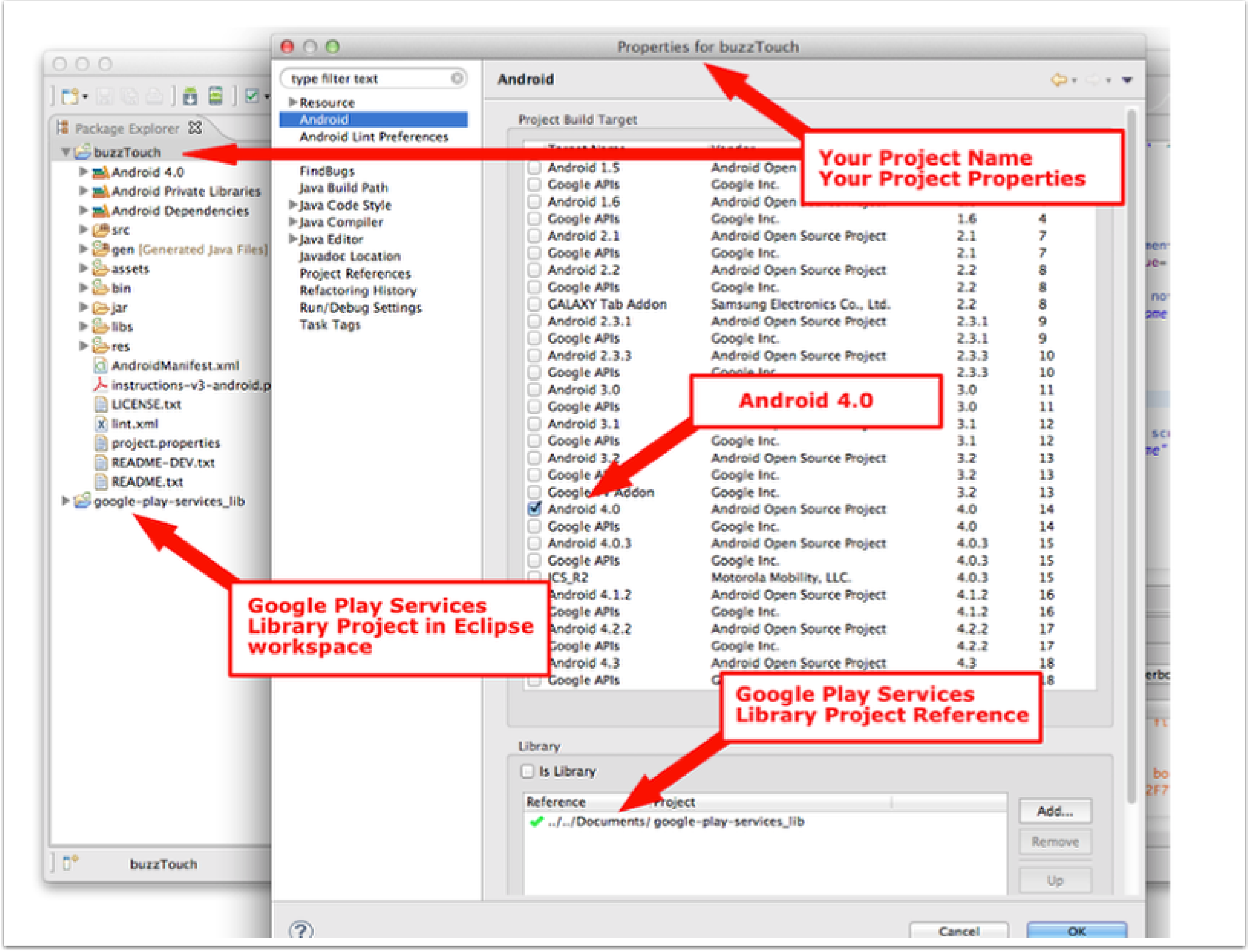
Referencing the Google Play Services Library in your Android BT Project

Preface



Your Project Name
Your Project Properties

Android 4.0

Google Play Services Library Project in Eclipse workspace

Google Play Services Library Project Reference

Project Build Target

	Target Name	Vendor		
☐	Android 1.5	Android Open [illegible]	[illegible]	[illegible]
☐	Google APIs	Google Inc.	[illegible]	[illegible]
☐	Android 1.6	Android Open [illegible]	[illegible]	[illegible]
☐	Google APIs	Google Inc.	1.6	4
☐	Android 2.1	Android Open Source Project	2.1	7
☐	Google APIs	Google Inc.	2.1	7
☐	Android 2.2	Android Open Source Project	2.2	8
☐	Google APIs	Google Inc.	2.2	8
☐	GALAXY Tab Addon	Samsung Electronics Co., Ltd.	2.2	8
☐	Android 2.3.1	Android Open Source Project	2.3.1	9
☐	Google APIs	Google Inc.	2.3.1	9
☐	Android 2.3.3	Android Open Source Project	2.3.3	10
☐	Google APIs	Google Inc.	2.3.3	10
☐	Android 3.0	[illegible]	3.0	11
☐	Google APIs	[illegible]	3.0	11
☐	Android 3.1	[illegible]	3.1	12
☐	Google APIs	Google Inc.	3.1	12
☐	Android 3.2	Android Open Source Project	3.2	13
☐	Google A[illegible]	Google Inc.	3.2	13
☐	Google [illegible] Addon	Google Inc.	3.2	13
☑	Android 4.0	Android Open Source Project	4.0	14
☐	Google APIs	Google Inc.	4.0	14
☐	Android 4.0.3	Android Open Source Project	4.0.3	15
☐	Google APIs	Google Inc.	4.0.3	15
☐	ICS_R2	Motorola Mobility, LLC.	4.0.3	15
☐	Android 4.1.2	Android Open Source Project	4.1.2	16
☐	Google APIs	Google Inc.	4.1.2	16
☐	Android 4.2.2	Android Open Source Project	4.2.2	17
☐	Google APIs	Google Inc.	4.2.2	17
☐	Android 4.3	Android Open Source Project	4.3	18
☐	Google APIs	G[illegible]	[illegible]	[illegible]8

