SBC/NSD/EC21 series Processor

EC21 / NSD21 / SBC21

Software User's Manual (Android)

RELEASE NOTES

Version	Date	NOTES
1.00	Mar. 2013	Initial release
1.20	Apr. 2013	Add FAQ
2.00	Sep. 2013	Add “Test Application“ Add “Download Firmware Image”
2.10	Oct. 2013	Modify env var for CEA HDMI (c600-sb-jb-b0)
3.00	Dec. 2013	Modify Chapter 6

Disclaimer

Table of Contents

Chapter 1: Introduction (Android 4.2)

This manual's content is mainly for guiding you how to use our products with Android system. It's including introduction of development tools, how to develop the application software and how to use graphical user interface for development. Furthermore, it also contains with descriptions and presentations of the application templates that this product attached.

Applicable products: SBC21 / EC21 / NSD21 series

If you need the hardware information about the above products, please refer to anther document – *Hardware User Manual*.

Features and introduction of Android:

- A Linux-based operating system.
- Open source.
- Designed primarily for touchscreen mobile.
- Designed to manage memory (RAM) to keep power consumption at a minimum.
- Support higher resolution, 2D display, 3D display, multi-language, Network, web browser, JAVA, music and media, photos and videos etc.

Chapter 2: Application Development Environment

2.1 System Requirement

There are referred to Android SDK system requirements, in Androird Developers.

Operating Systems

- Windows XP 32-bit / Vista 32- or 64-bit / Windows 7 32- or 64-bit
- Mac OS X 10.5.8 or later (x86 only)
- Linux (tested on Ubuntu Linux, Lucid Lynx)

  - GNU C Library (glibc) 2.7 or later is required.
  - On Ubuntu Linux, version 8.04 or later is required.
  - 64-bit distributions must be capable of running 32-bit applications.

Eclipse IDE

- Java Development Kit (JDK) 6
- Eclipse 3.6.2 or greater
- Eclipse JDT plugin (included in most Eclipse IDE packages)
- Android Development Tools plugin (recommended)

Other development environments

- Android API Level 17

Note:
- Eclipse 3.5 (Galileo) is no longer supported with the latest version of ADT.
- Some Linux distributions may include JDK 1.4 or Gnu Compiler for Java, both of which are not supported for Android development.

The following contents are operating in ***Windows*** as example.

2.2 Install JAVA (JDK)

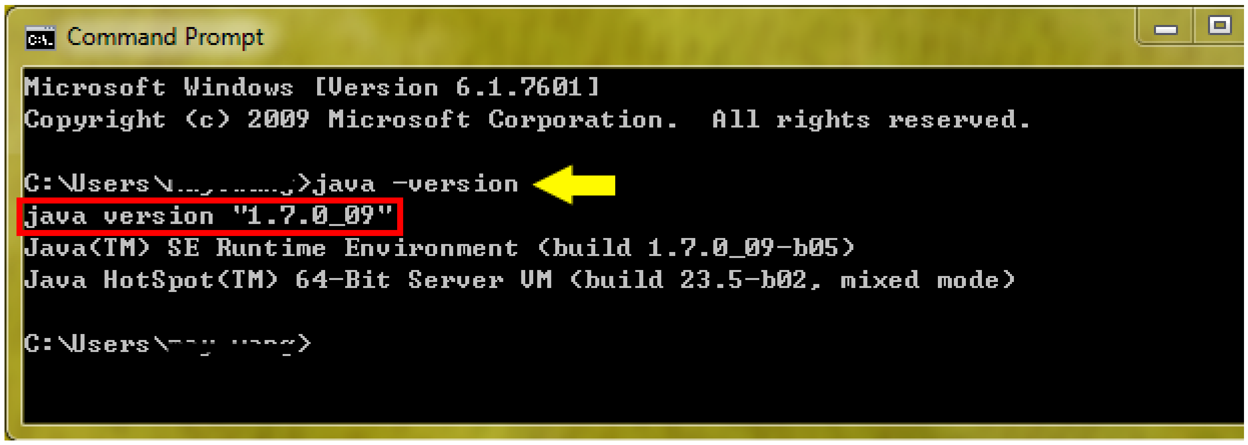
Eclipse is a Java-based application, so you need to install a JVM – **JDK** which includes complete *JRE* Java runtime environment, *javac* Java compiler, *jar* archiver, *javadoc* documentation generator, *jdb* debugger…etc.

Download and install JDK:

http://www.oracle.com/technetwork/java/javase/downloads/index.html

You can choose JDK 7 or JDK 6 (recommended).
After installation completes, you can check whether the Java Platform is ready in your computer.

Run **Command Prompt** and keyin **`java -version`**



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\[illegible]>java -version
java version "1.7.0_09"
Java(TM) SE Runtime Environment (build 1.7.0_09-b05)
Java HotSpot(TM) 64-Bit Server VM (build 23.5-b02, mixed mode)

