on the screen



Get the Map Object

- In the Activity object's onCreate() method, get a handle to the GoogleMap object in the MapFragment

```
GoogleMap mMap = ((MapFragment)
getFragmentManager().findFragmentById(R.id.map)).getMap();
```

Configure the Map

- To set view options for a map, you modify its GoogleMap object
  - The camera position, including: location, zoom, bearing and tilt
  - The map type (e.g. normal, satellite, etc.)
  - Whether the zoom buttons and/or compass appear on screen
  - Which gestures a user can use to manipulate the camera
- You can configure these settings via <fragment> attributes in the layout file, or do it programmatically

Outline

- Google Play Service Setup
- Making Your App Location-Aware
- Google Android Map API
  - Set up
  - Map
  - Markers
  - Info Windows

The Markers

- Markers identify locations on the map. Markers are objects of type Marker, and are added to the map with the GoogleMap.addMarker(markerOptions) method

```
private GoogleMap mMap;
mMap = ((MapFragment)
getFragmentManager().findFragmentById(R.id.map)).getMap();
mMap.addMarker(new MarkerOptions()
        .position(new LatLng(0, 0))
        .title("Hello world"));
```

Customize the Marker

- Markers support customization through the following properties:
  - Position (Required)
  - Draggable
  - Icon
  - Anchor
  - Alpha
  - Title
  - Snippet

Marker Events

- You can listen to the following events:
  - Marker click events
  - Marker drag events

Marker Click Event

- You can use an OnMarkerClickListener to listen for click events on the marker. To set this listener on the map, call GoogleMap.setOnMarkerClickListener(OnMarkerClickListener)
- When a user clicks on a marker, onMarkerClick(Marker) will be called and the marker will be passed through as an argument

Marker Drag Event

- You can use an OnMarkerDragListener to listen for drag events on a marker. To set this listener on the map, call GoogleMap.setOnMarkerDragListener(OnMarkerDragListener)

- When a marker is dragged, onMarkerDragStart(Marker) is called initially. While the marker is being dragged, onMarkerDrag(Marker) is called constantly. At the end of the drag onMarkerDragEnd(Marker) is called

Reference

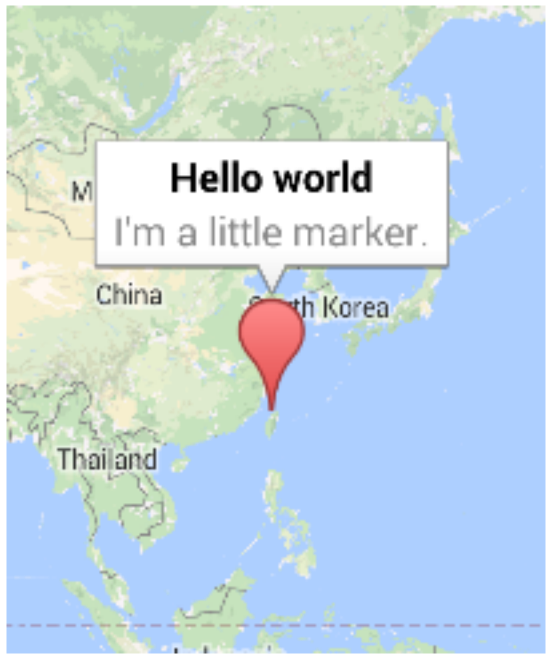
- Markers

Outline

- Google Play Service Setup
- Making Your App Location-Aware
- Google Android Map API
  - Set up
  - Map
  - Markers
  - Info Windows

The Info Window

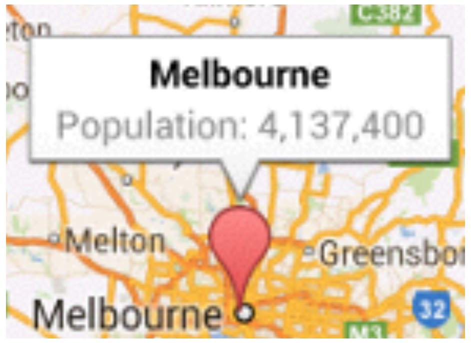
- An info window displays text or images in a popup window above the map. Info windows are always anchored to a marker. Their default behavior is to display when the marker is tapped



Add an Info Window

- The simplest way to add an info window is to set the title() and snippet() methods of the corresponding marker. Setting these properties will cause an info window to appear whenever that marker is clicked

```
static final LatLng MELBOURNE = new LatLng(-37.81319, 144.96298);
Marker melbourne = mMap.addMarker(new MarkerOptions()
                                  .position(MELBOURNE)
                                  .title("Melbourne")
                                  .snippet("Population: 4,137,400"));
```



Customize the Info Window

- To do this, you must create a concrete implementation of the InfoWindowAdapter interface and then call GoogleMap.setInfoWindowAdapter() with your implementation

- The interface contains two methods for you to implement: getInfoWindow(Marker) and getInfoContents(Marker)

Customize the Info Window

- The API will first call getInfoWindow(Marker) and if null is returned, it will then call getInfoContents(Marker). If this also returns null, then the default info window will be used
- getInfoWindow(Marker)
  - Allows you to provide a view that will be used for the entire info window
- getInfoContents(Marker)
  - Allows you to just customize the contents of the window but still keep the default info window frame and background

Info Window Events

- You can use an OnInfoWindowClickListener to listen to click events on an info window
- To set this listener on the map, call GoogleMap.setOnInfoWindowClickListener(OnInfoWindowClickListener)
- When a user clicks on an info window, onInfoWindowClick(Marker) will be called

Note

- The info window that is drawn is not a live view. The view is rendered as an image at the time it is returned

- This means that any subsequent changes to the view will not be reflected by the info window on the map

    - To update the info window later (for example, after an image has loaded), call showInfoWindow()

Note

- Furthermore, the info window will not respect any of the interactivity typical for a normal view such as touch or gesture events

- However you can listen to a generic click event on the whole info window as described in the section below

Reference

- Info Windows

Reference

- Google Android Map API v2
- Sample code
  - must run on a real device