Library

☐ Is Library

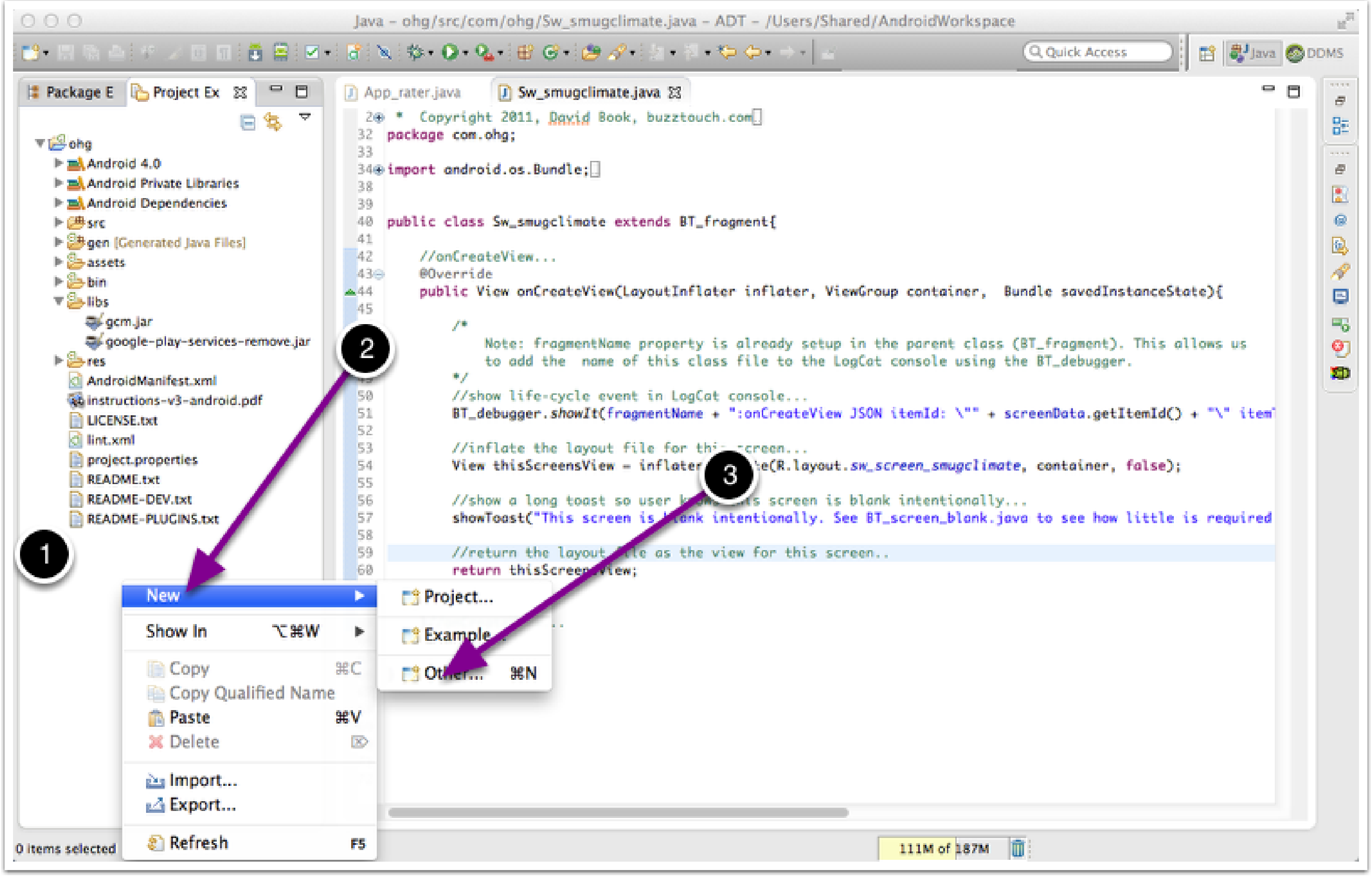
Reference	Project
✓ ../../Documents/google-play-services_lib	

David has explained to us the migration of Android Technology with respect to Google. No longer are some google services (like Maps) included 'by default' with your android project. Therefore you must reference the Google Play Services library within your project.

But how?

Answer: Like any other Android project. Sort of. I was a bit confused at first, but now I understand, and you can too. Unless you already did.