C:\Users\[illegible]>
```

- If it show java version "1.5.X_XX" refers you have installed JDK 5.
- If it show java version "1.6.X_XX" refers you have installed JDK 6.
- If it show java version "1.7.X_XX" refers you have installed JDK 7.

Your java version needs JDK 6 at least; otherwise you should re-install JDK 6.

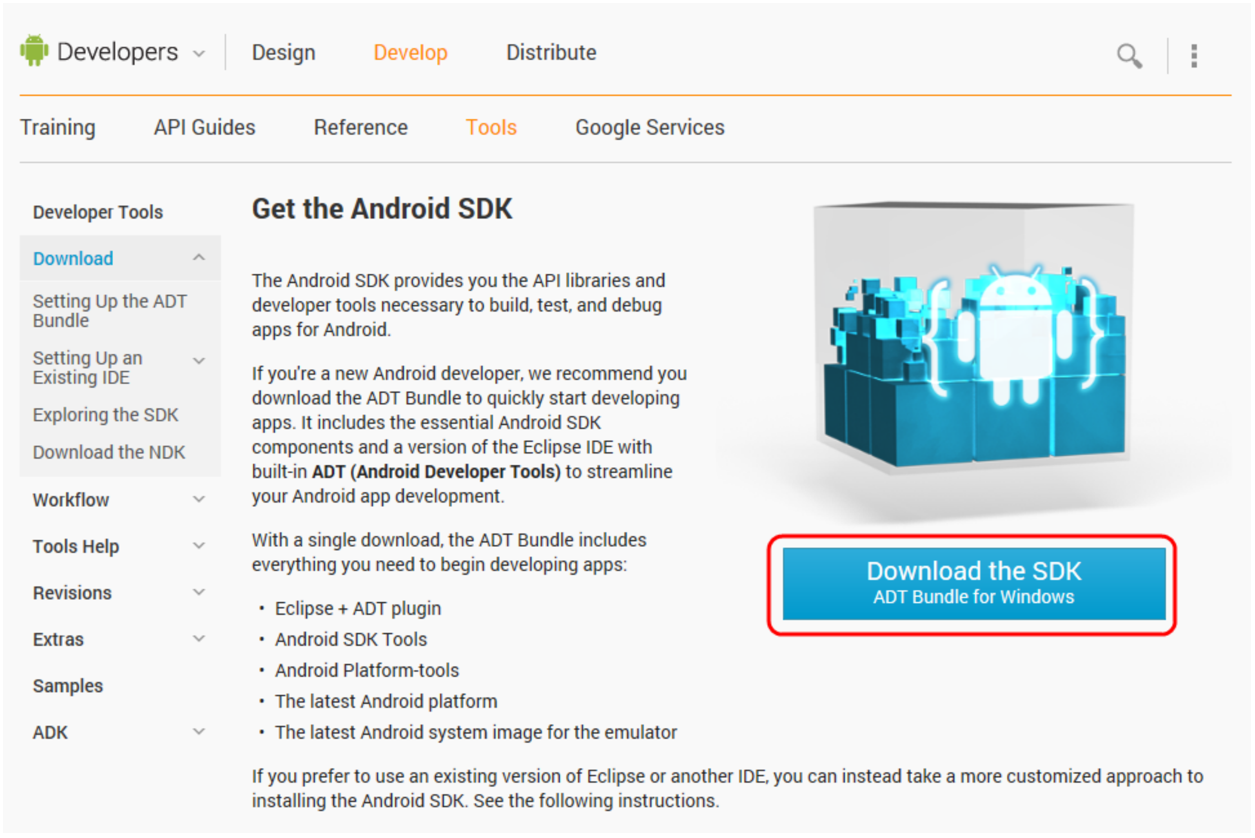
2.3 Install Android SDK

For developing Android, prepare the Android software development kit is necessary. Except JDK, there is something you also need to download and install:

- ✓ Eclipse IDE - http://www.eclipse.org/downloads/
- ✓ Android Developer Tools - http://developer.android.com/sdk/installing/installing-adt.html
- ✓ Android SDK Tool - http://developer.android.com/sdk/exploring.html#UpdatingComponents

Of course you can download and install one by one from those official or resource websites (see the link above). But now there is a method that you can quickly set up the development environment for a new Android developer. If you are the first time to developing Android, try it. Go to the Android Developers website and download the ADT Bundle that almost includes everything you need to begin developing app.

1. Go http://developer.android.com/sdk/index.html. There is an existing ADT Bundel for Windows.



Get the Android SDK

The Android SDK provides you the API libraries and developer tools necessary to build, test, and debug apps for Android.

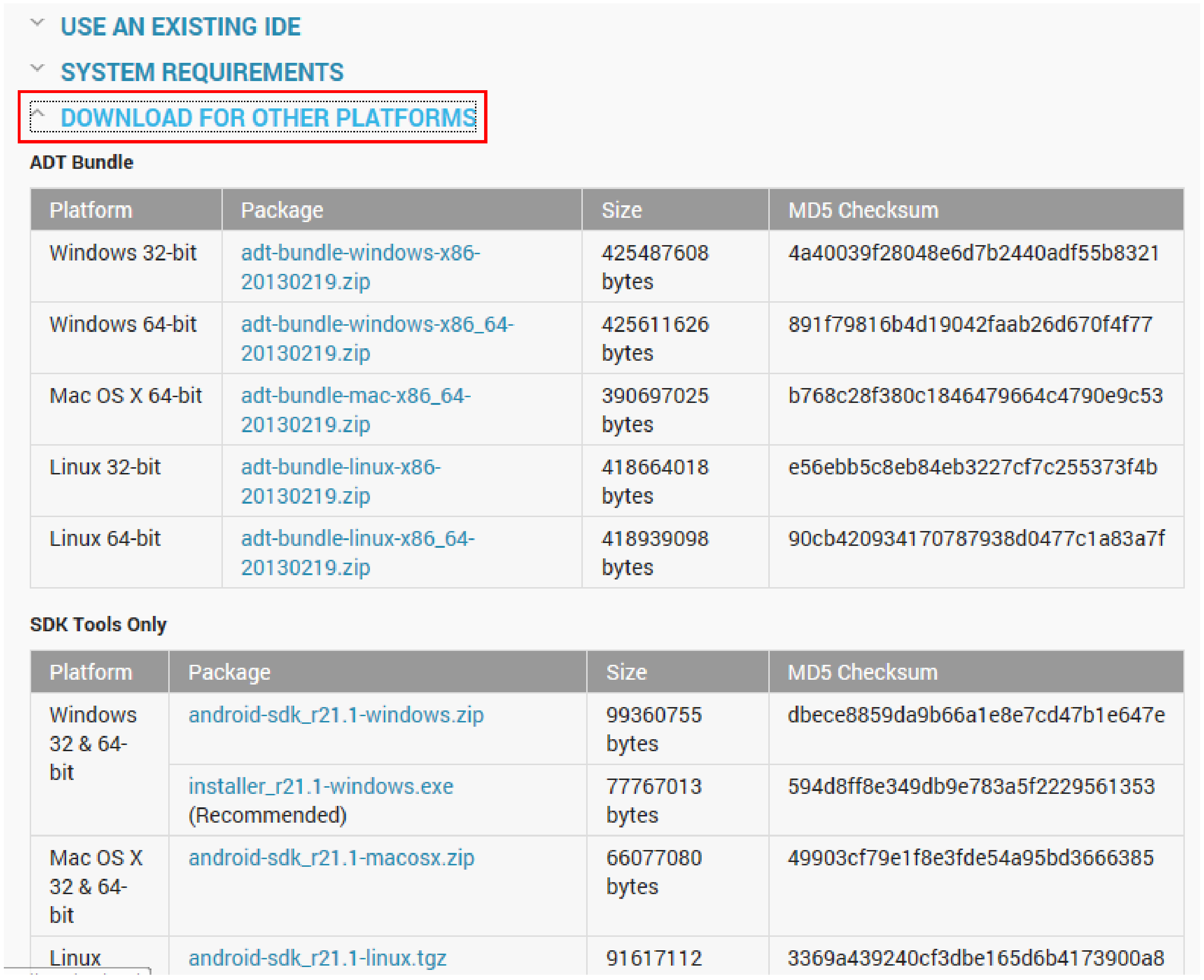
If you're a new Android developer, we recommend you download the ADT Bundle to quickly start developing apps. It includes the essential Android SDK components and a version of the Eclipse IDE with built-in **ADT (Android Developer Tools)** to streamline your Android app development.

With a single download, the ADT Bundle includes everything you need to begin developing apps:

- Eclipse + ADT plugin
- Android SDK Tools
- Android Platform-tools
- The latest Android platform
- The latest Android system image for the emulator

If you prefer to use an existing version of Eclipse or another IDE, you can instead take a more customized approach to installing the Android SDK. See the following instructions.

2. If your development platform is different, see the below and choose appropriate package for your platform.



USE AN EXISTING IDE

SYSTEM REQUIREMENTS

DOWNLOAD FOR OTHER PLATFORMS

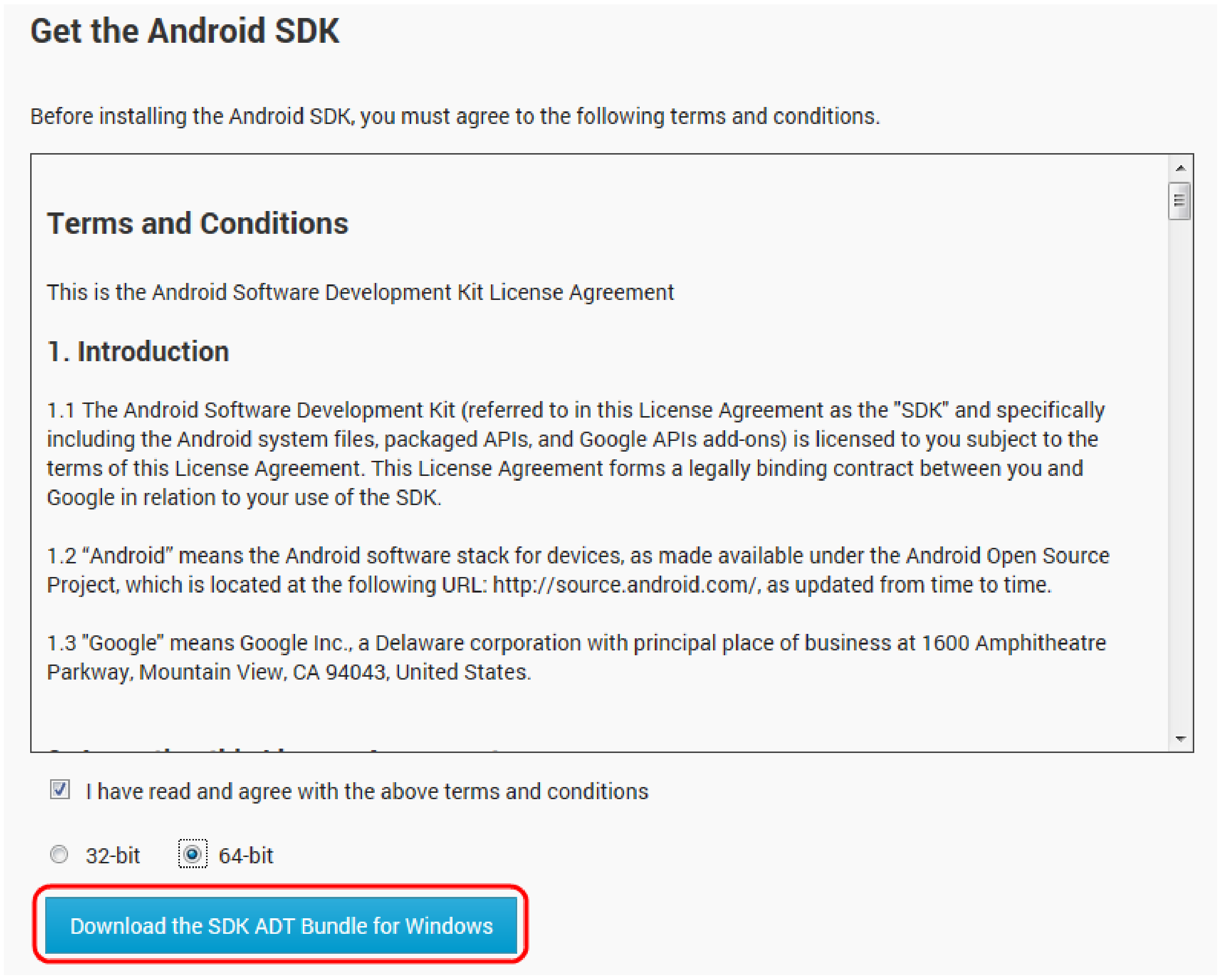
ADT Bundle

Platform	Package	Size	MD5 Checksum
Windows 32-bit	adt-bundle-windows-x86-20130219.zip	425487608 bytes	4a40039f28048e6d7b2440adf55b8321
Windows 64-bit	adt-bundle-windows-x86_64-20130219.zip	425611626 bytes	891f79816b4d19042faab26d670f4f77
Mac OS X 64-bit	adt-bundle-mac-x86_64-20130219.zip	390697025 bytes	b768c28f380c1846479664c4790e9c53
Linux 32-bit	adt-bundle-linux-x86-20130219.zip	418664018 bytes	e56ebb5c8eb84eb3227cf7c255373f4b
Linux 64-bit	adt-bundle-linux-x86_64-20130219.zip	418939098 bytes	90cb420934170787938d0477c1a83a7f

SDK Tools Only

Platform	Package	Size	MD5 Checksum
Windows 32 & 64-bit	android-sdk_r21.1-windows.zip	99360755 bytes	dbece8859da9b66a1e8e7cd47b1e647e
	installer_r21.1-windows.exe (Recommended)	77767013 bytes	594d8ff8e349db9e783a5f2229561353
Mac OS X 32 & 64-bit	android-sdk_r21.1-macosx.zip	66077080 bytes	49903cf79e1f8e3fde54a95bd3666385
Linux	android-sdk_r21.1-linux.tgz	91617112	3369a439240cf3dbe165d6b4173900a8

3. Accept the license, select your system type and then download the SDK ADT Bundle.



Get the Android SDK