1) Add the Google Play Services Project to your Eclipse Workspace



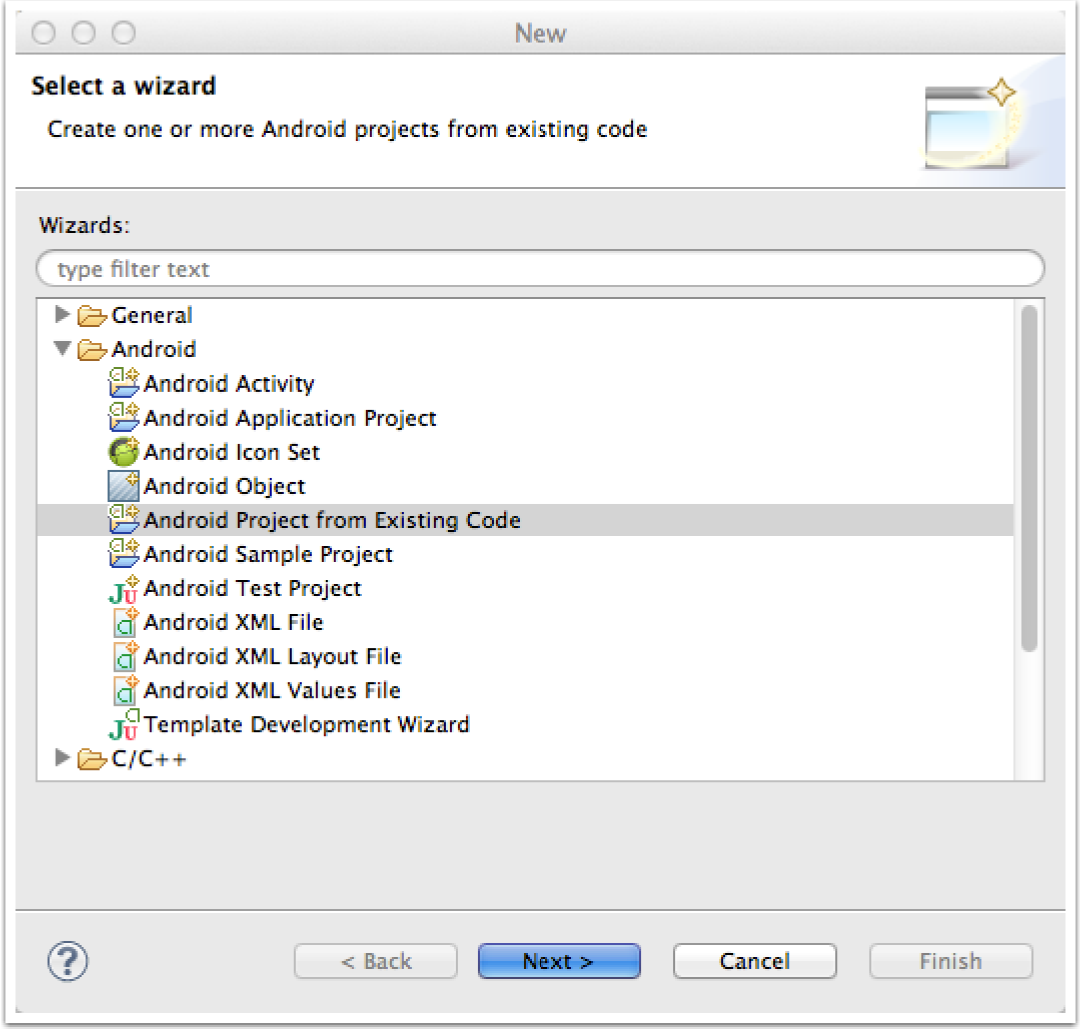
```
 * Copyright 2011, David Book, buzztouch.com
package com.ohg;

import android.os.Bundle;


public class Sw_smugclimate extends BT_fragment{

    //onCreateView...
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,  Bundle savedInstanceState){

        /*
            Note: fragmentName property is already setup in the parent class (BT_fragment). This allows us
            to add the  name of this class file to the LogCat console using the BT_debugger.
        */
        //show life-cycle event in LogCat console...
        BT_debugger.showIt(fragmentName + ":onCreateView JSON itemId: \"" + screenData.getItemId() + "\" item

        //inflate the layout file for this screen...
        View thisScreensView = inflater.inflate(R.layout.sw_screen_smugclimate, container, false);

        //show a long toast so user knows this screen is blank intentionally...
        showToast("This screen is blank intentionally. See BT_screen_blank.java to see how little is required

        //return the layout file as the view for this screen..
        return thisScreensView;
```

Before you start, you should have already added your BT Android project in your Eclipse 'Work Space'. (In the example, we're using my project "OHG")

For the most part, adding the GPS Library is like adding any other project into your workspace...

1) Right click within your Eclipse Project Explorer
2) Select "New"
3) Select "Other"

2) Select "Android Project from Existing Code"



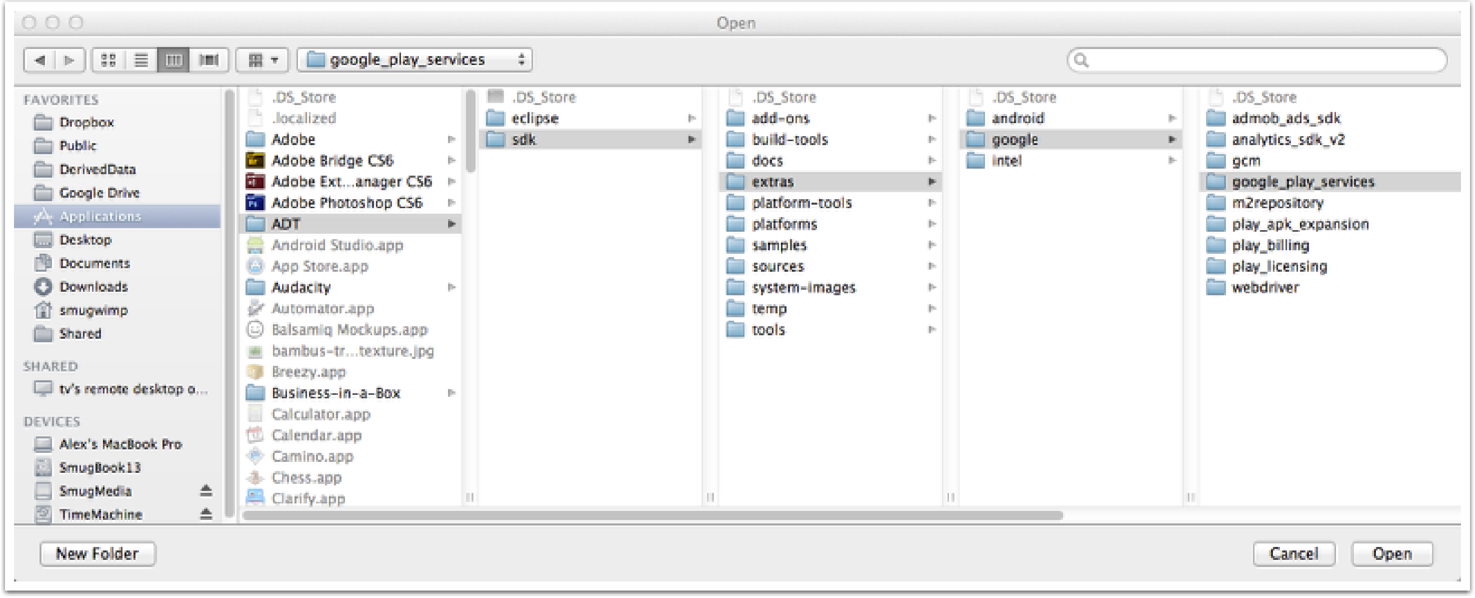
Select a wizard

Create one or more Android projects from existing code

Select Android Project from Existing Code

By the way, you can also do this via the Menu. Choose Menu "File" then "New" then "Other", or "Import", depending on your familiarity. If you're used to another method to add a project to the workspace, it's probably just as effective or efficient. The main goal right now is just add the project to the workspace, and there is more than one way to accomplish this goal.

3) Navigate to the Google Play Services Project



Navigate to your Google Play Services directory. It's included in the 'extras' of your ADT installation. Usually. This also assumes that you've downloaded it via the Android SDK Manager. If you don't have a Google Play Services directory, you'll need to launch your Android SDK Manager, scroll down to the bottom of your choices, and under 'extras' choose 'Google Play Services'. When you've finished the installation, you can continue...

On the Macintosh, this project is located in your Applications Folder, with your ADT install.

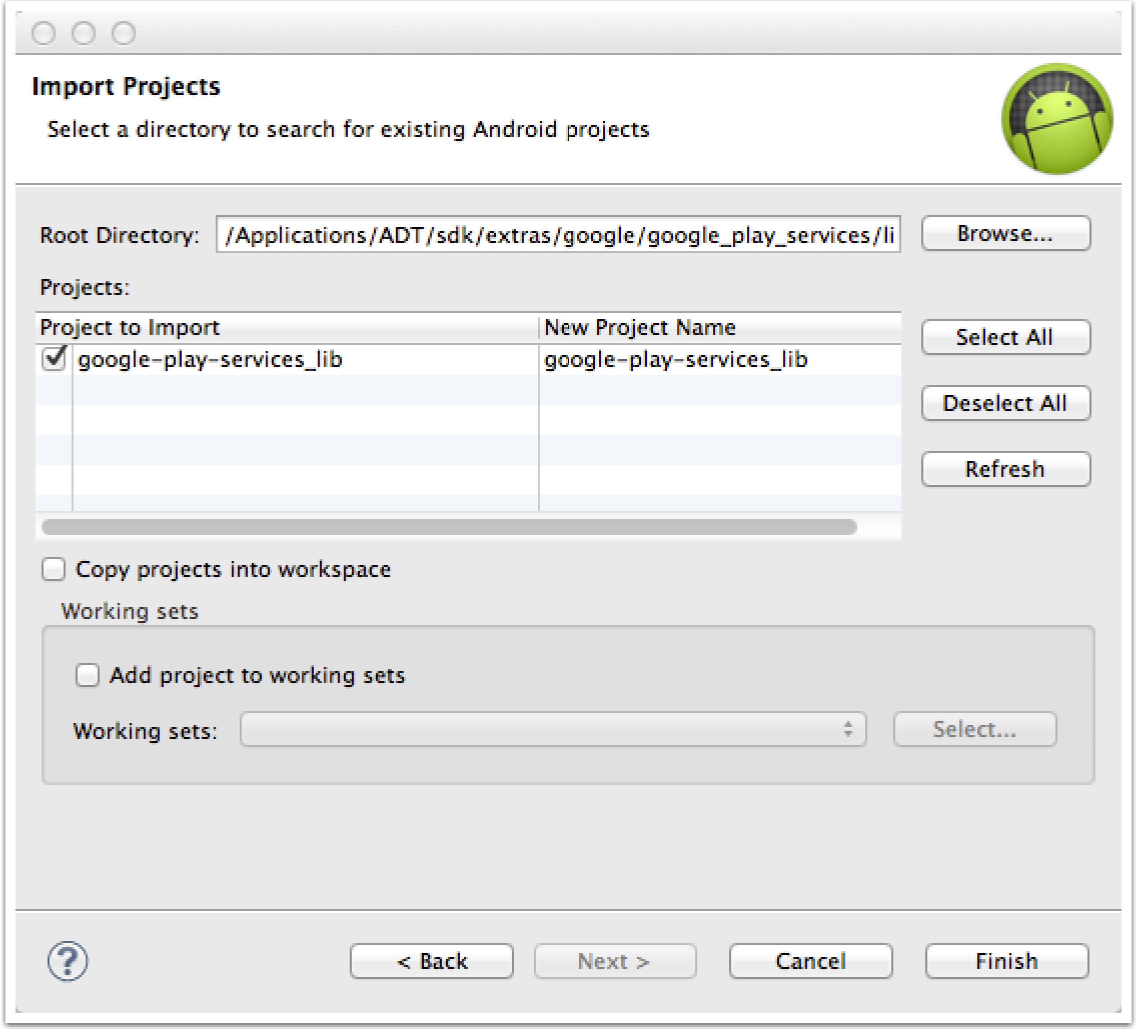
Applications > ADT > sdk > extras > google > google_play_services

In Windows it should be located:

<android-sdk>/extras/google/google_play_services/

Select that directory the way you would any project you're adding to the workspace.