Before installing the Android SDK, you must agree to the following terms and conditions.

Terms and Conditions

This is the Android Software Development Kit License Agreement

1. Introduction

1.1 The Android Software Development Kit (referred to in this License Agreement as the "SDK" and specifically including the Android system files, packaged APIs, and Google APIs add-ons) is licensed to you subject to the terms of this License Agreement. This License Agreement forms a legally binding contract between you and Google in relation to your use of the SDK.

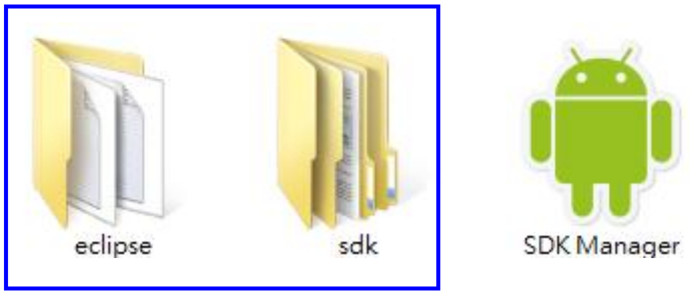
1.2 "Android" means the Android software stack for devices, as made available under the Android Open Source Project, which is located at the following URL: http://source.android.com/, as updated from time to time.

1.3 "Google" means Google Inc., a Delaware corporation with principal place of business at 1600 Amphitheatre Parkway, Mountain View, CA 94043, United States.

☑ I have read and agree with the above terms and conditions

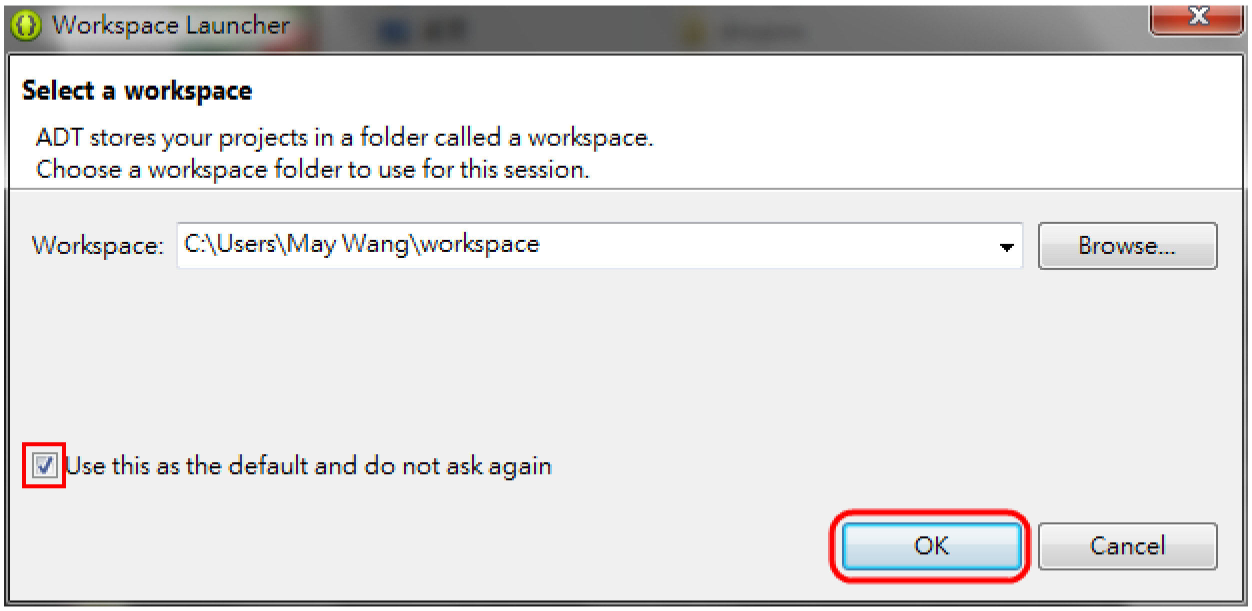
32-bit [✓ 64-bit]

4. The file you download is a zip file including an eclipse folder, sdk folder. Decompress the file to your computer (for example, in C:\).



5. Now, open the eclipse folder and double-click eclipse.exe to start.

>>>



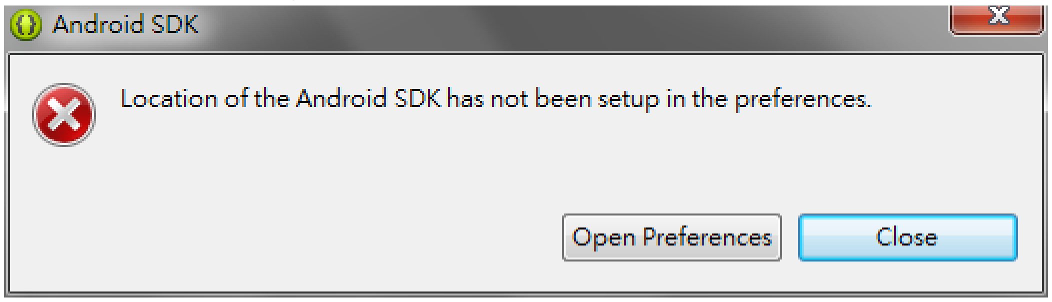
Select a workspace

ADT stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\Users\May Wang\workspace

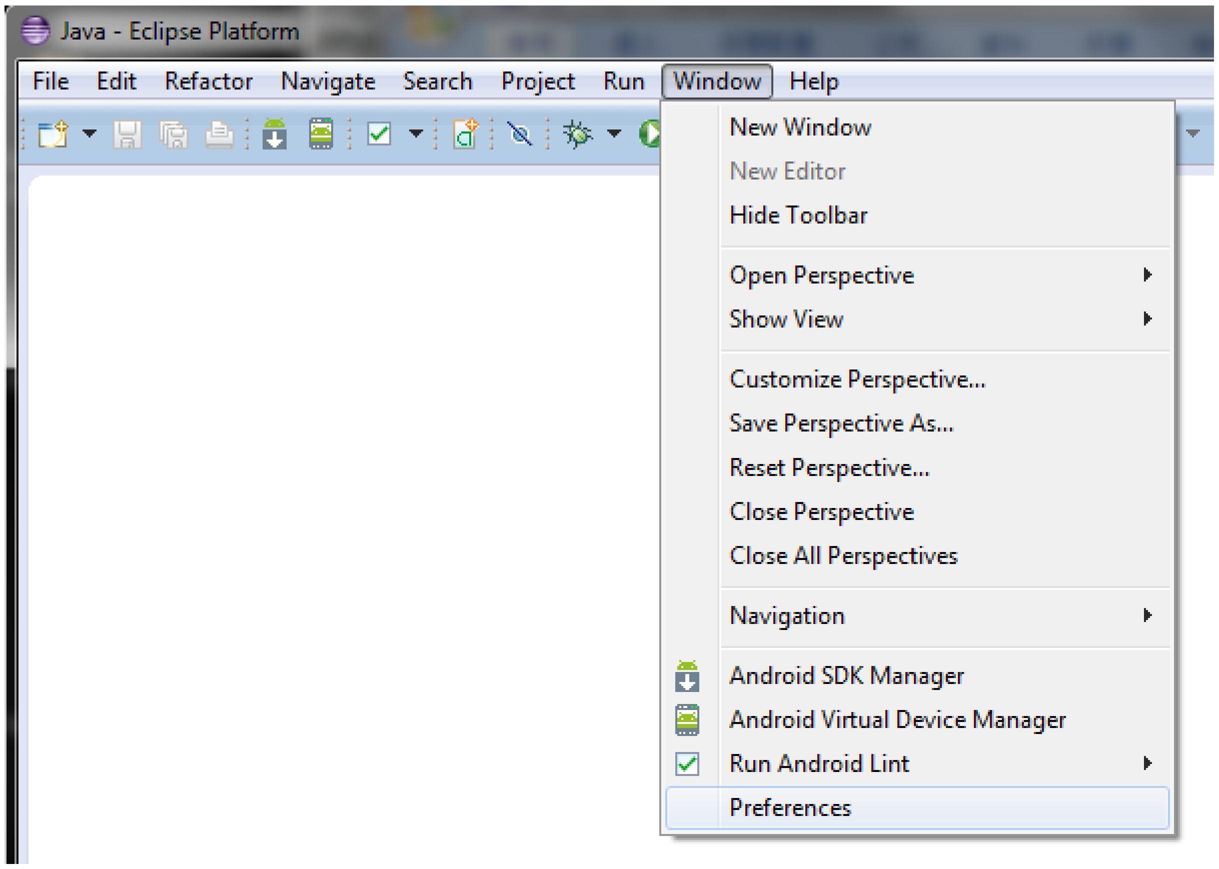
[✓] Use this as the default and do not ask again

6. If it show the below dialog, click **Close**.

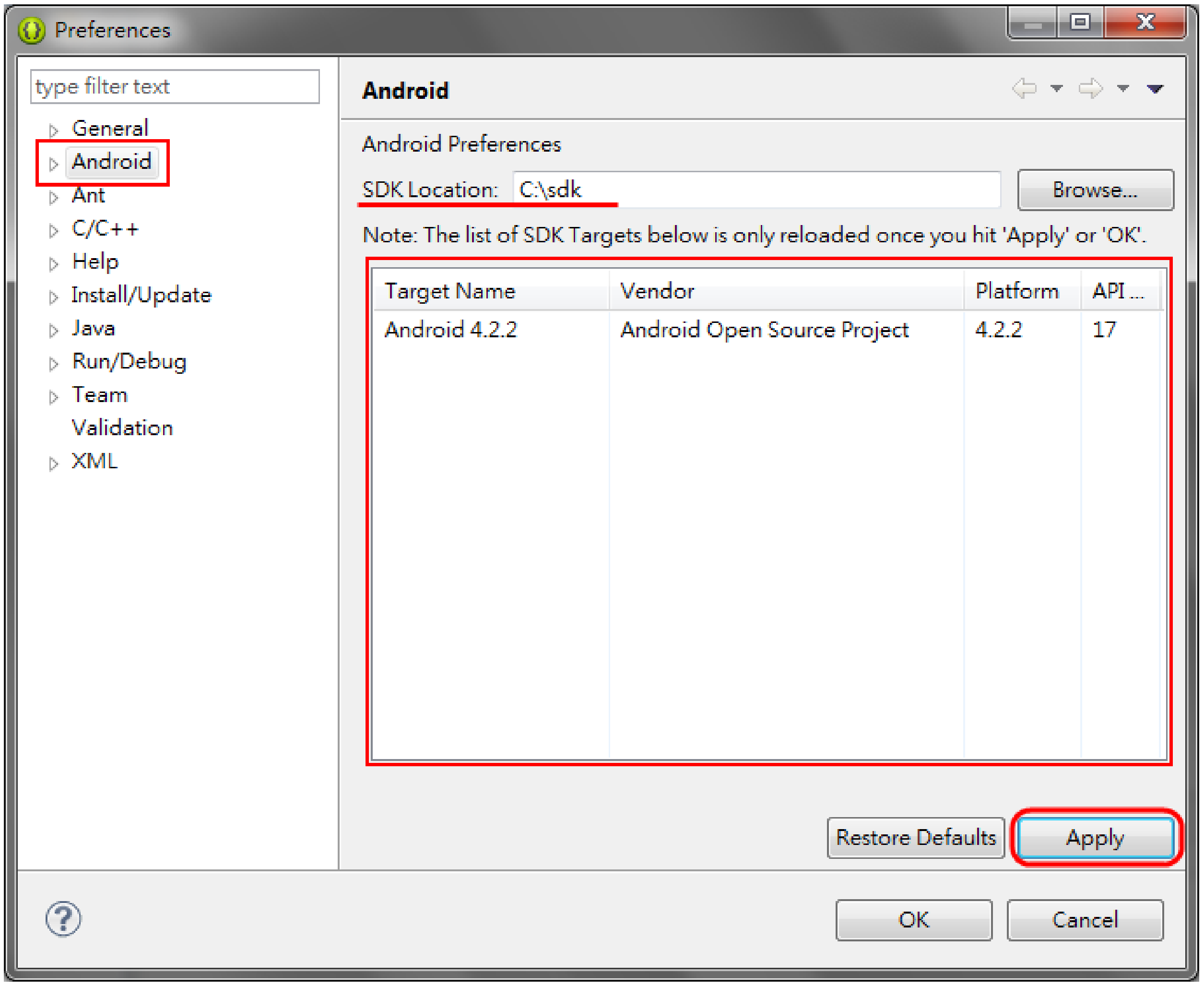


Location of the Android SDK has not been setup in the preferences.

7. Go **Window → Preferences**



8. Select **Android** → Edit the text of **SDK Location** or click the **Browse…** to select your Android SDK folder you just decompress. Click **Apply** and the middle would show the list of SDK Target you have.



Android

Android Preferences

SDK Location: C:\sdk

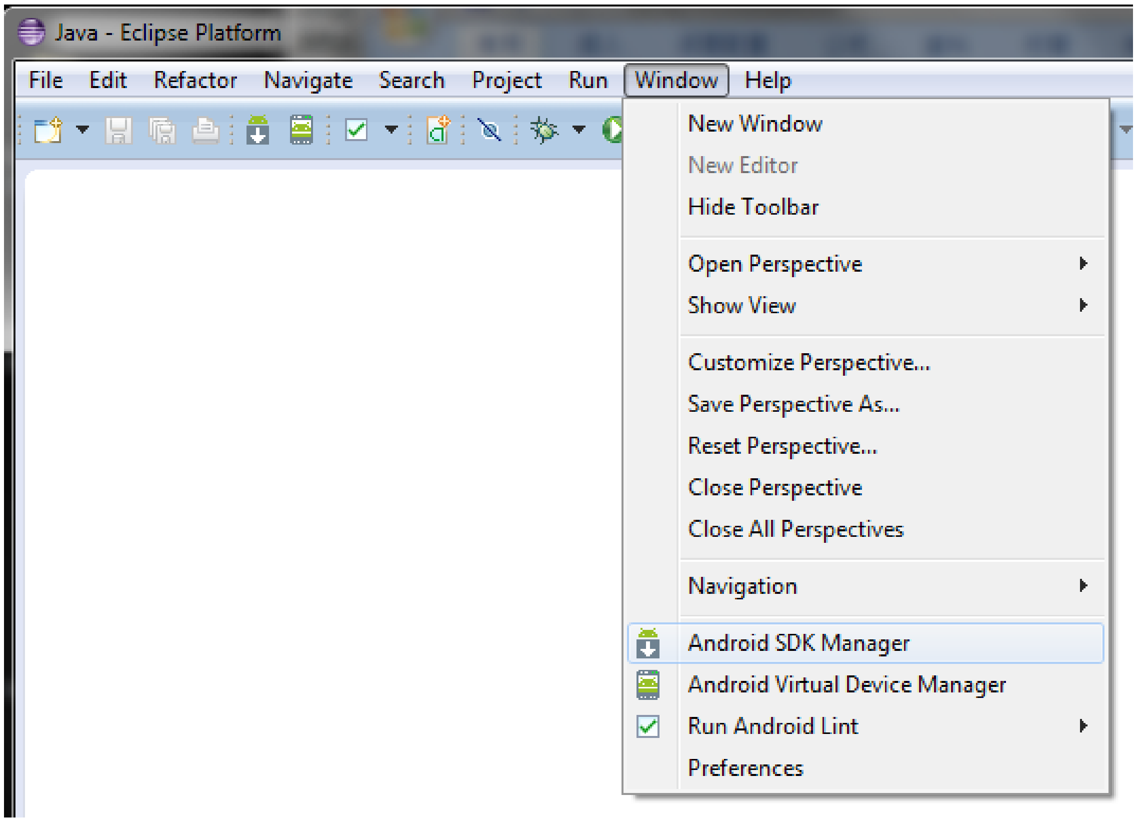
Note: The list of SDK Targets below is only reloaded once you hit 'Apply' or 'OK'.

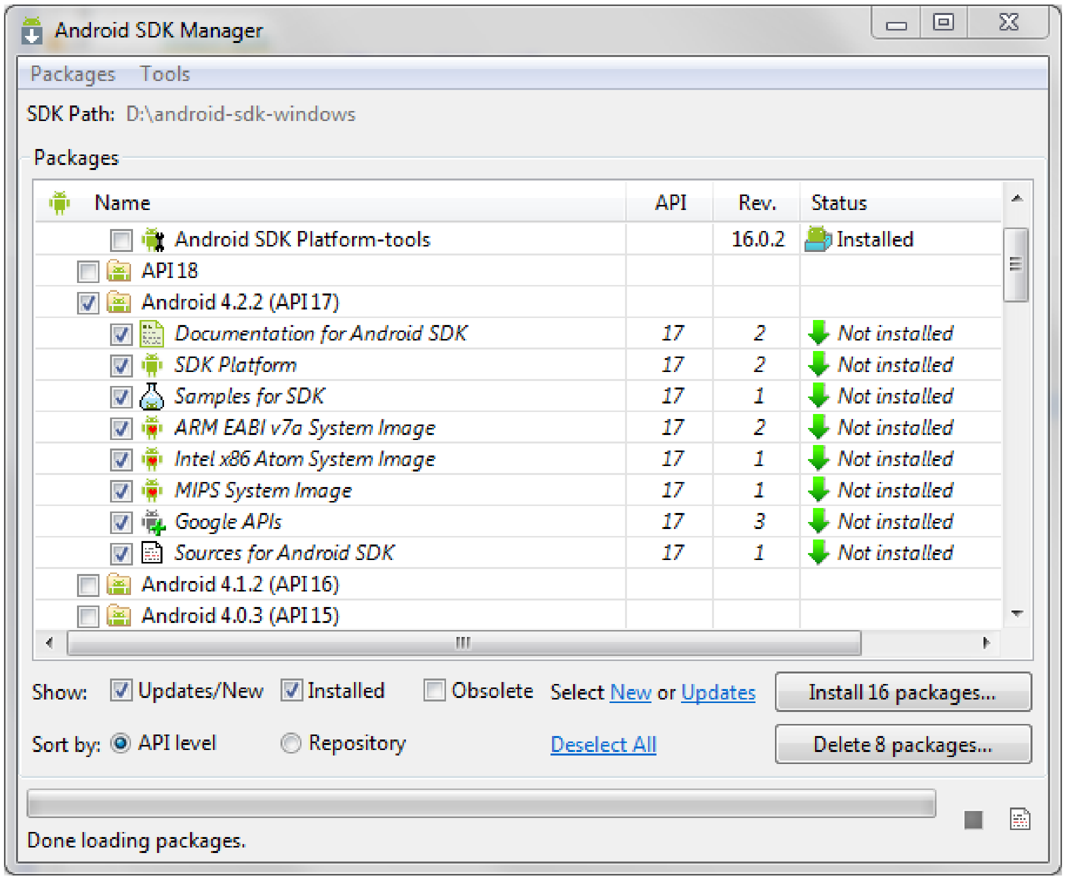
Target Name	Vendor	Platform	API ...
Android 4.2.2	Android Open Source Project	4.2.2	17

2.4 Install Android API Level 17 (For Android 4.2)

Next, for this case, we need more SDK Platform-tools for other Android platform. You can download then via using the Android SDK Manager.

1. Launch the SDK Manager. In Eclipse, select **Window → Android SDK Manager**.





Android SDK Manager

Packages Tools

SDK Path: D:\android-sdk-windows

Packages

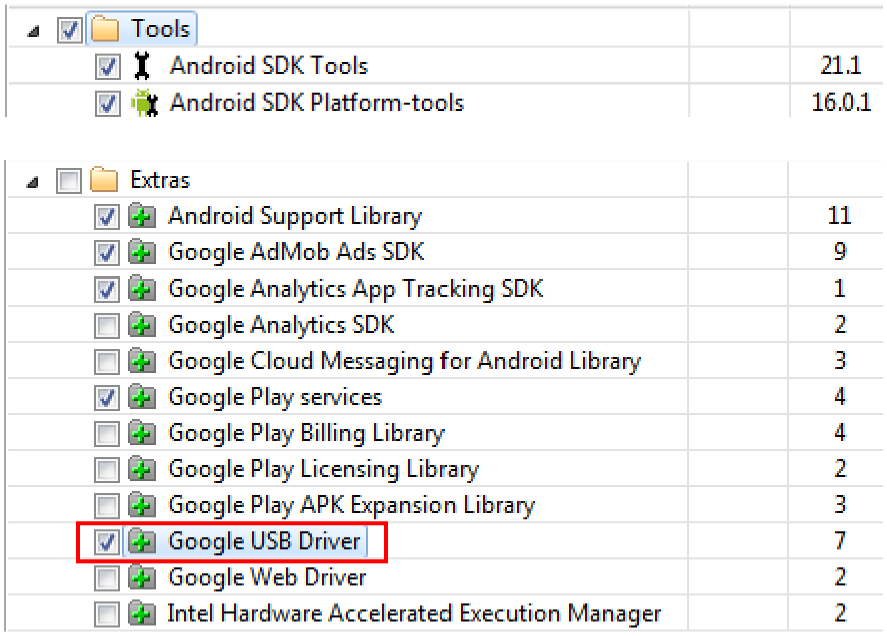
Name	API	Rev.	Status
☐ Android SDK Platform-tools		16.0.2	Installed
☐ API 18			
☑ Android 4.2.2 (API 17)			
☑ *Documentation for Android SDK*	*17*	*2*	*Not installed*
☑ *SDK Platform*	*17*	*2*	*Not installed*
☑ *Samples for SDK*	*17*	*1*	*Not installed*
☑ *ARM EABI v7a System Image*	*17*	*2*	*Not installed*
☑ *Intel x86 Atom System Image*	*17*	*1*	*Not installed*
☑ *MIPS System Image*	*17*	*1*	*Not installed*
☑ *Google APIs*	*17*	*3*	*Not installed*
☑ *Sources for Android SDK*	*17*	*1*	*Not installed*
☐ Android 4.1.2 (API 16)			
☐ Android 4.0.3 (API 15)			

Show: ☑ Updates/New ☑ Installed ☐ Obsolete Select New or Updates [Install 16 packages...]

Sort by: ● API level ○ Repository Deselect All [Delete 8 packages...]

Done loading packages.

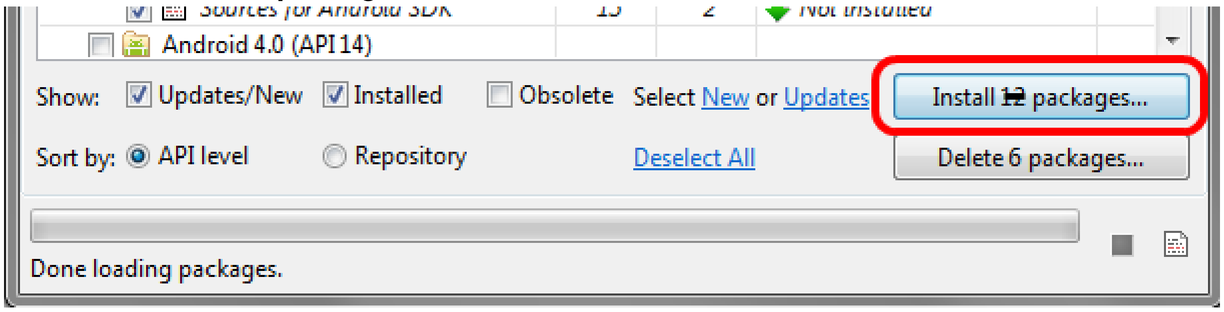