4) Select "Google Play Services"



Import Projects

Select a directory to search for existing Android projects

Root Directory: /Applications/ADT/sdk/extras/google/google_play_services/li

Projects:

Project to Import	New Project Name
☑ google-play-services_lib	google-play-services_lib

□ Copy projects into workspace

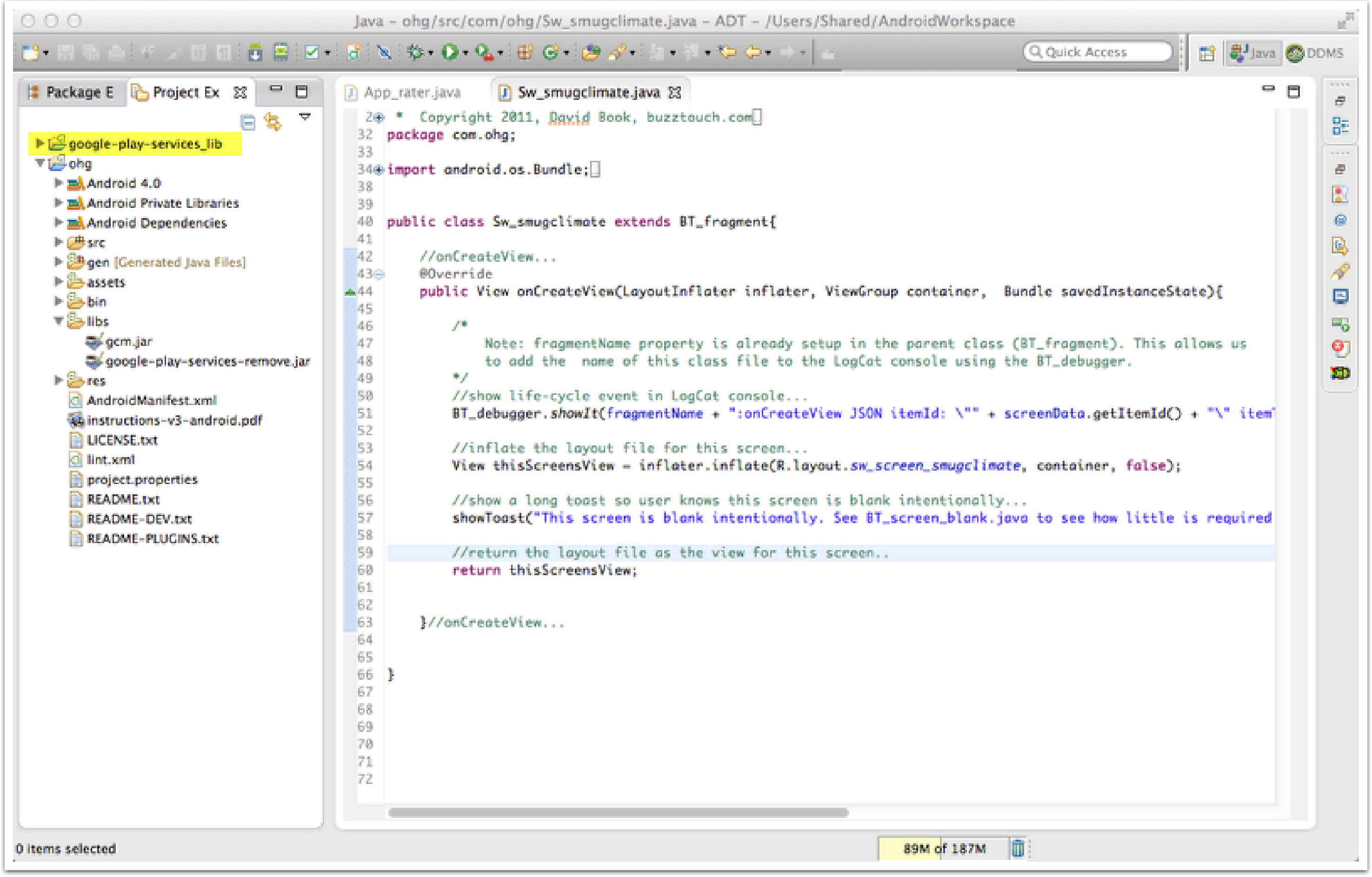
Working sets

□ Add project to working sets

Working sets:

Navigate to the Google Play Services directory, and select "Finish"

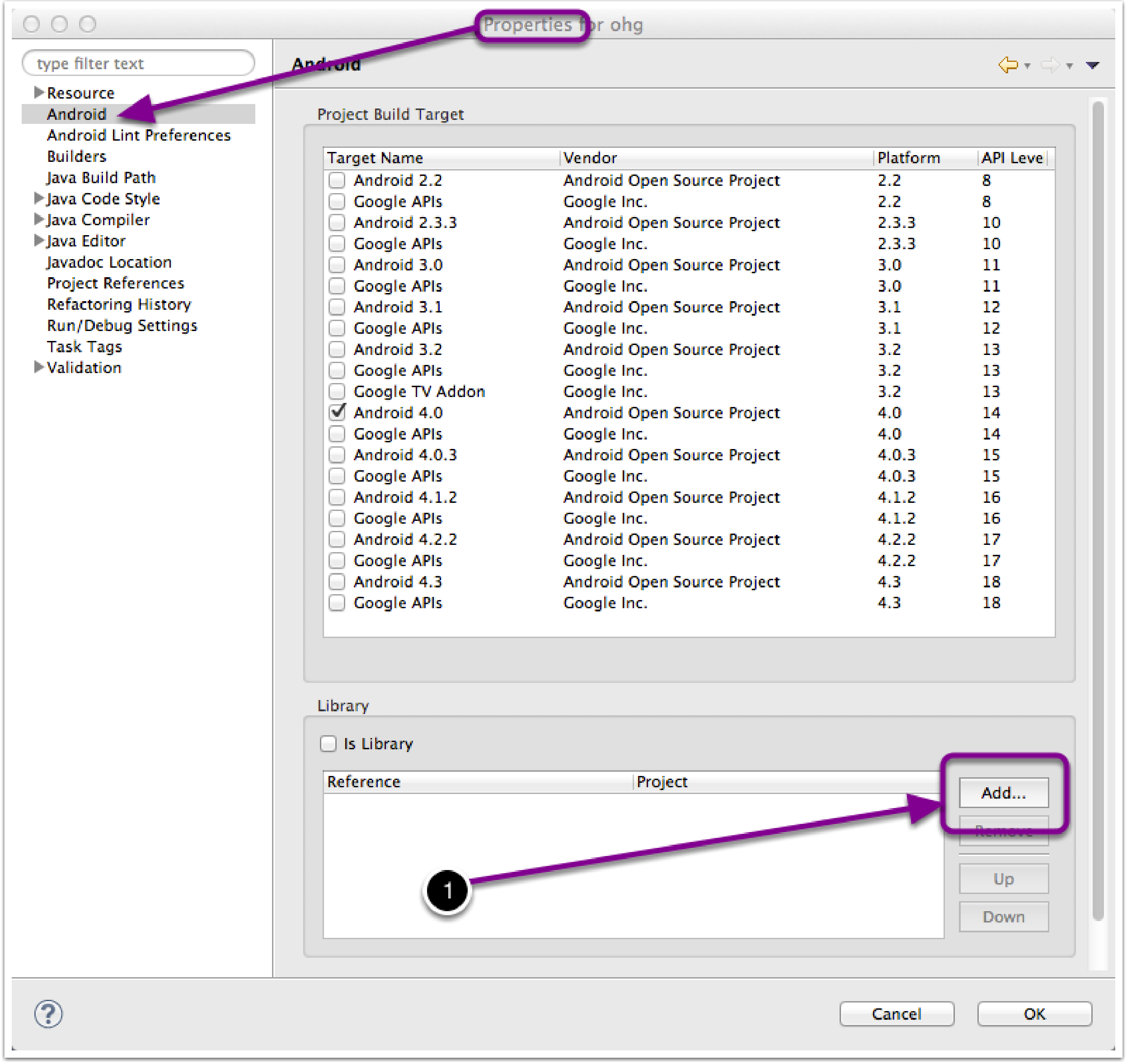
5) Confirm it's existance in your Workspace



```
 *  Copyright 2011, David Book, buzztouch.com
package com.ohg;

import android.os.Bundle;


public class Sw_smugclimate extends BT_fragment{

    //onCreateView...
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,  Bundle savedInstanceState){

        /*
            Note: fragmentName property is already setup in the parent class (BT_fragment). This allows us
            to add the  name of this class file to the LogCat console using the BT_debugger.
        */
        //show life-cycle event in LogCat console...
        BT_debugger.showIt(fragmentName + ":onCreateView JSON itemId: \"" + screenData.getItemId() + "\" item

        //inflate the layout file for this screen...
        View thisScreensView = inflater.inflate(R.layout.sw_screen_smugclimate, container, false);

        //show a long toast so user knows this screen is blank intentionally...
        showToast("This screen is blank intentionally. See BT_screen_blank.java to see how little is required

        //return the layout file as the view for this screen..
        return thisScreensView;


    }//onCreateView...


}
```

The position may be different, depending on the spelling of your project. It seems to be alphabetical. But you want to see that project in your workspace, and you don't want to see any red.

6) Reference the Library within your project



Project Build Target

	Target Name	Vendor	Platform	API Leve
☐	Android 2.2	Android Open Source Project	2.2	8
☐	Google APIs	Google Inc.	2.2	8
☐	Android 2.3.3	Android Open Source Project	2.3.3	10
☐	Google APIs	Google Inc.	2.3.3	10
☐	Android 3.0	Android Open Source Project	3.0	11
☐	Google APIs	Google Inc.	3.0	11
☐	Android 3.1	Android Open Source Project	3.1	12
☐	Google APIs	Google Inc.	3.1	12
☐	Android 3.2	Android Open Source Project	3.2	13
☐	Google APIs	Google Inc.	3.2	13
☐	Google TV Addon	Google Inc.	3.2	13
☑	Android 4.0	Android Open Source Project	4.0	14
☐	Google APIs	Google Inc.	4.0	14
☐	Android 4.0.3	Android Open Source Project	4.0.3	15
☐	Google APIs	Google Inc.	4.0.3	15
☐	Android 4.1.2	Android Open Source Project	4.1.2	16
☐	Google APIs	Google Inc.	4.1.2	16
☐	Android 4.2.2	Android Open Source Project	4.2.2	17
☐	Google APIs	Google Inc.	4.2.2	17
☐	Android 4.3	Android Open Source Project	4.3	18
☐	Google APIs	Google Inc.	4.3	18