2. Except Tools – **Android SDK Tools** and **Android SDK Platform-tools** – also select Android API packages you need. In this case, we need use **Android 4.2 (API 17)**. Moreover, for your future development more convenient, you have better select **Google USB Driver**.



☑ Tools	
☑ Android SDK Tools	21.1
☑ Android SDK Platform-tools	16.0.1

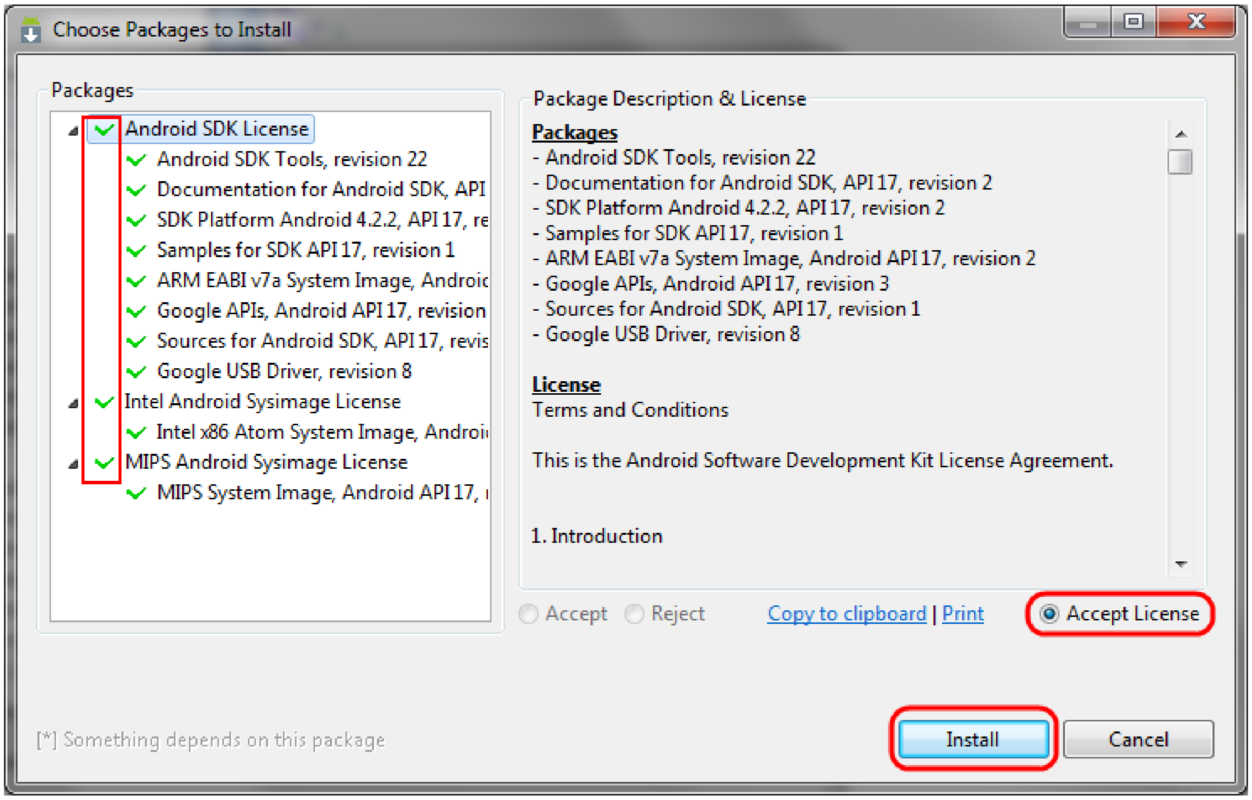
☐ Extras	
☑ Android Support Library	11
☑ Google AdMob Ads SDK	9
☑ Google Analytics App Tracking SDK	1
☐ Google Analytics SDK	2
☐ Google Cloud Messaging for Android Library	3
☑ Google Play services	4
☐ Google Play Billing Library	4
☐ Google Play Licensing Library	2
☐ Google Play APK Expansion Library	3
☑ Google USB Driver	7
☐ Google Web Driver	2
☐ Intel Hardware Accelerated Execution Manager	2

3. Now click **Install packages…**



Done loading packages.

4. Choose **Accept License** for all packages, and then click **Install**.


Choose Packages to Install
Packages
Android SDK License
Android SDK Tools, revision 22
Documentation for Android SDK, API
SDK Platform Android 4.2.2, API 17, re
Samples for SDK API 17, revision 1
ARM EABI v7a System Image, Androic
Google APIs, Android API 17, revision
Sources for Android SDK, API 17, revis
Google USB Driver, revision 8
Intel Android Sysimage License
Intel x86 Atom System Image, Androi
MIPS Android Sysimage License
MIPS System Image, Android API 17, i
Package Description & License

Packages
- Android SDK Tools, revision 22
- Documentation for Android SDK, API 17, revision 2
- SDK Platform Android 4.2.2, API 17, revision 2
- Samples for SDK API 17, revision 1
- ARM EABI v7a System Image, Android API 17, revision 2
- Google APIs, Android API 17, revision 3
- Sources for Android SDK, API 17, revision 1
- Google USB Driver, revision 8

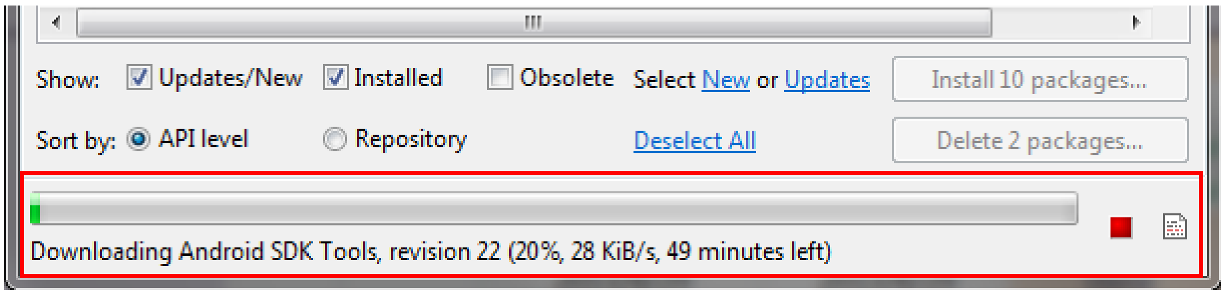
License
Terms and Conditions

This is the Android Software Development Kit License Agreement.

1. Introduction

Accept Reject Copy to clipboard | Print [✓ Accept License]
[*] Something depends on this package
Install
Cancel

5. Now start downloading SDK and installing.

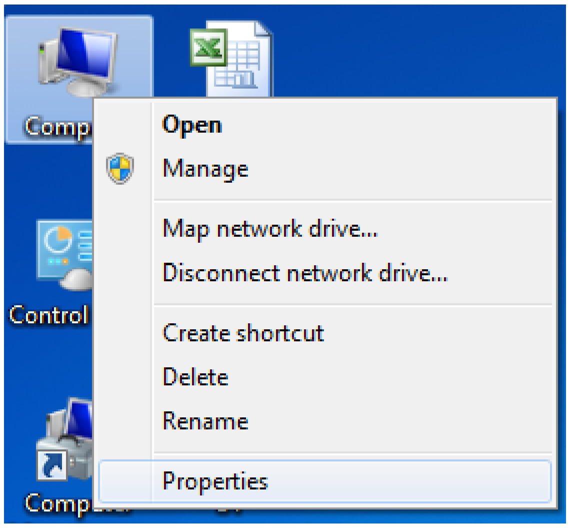

Show: Updates/New Installed Obsolete Select New or Updates
Install 10 packages...
Sort by: API level Repository
Deselect All
Delete 2 packages...

Downloading Android SDK Tools, revision 22 (20%, 28 KiB/s, 49 minutes left)

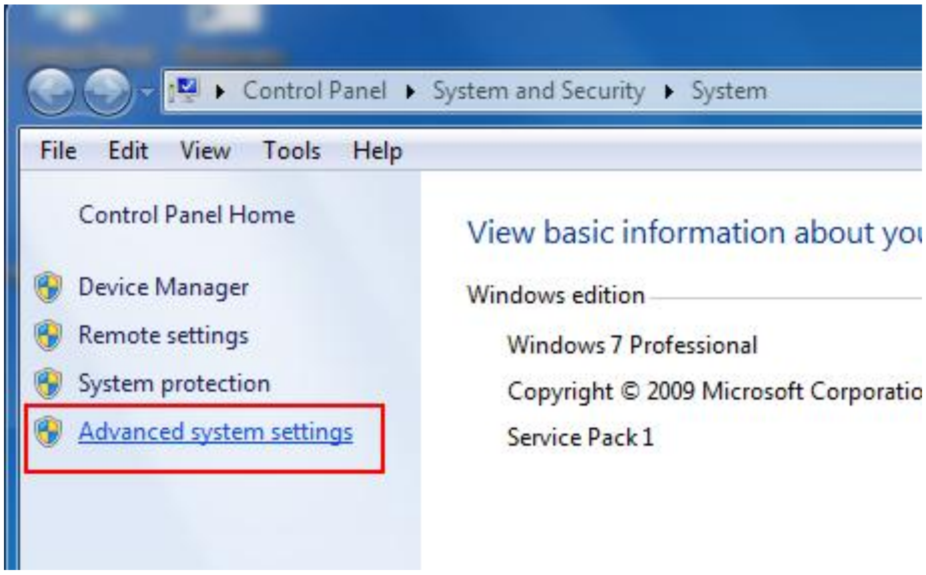
2.5 Setting environment variables

1. Add the path of SDK into the host PC can make you access the Android SDK tools easier.

1.1. Right-click **Computer**, select **Properties**.



1.2. In Windows 7, click **Advanced system settings** at left.



View basic information about yo

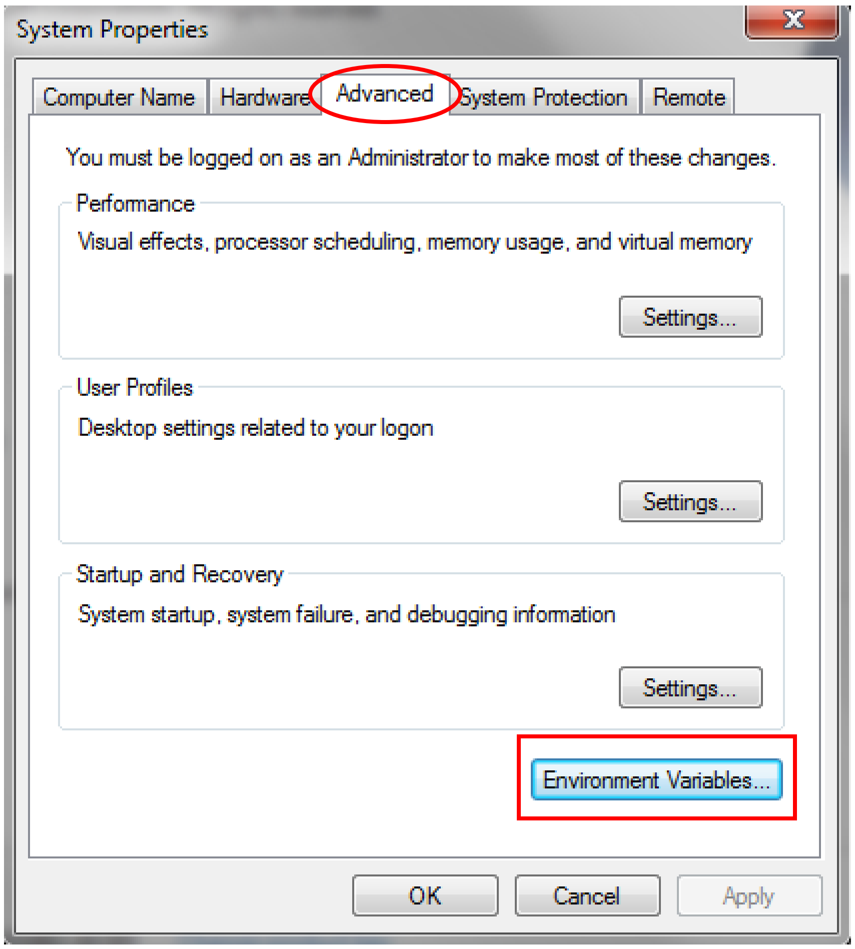
Windows edition

Windows 7 Professional

Copyright © 2009 Microsoft Corporatio

Service Pack 1

1.3. Select **Advanced** and click **Environment Variables…**



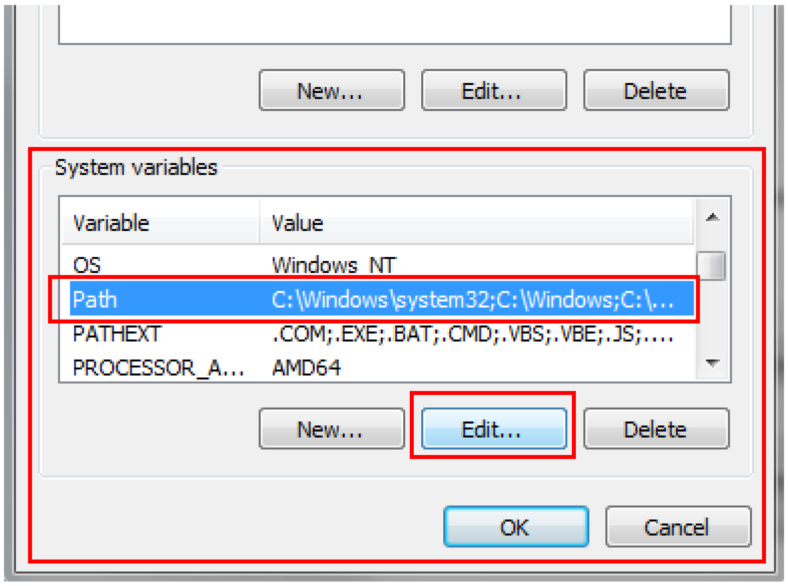
You must be logged on as an Administrator to make most of these changes.

Visual effects, processor scheduling, memory usage, and virtual memory

Desktop settings related to your logon

System startup, system failure, and debugging information

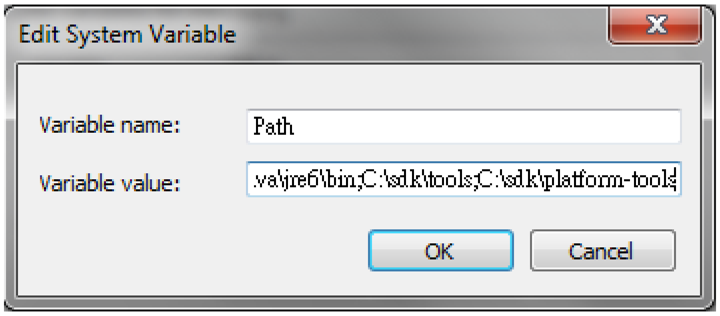
1.4. In System variables box, select **Path** and click **Edit…**



Variable	Value
OS	Windows_NT
Path	C:\Windows\system32;C:\Windows;C:\...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....
PROCESSOR_A...	AMD64

1.5. In Variable value, add the folder path of Android SDK you installt. For example, this case, `C:\sdk`:

```
;<sdk>\tools;<sdk>\platform-tools
```



Variable name: Path

Variable value: va\jre6\bin;C:\sdk\tools;C:\sdk\platform-tools

2.6 Install ADB

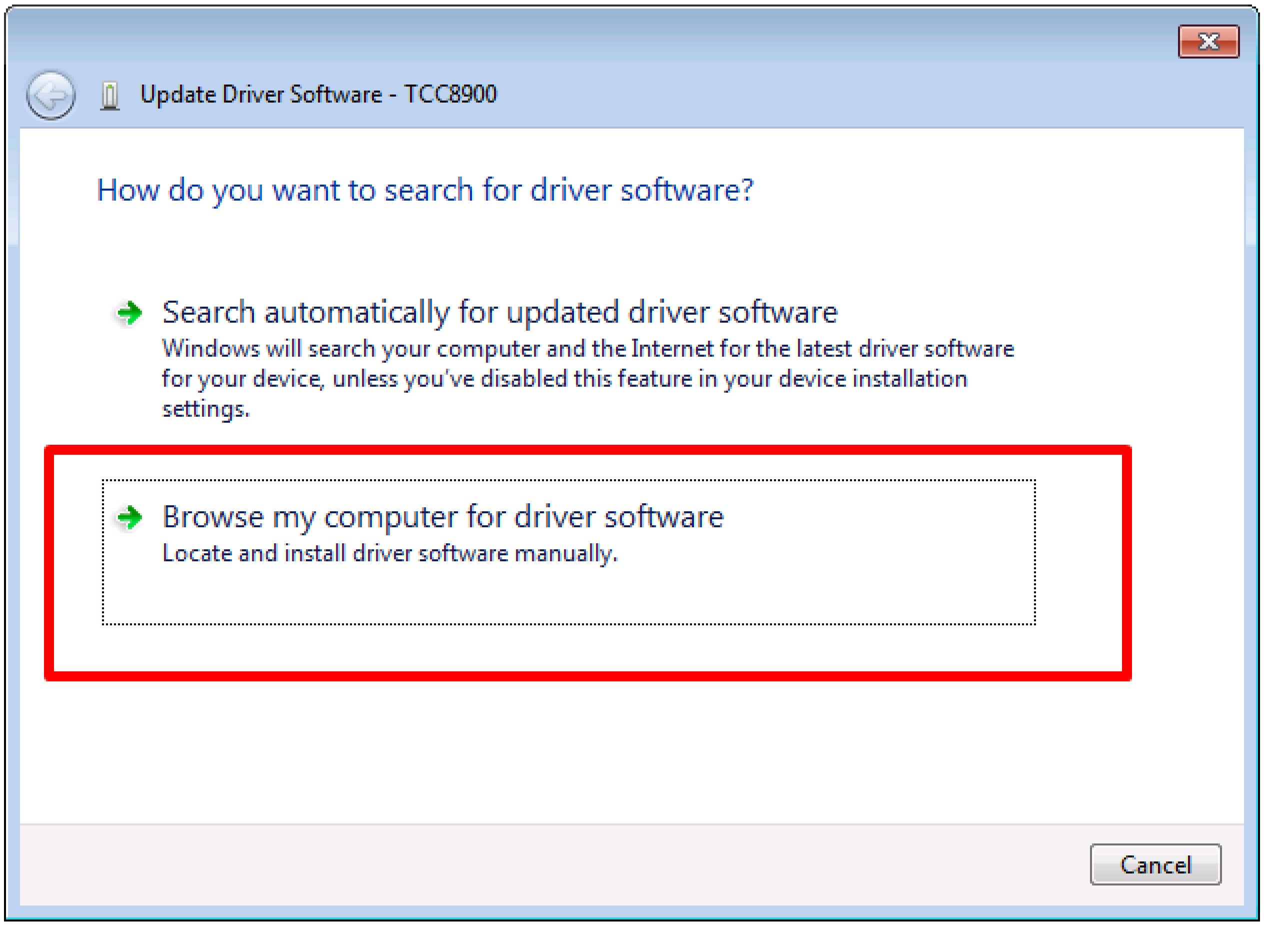
Android Debug Bridge (adb) is a tool that lets you can communicate between device and PC. More information in:

http://developer.android.com/tools/help/adb.html

For the first time connection, the host PC would prompt you to install software for the Android ADB Interface.

1. Replace the USB driver in `<sdk>\ extras\google\usb_driver` by the new driver we offer.
2. Connect the device to your PC.
3. Search the device and install.

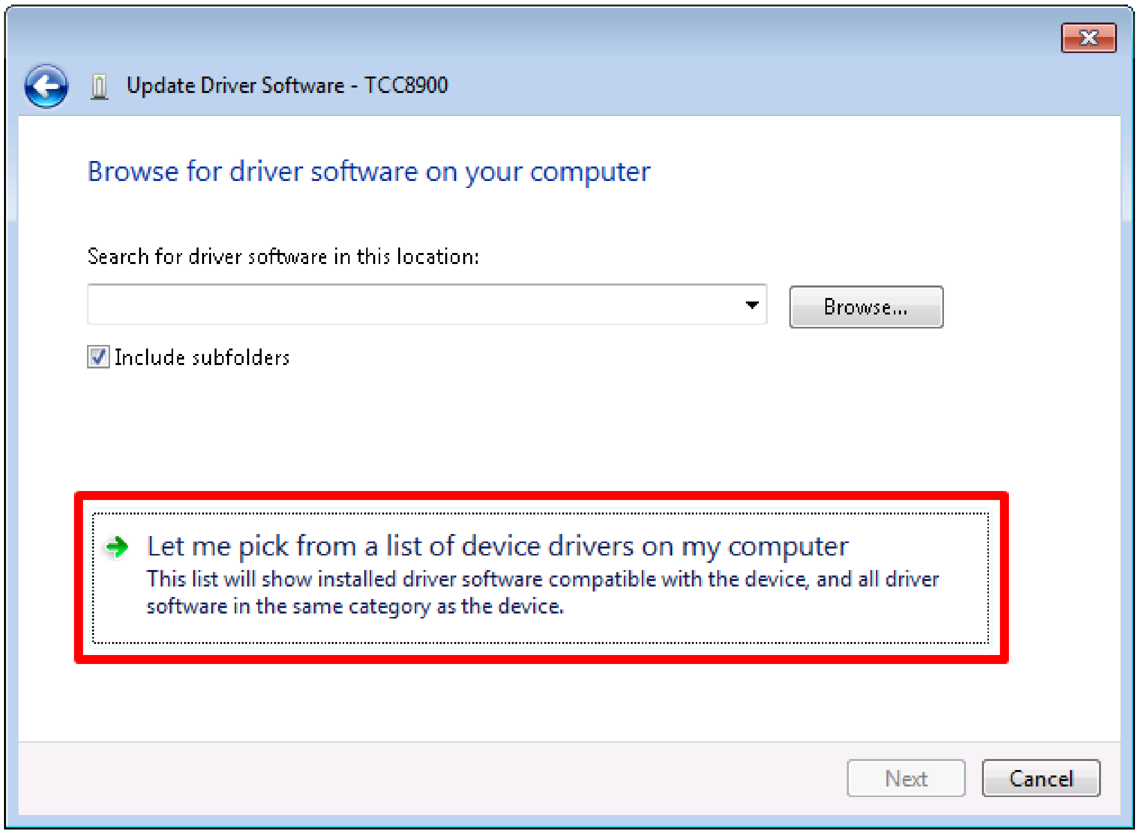
☐ In Windows 7:


Update Driver Software - TCC8900

How do you want to search for driver software?

Search automatically for updated driver software
Windows will search your computer and the Internet for the latest driver software for your device, unless you've disabled this feature in your device installation settings.

Browse my computer for driver software
Locate and install driver software manually.


Update Driver Software - TCC8900

Browse for driver software on your computer

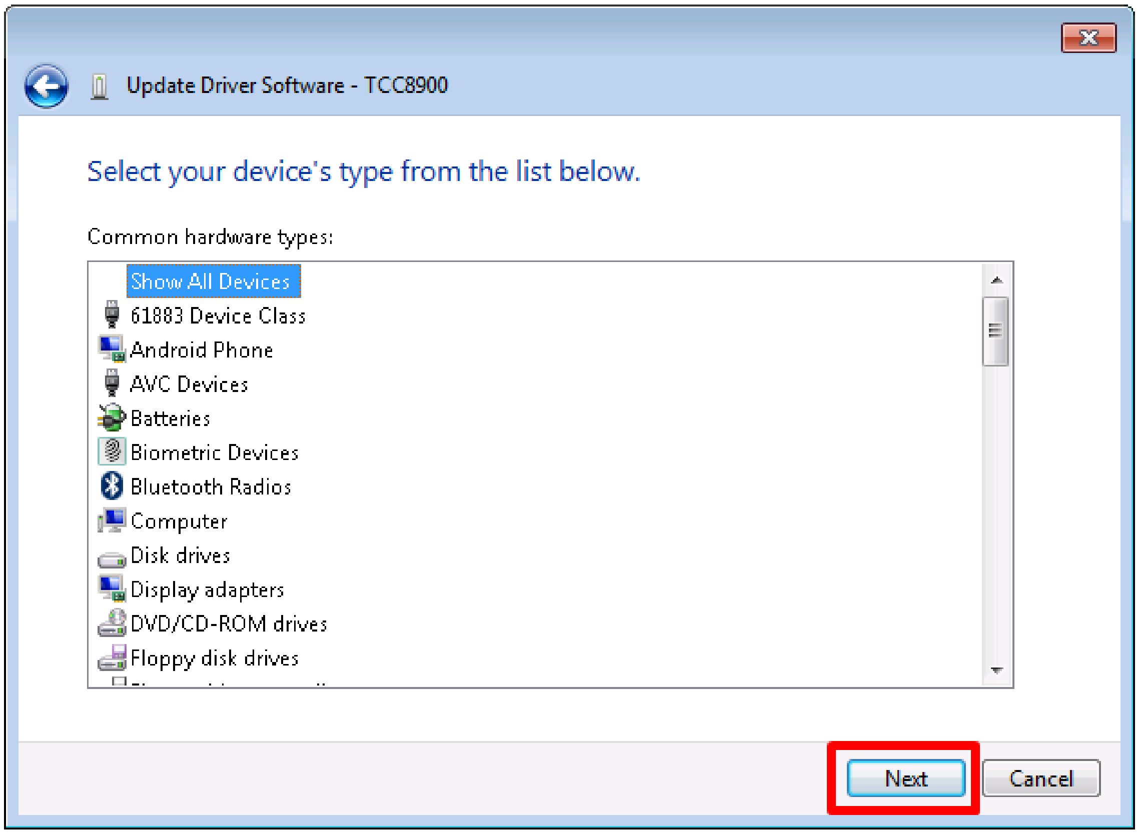
Search for driver software in this location:

Browse...

☑ Include subfolders

Let me pick from a list of device drivers on my computer
This list will show installed driver software compatible with the device, and all driver software in the same category as the device.

Next Cancel


Update Driver Software - TCC8900

Select your device's type from the list below.

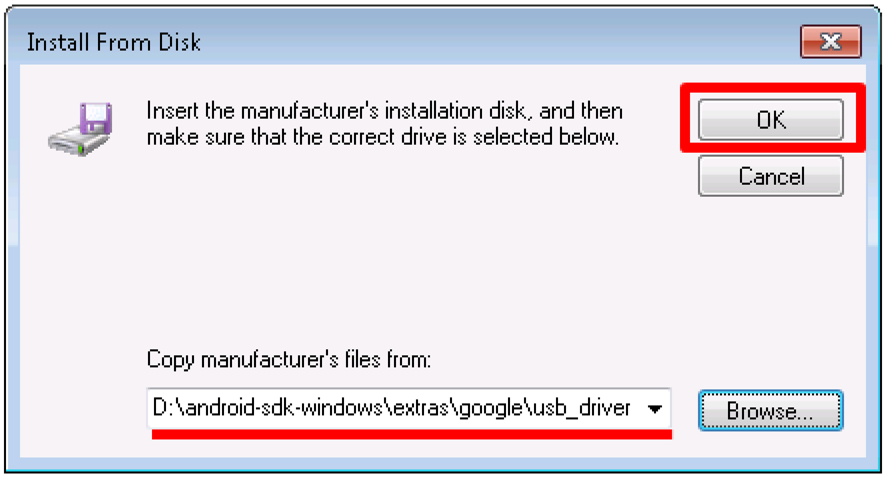
Common hardware types:

Show All Devices
61883 Device Class
Android Phone
AVC Devices
Batteries
Biometric Devices