Library

☐ Is Library

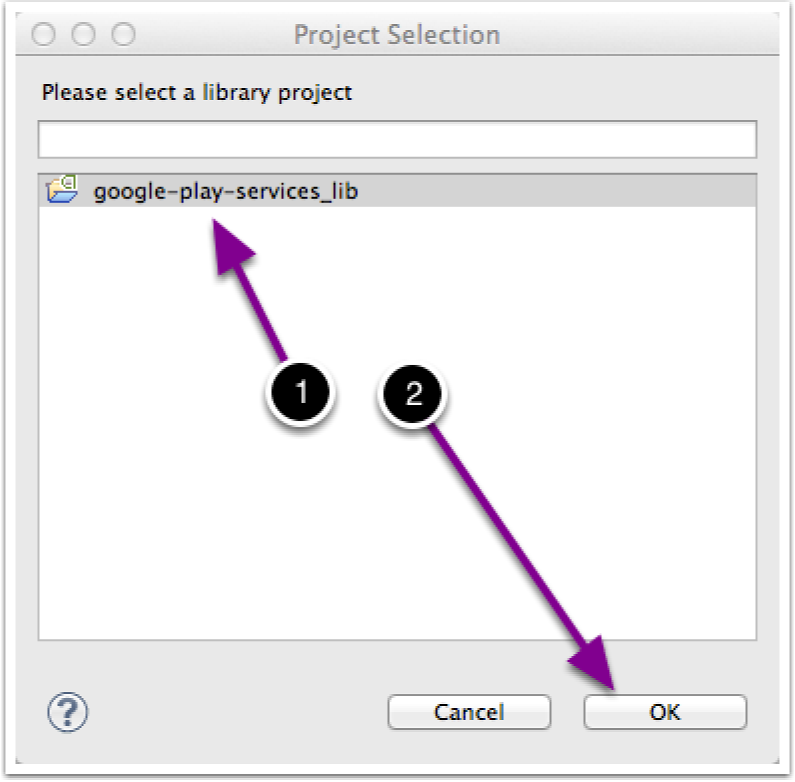
Reference	Project

Reference the newly added workspace project (namely, Google Play Services) by going to your project properties (the same place you set your API Level)

Right click on 'your' BT project. Choose 'Properties'. When the properties dialog appears, choose 'Android'

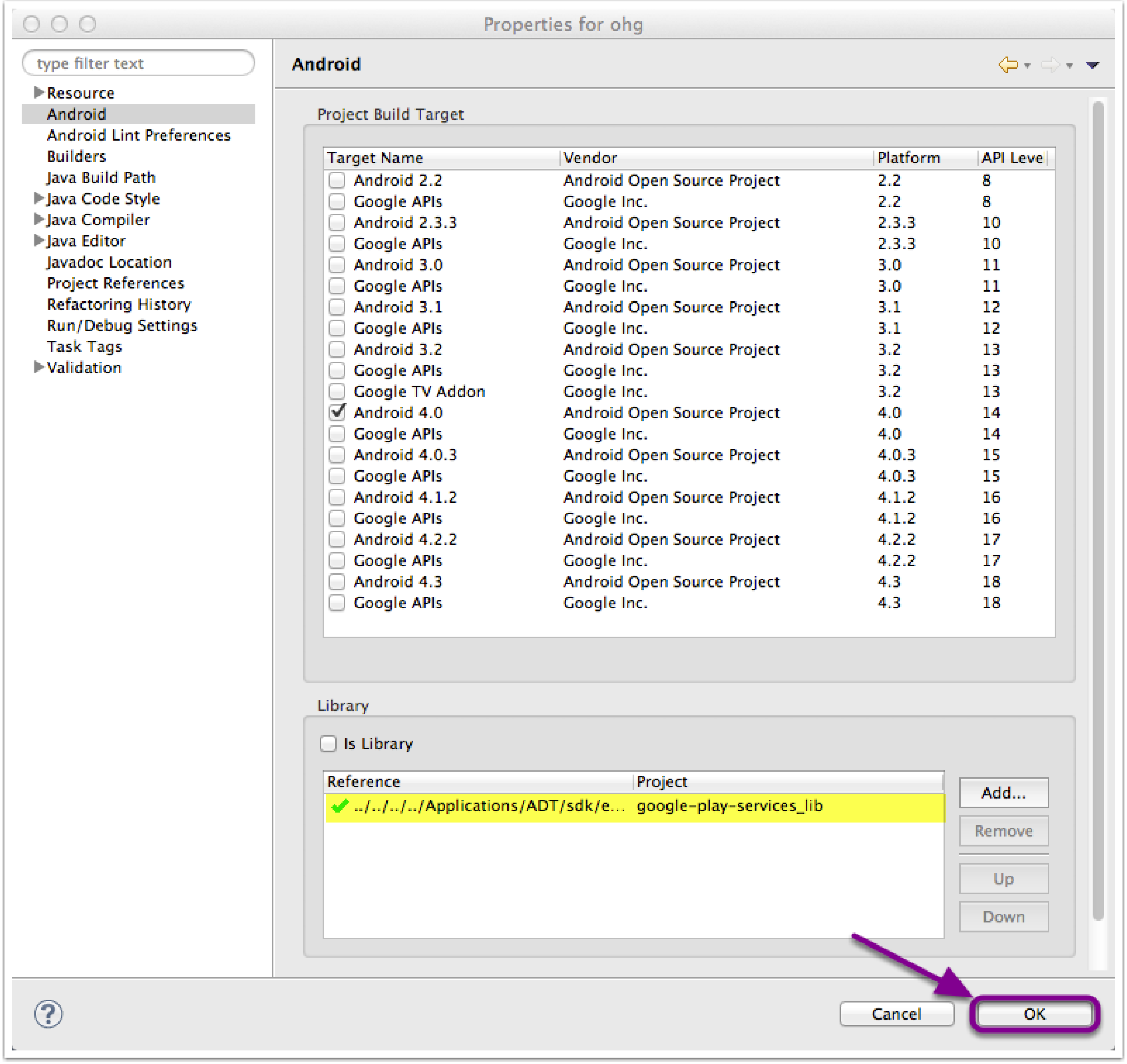
1) In the lower portion of the window display, choose 'Add...' from the library section.