Bluetooth Radios
Computer
Disk drives
Display adapters
DVD/CD-ROM drives
Floppy disk drives

Next Cancel

Input the path of Android SDK USB driver:

`<sdk>\ extras\google\usb_driver`

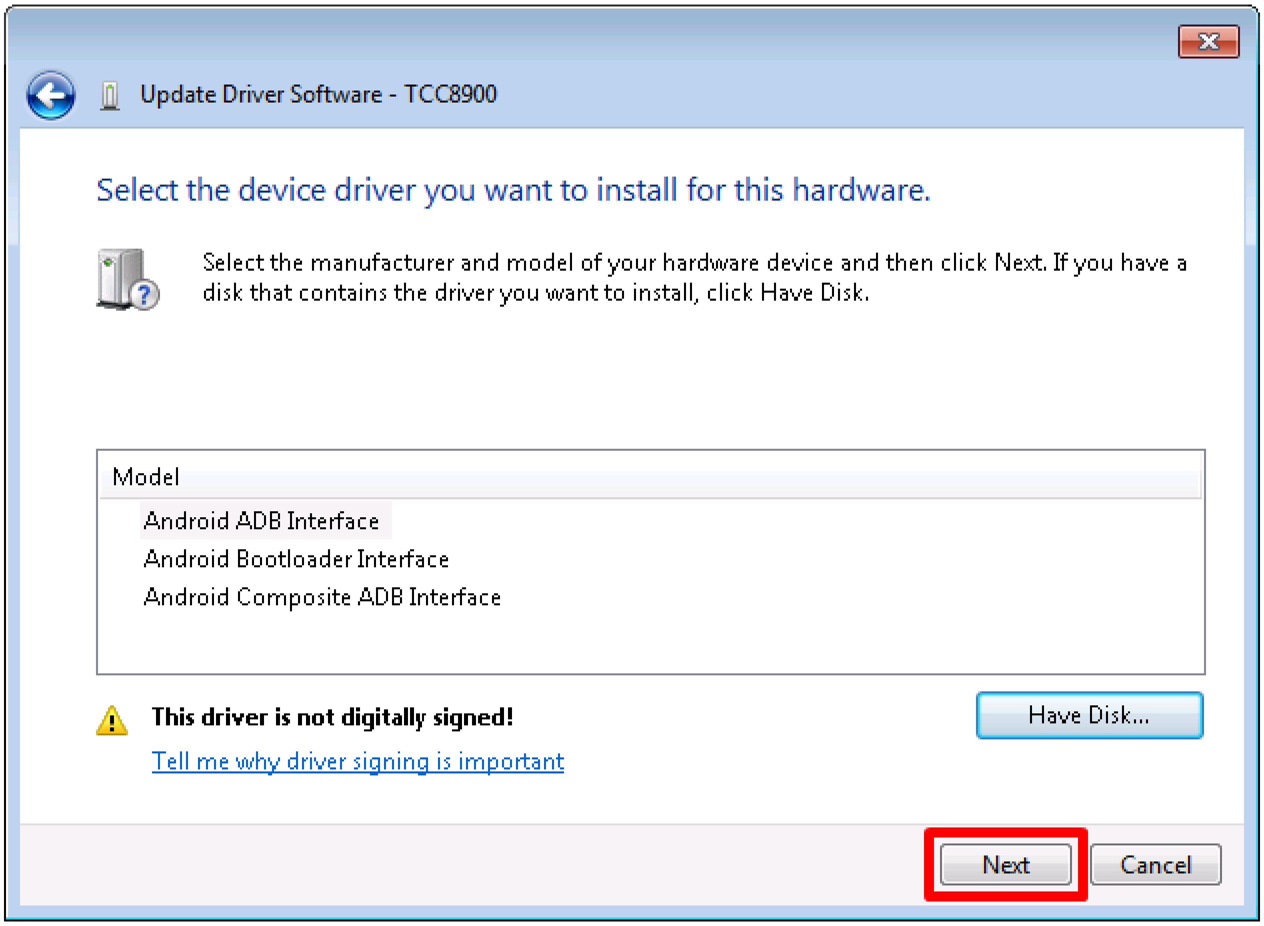


Install From Disk

Insert the manufacturer's installation disk, and then make sure that the correct drive is selected below.

Copy manufacturer's files from:

D:\android-sdk-windows\extras\google\usb_driver



Update Driver Software - TCC8900

Select the device driver you want to install for this hardware.

Select the manufacturer and model of your hardware device and then click Next. If you have a disk that contains the driver you want to install, click Have Disk.

Model

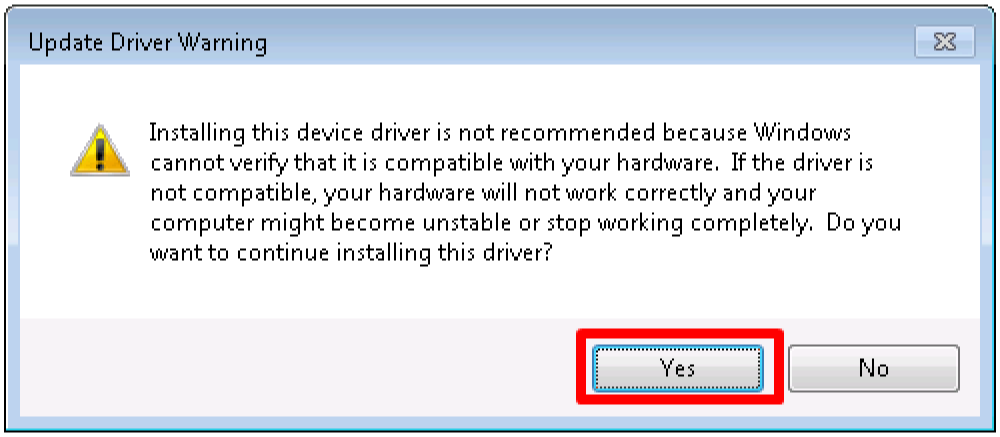
Android ADB Interface

Android Bootloader Interface

Android Composite ADB Interface

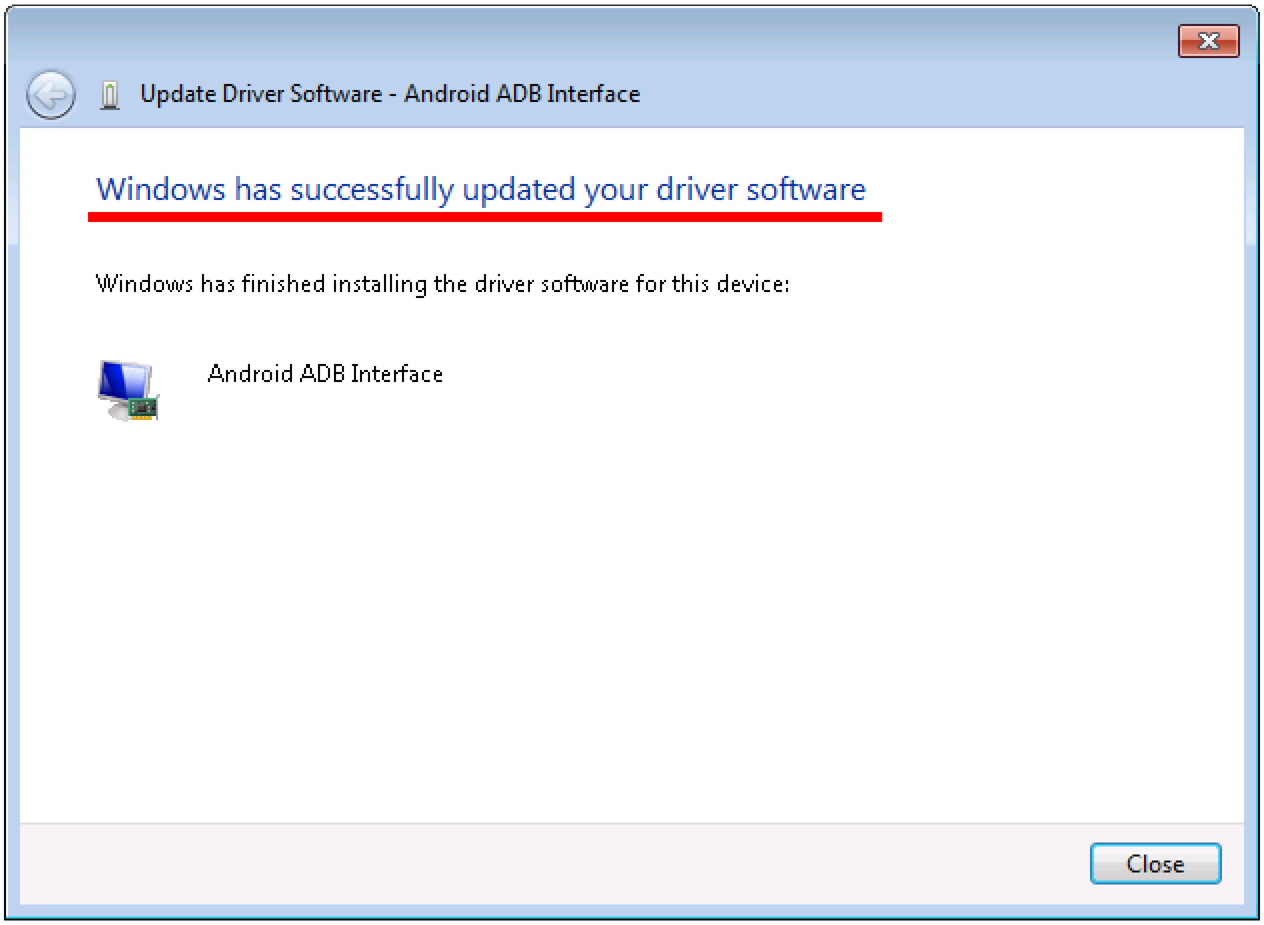
This driver is not digitally signed!

Tell me why driver signing is important



Update Driver Warning

Installing this device driver is not recommended because Windows cannot verify that it is compatible with your hardware. If the driver is not compatible, your hardware will not work correctly and your computer might become unstable or stop working completely. Do you want to continue installing this driver?



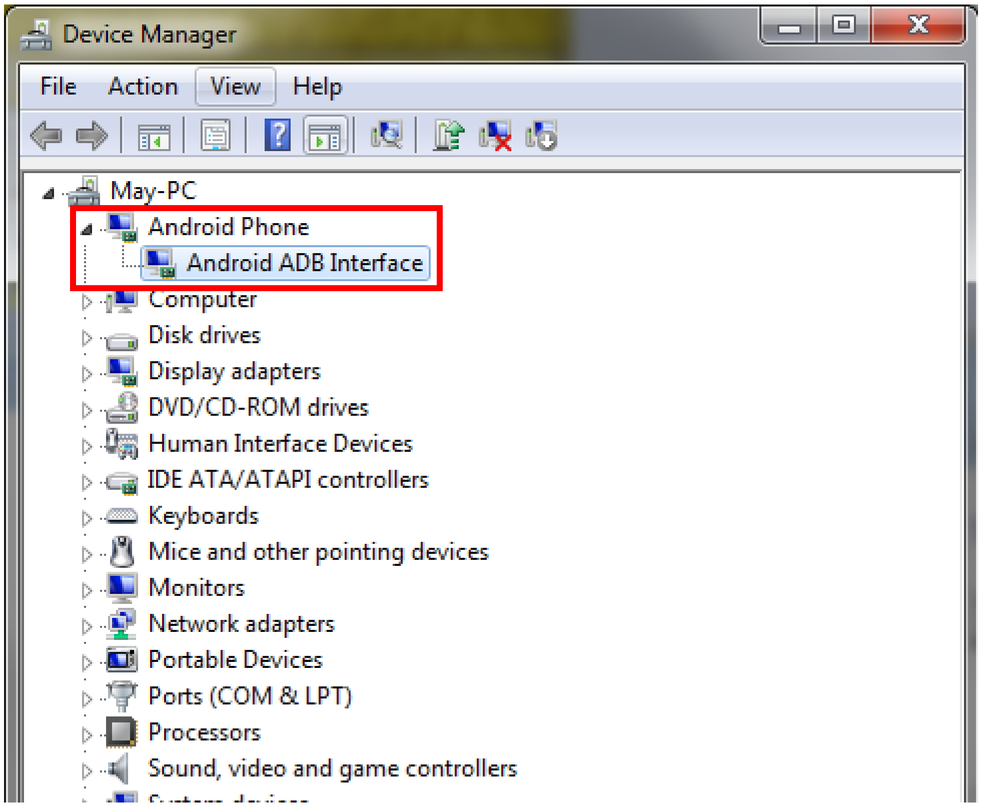
Update Driver Software - Android ADB Interface

Windows has successfully updated your driver software

Windows has finished installing the driver software for this device:

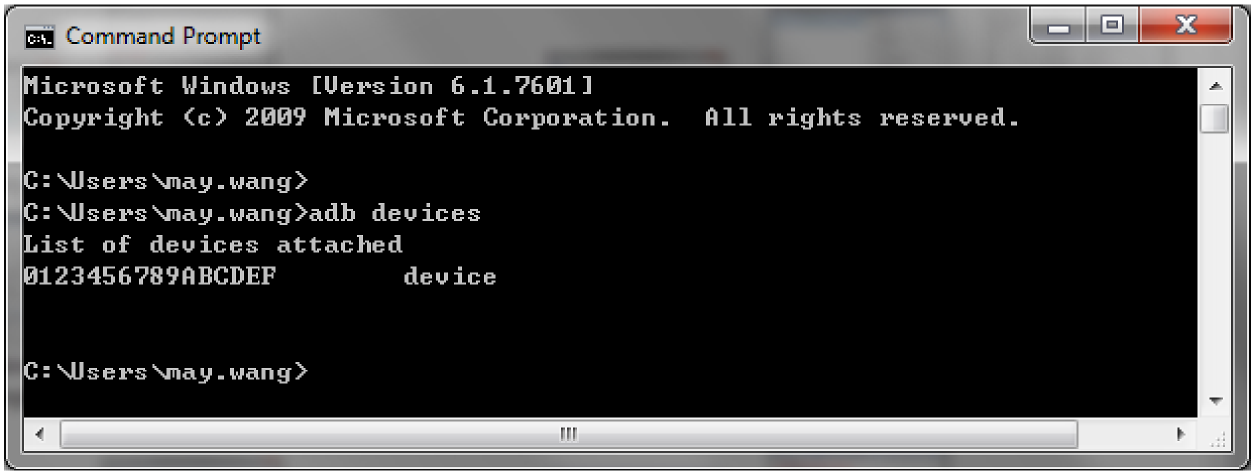
Android ADB Interface

4. Then you will see there is an **Android ADB Interface** in Device Manager.



5. Run Command Prompt and type below command to see the device is attached.

```
> adb devices
```



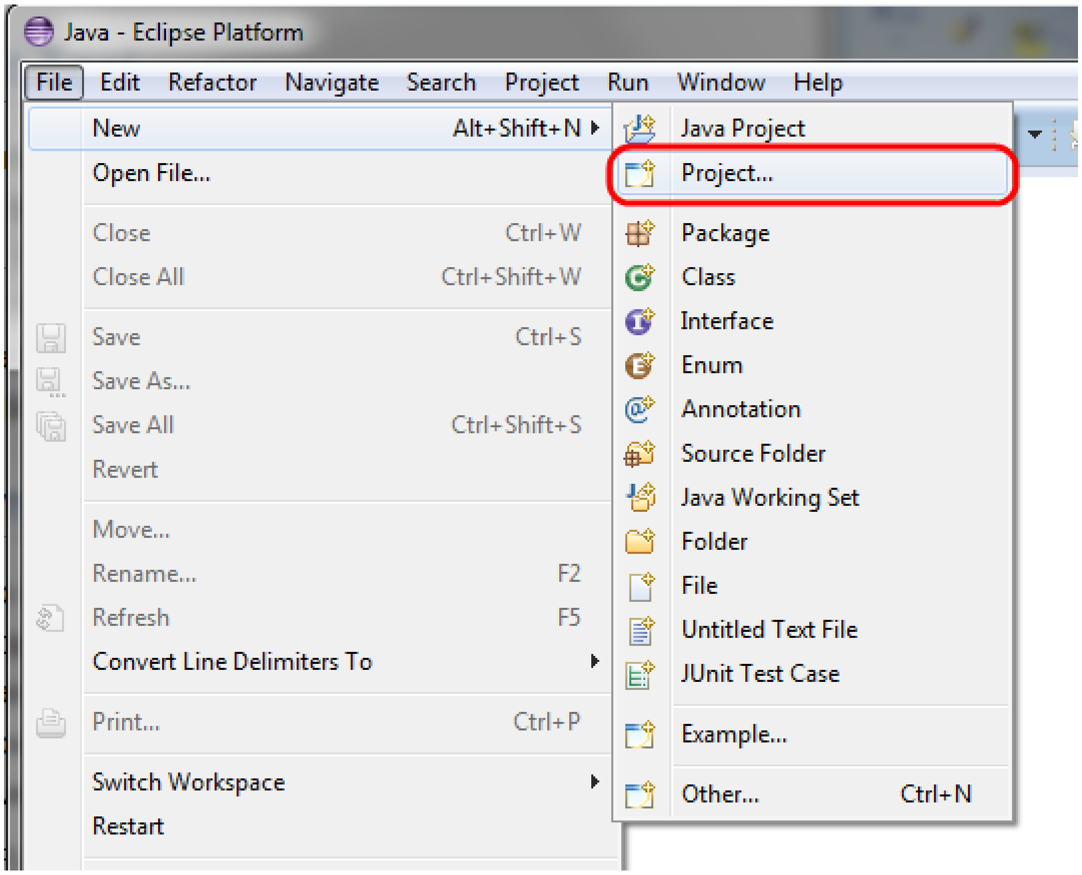
```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\may.wang>
C:\Users\may.wang>adb devices
List of devices attached
0123456789ABCDEF        device