Selecting the google-play-services library



1) In the project selection window, choose 'Google Play Services'. This won't be seen until you add the Google Play Services project to your workspace.
2) Press OK

Confirming Addition of Google-Play-Services



Project Build Target

Target Name	Vendor	Platform	API Leve
□ Android 2.2	Android Open Source Project	2.2	8
□ Google APIs	Google Inc.	2.2	8
□ Android 2.3.3	Android Open Source Project	2.3.3	10
□ Google APIs	Google Inc.	2.3.3	10
□ Android 3.0	Android Open Source Project	3.0	11
□ Google APIs	Google Inc.	3.0	11
□ Android 3.1	Android Open Source Project	3.1	12
□ Google APIs	Google Inc.	3.1	12
□ Android 3.2	Android Open Source Project	3.2	13
□ Google APIs	Google Inc.	3.2	13
□ Google TV Addon	Google Inc.	3.2	13
☑ Android 4.0	Android Open Source Project	4.0	14
□ Google APIs	Google Inc.	4.0	14
□ Android 4.0.3	Android Open Source Project	4.0.3	15
□ Google APIs	Google Inc.	4.0.3	15
□ Android 4.1.2	Android Open Source Project	4.1.2	16
□ Google APIs	Google Inc.	4.1.2	16
□ Android 4.2.2	Android Open Source Project	4.2.2	17
□ Google APIs	Google Inc.	4.2.2	17
□ Android 4.3	Android Open Source Project	4.3	18
□ Google APIs	Google Inc.	4.3	18

Library

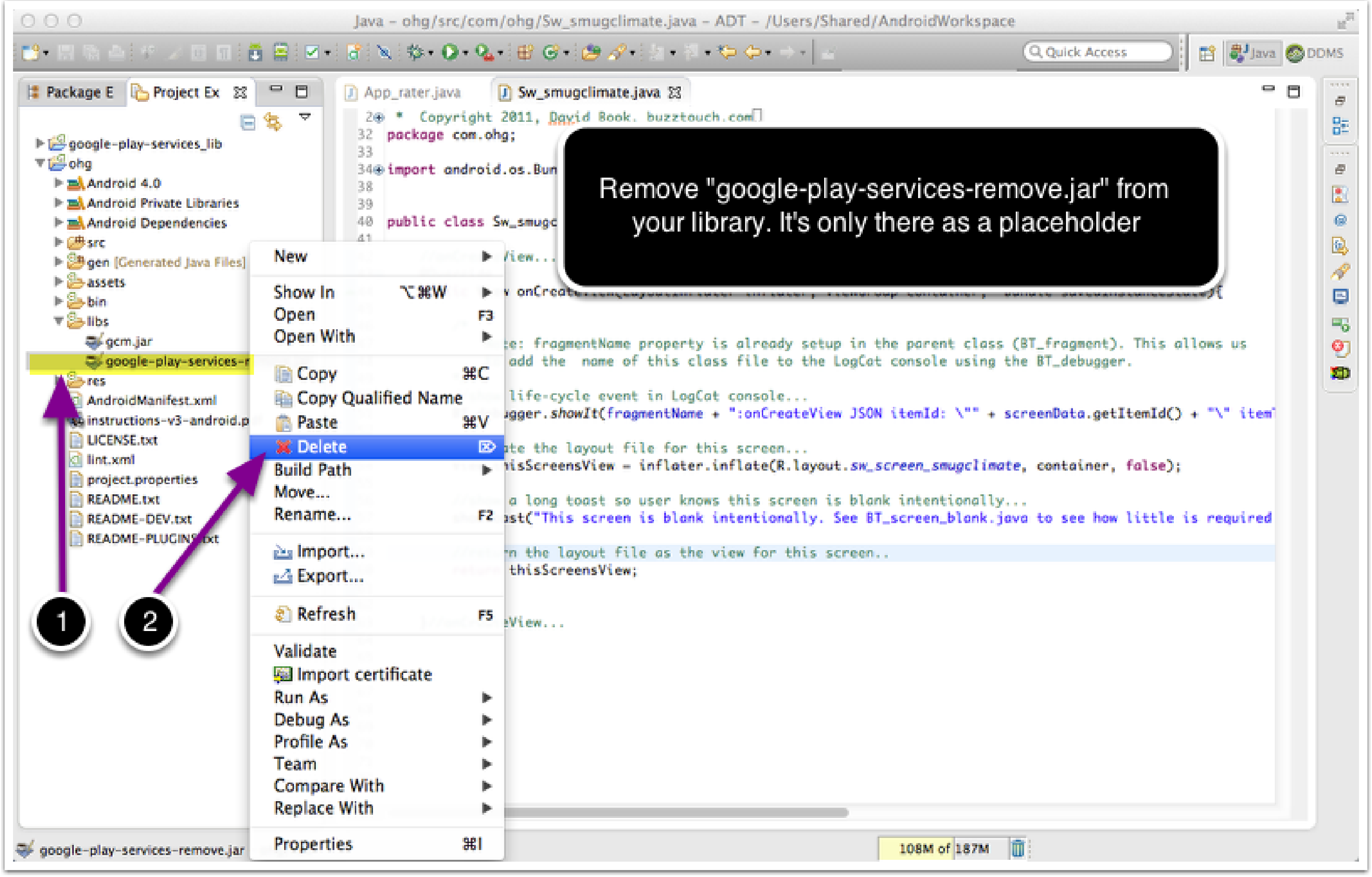
□ Is Library

Reference	Project
✓ ../../../../Applications/ADT/sdk/e...	google-play-services_lib

If available and added correctly, you'll see the Green Checkmark indicating a valid reference.

Press "OK" to return to your project IDE.

Remove 'google-play-services-remove.jar' from your project



```
 * Copyright 2011, David Book, buzztouch.com
package com.ohg;
import android.os.Bun
public class Sw_smugc
```

Remove "google-play-services-remove.jar" from your library. It's only there as a placeholder

1) Locate "google-play-services-remove.jar" in your project. It should be under "libs"
2) Right-Click and delete it.

The "google-play-services-remove.jar" is only a temporary place holder for the actual Google Play Services Library, which is not automatically added to your project. This is because like everything else, Google Play Services may be regularly updated, and to provide an antiquated SDK in your project isn't a good thing. So the google-play-services-remove.jar is merely a file that keeps your project from exploding in red (errors) before you have an opportunity to add the legitimate and current Google Play Services.

You should be done. Go forth and develop with confidence!

Cheers!

-- Smug