C:\Users\may.wang>
```

Chapter 3: Android UI Design: Graphical Layout Tool

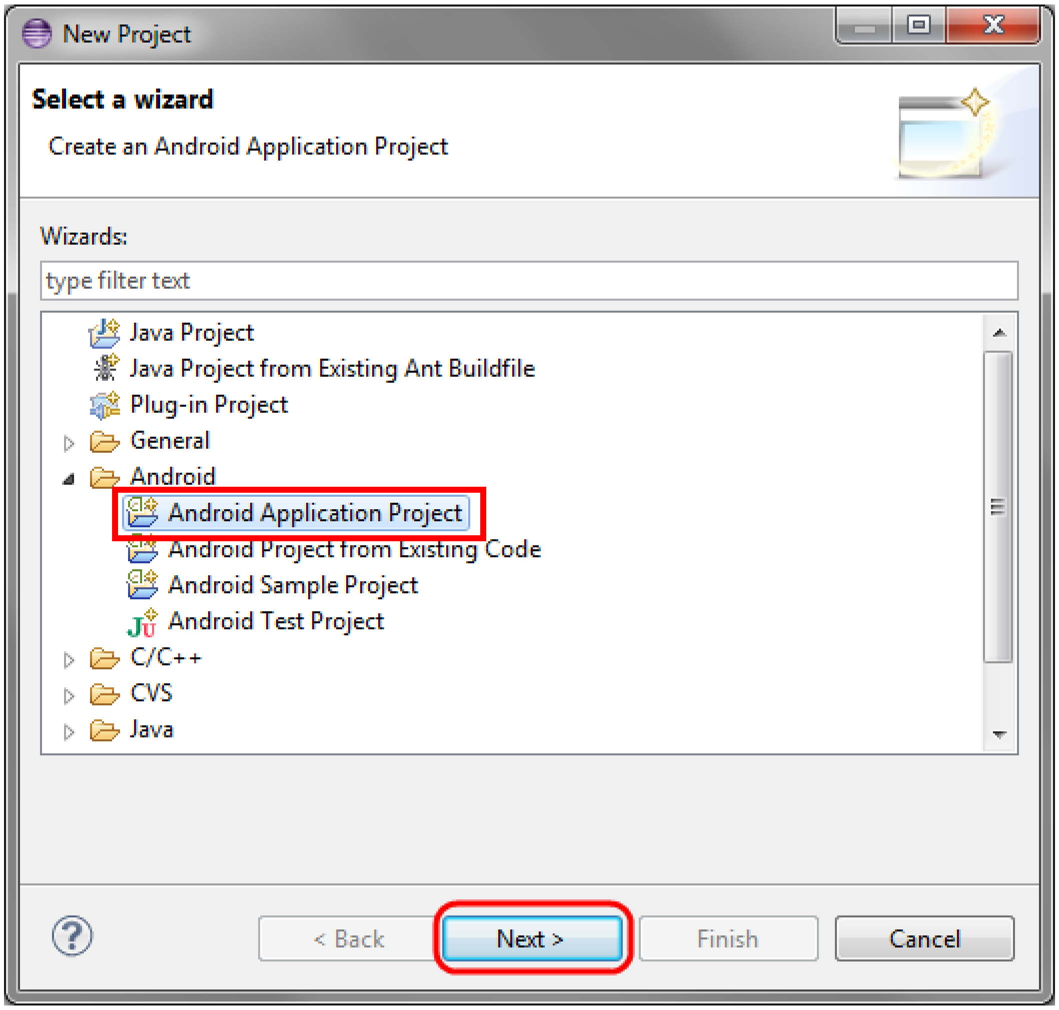
This chapter guides you how to design a simple APP via Eclipse IDE.

3.1 New a Project

1. The first step, you need to create a new Android Project. Select **File → New → Project…**



2. Select **Android Application Project** and click **Next >**.



New Project

Select a wizard

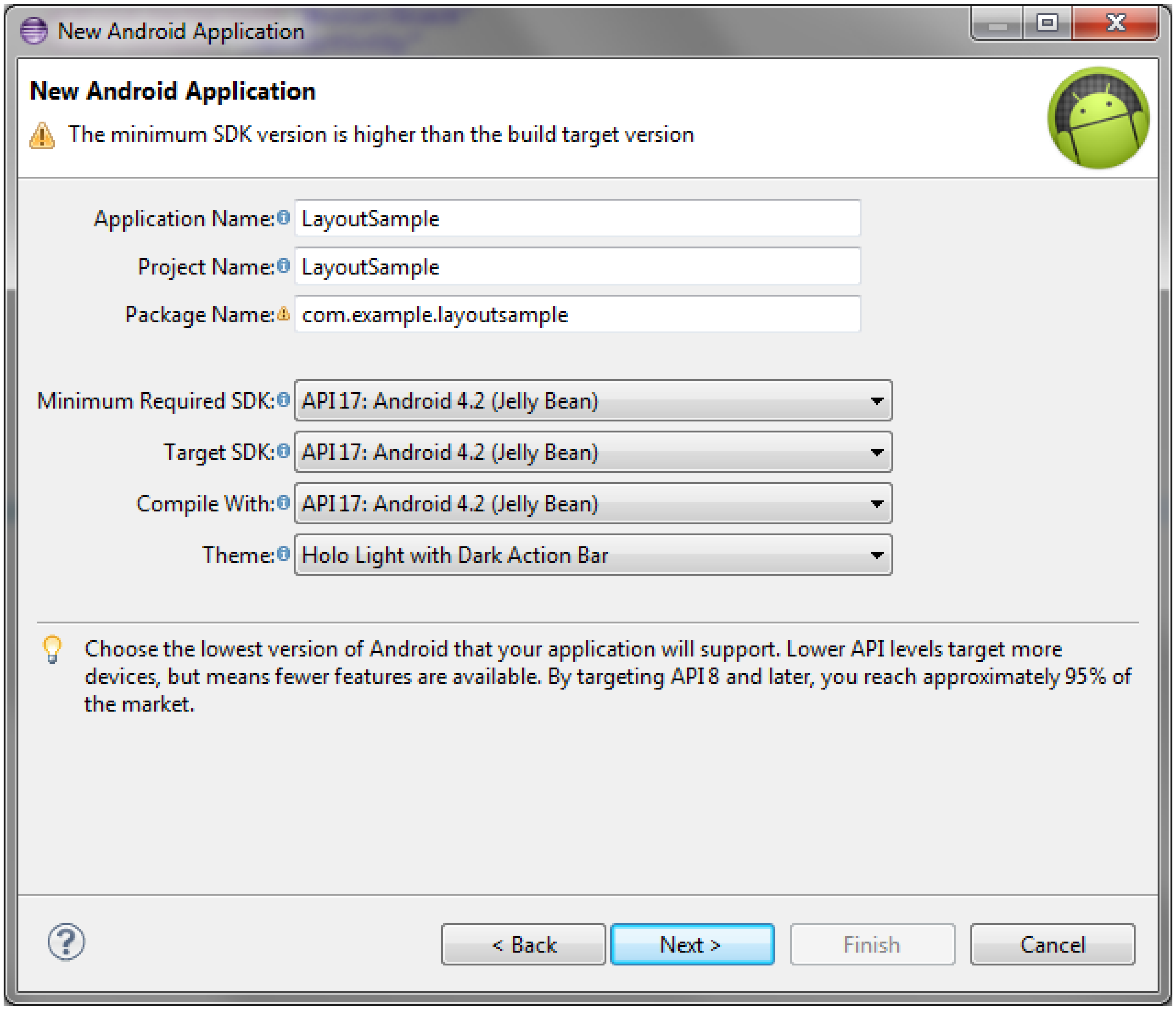
Create an Android Application Project

Wizards:

type filter text

- Java Project
- Java Project from Existing Ant Buildfile
- Plug-in Project
- General
- Android
  - Android Application Project
  - Android Project from Existing Code
  - Android Sample Project
  - Android Test Project
- C/C++
- CVS
- Java

3. Input the application and project name.
Minimum Required SDK could be lower so that you can run your app in more devices. Select appropriate Target SDK version for you. For example, here, we use API 17 (Android 4.2). Then **Next >**. (*Lower API levels target more devices, but means fewer features are available.*)



New Android Application

The minimum SDK version is higher than the build target version

Application Name: LayoutSample

Project Name: LayoutSample

Package Name: com.example.layoutsample

Minimum Required SDK: API 17: Android 4.2 (Jelly Bean)

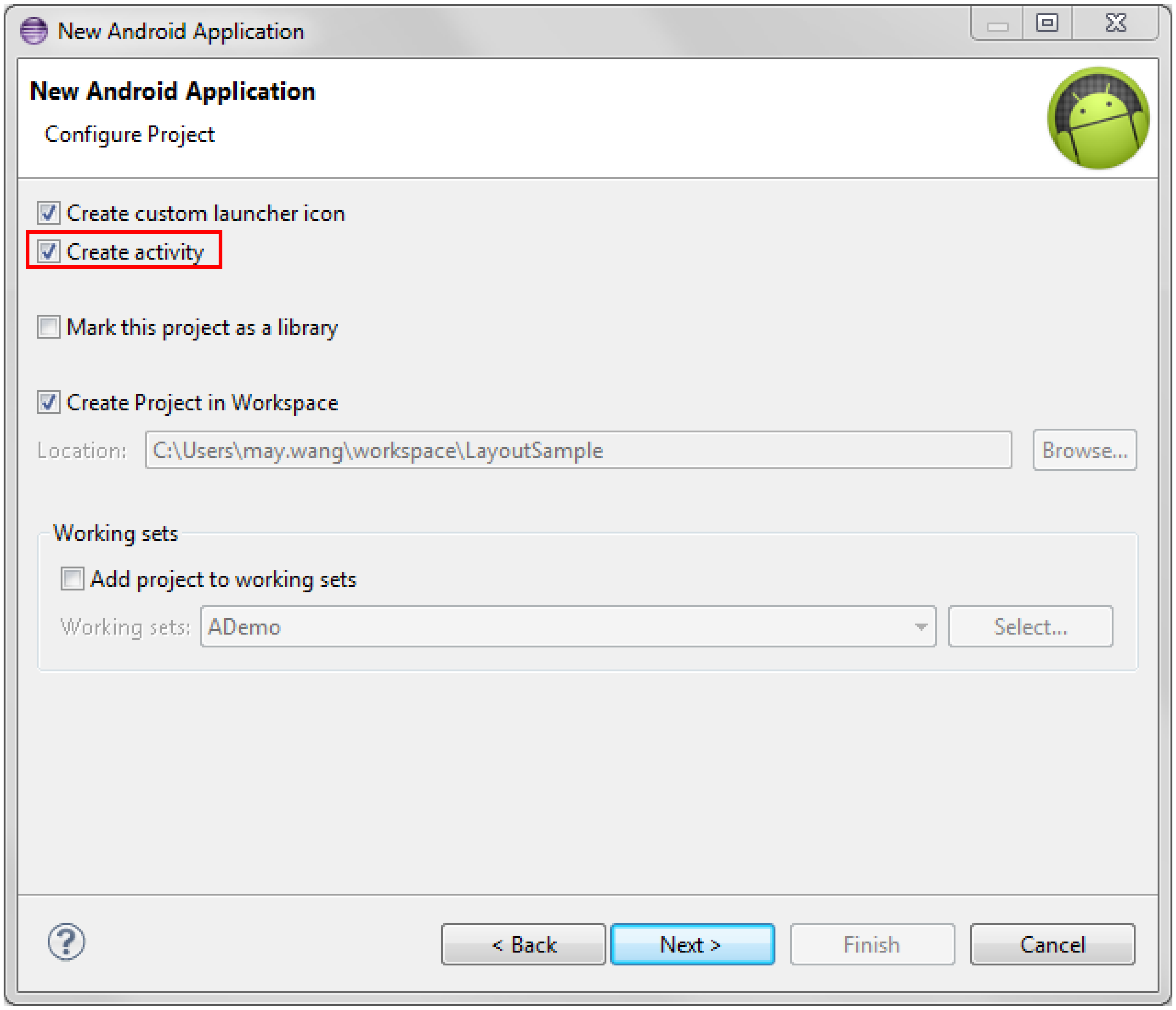
Target SDK: API 17: Android 4.2 (Jelly Bean)

Compile With: API 17: Android 4.2 (Jelly Bean)

Theme: Holo Light with Dark Action Bar

Choose the lowest version of Android that your application will support. Lower API levels target more devices, but means fewer features are available. By targeting API 8 and later, you reach approximately 95% of the market.

4. Configure your project. Check *Create custom launcher icon* could let you design your icon of App. Please check *Create activity* that could be easier to use if you are the first time.Then click **Next >**.


New Android Application

New Android Application

Configure Project

- [✓] Create custom launcher icon
- [✓] Create activity
- [ ] Mark this project as a library
- [✓] Create Project in Workspace

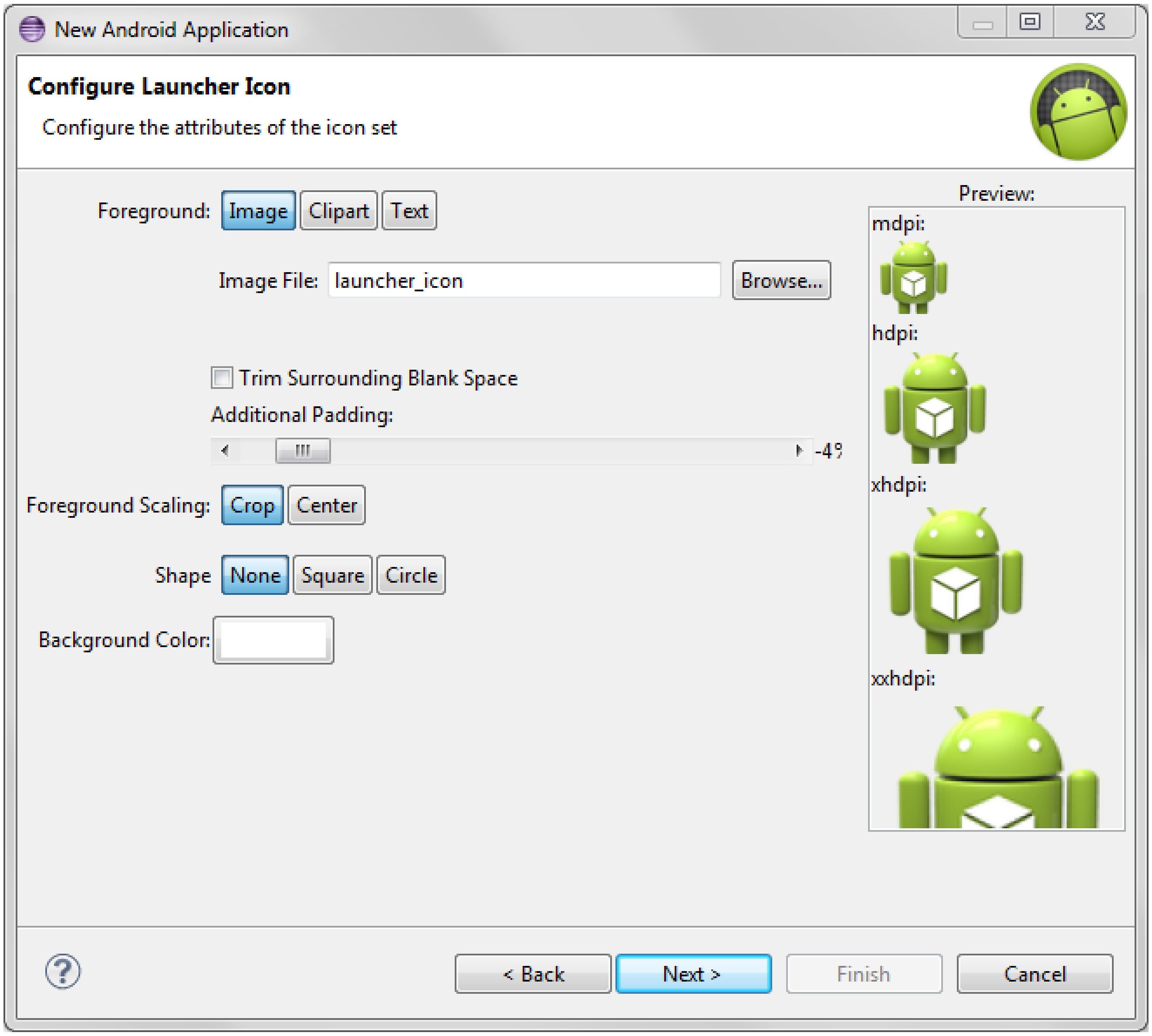
Location: C:\Users\may.wang\workspace\LayoutSample

Working sets

- [ ] Add project to working sets

Working sets: ADemo

5. Configure launcher icon if you check to create icon. Then **Next**.


New Android Application

Configure Launcher Icon

Configure the attributes of the icon set

Foreground: [✓ Image] Clipart Text

Image File: launcher_icon

Trim Surrounding Blank Space

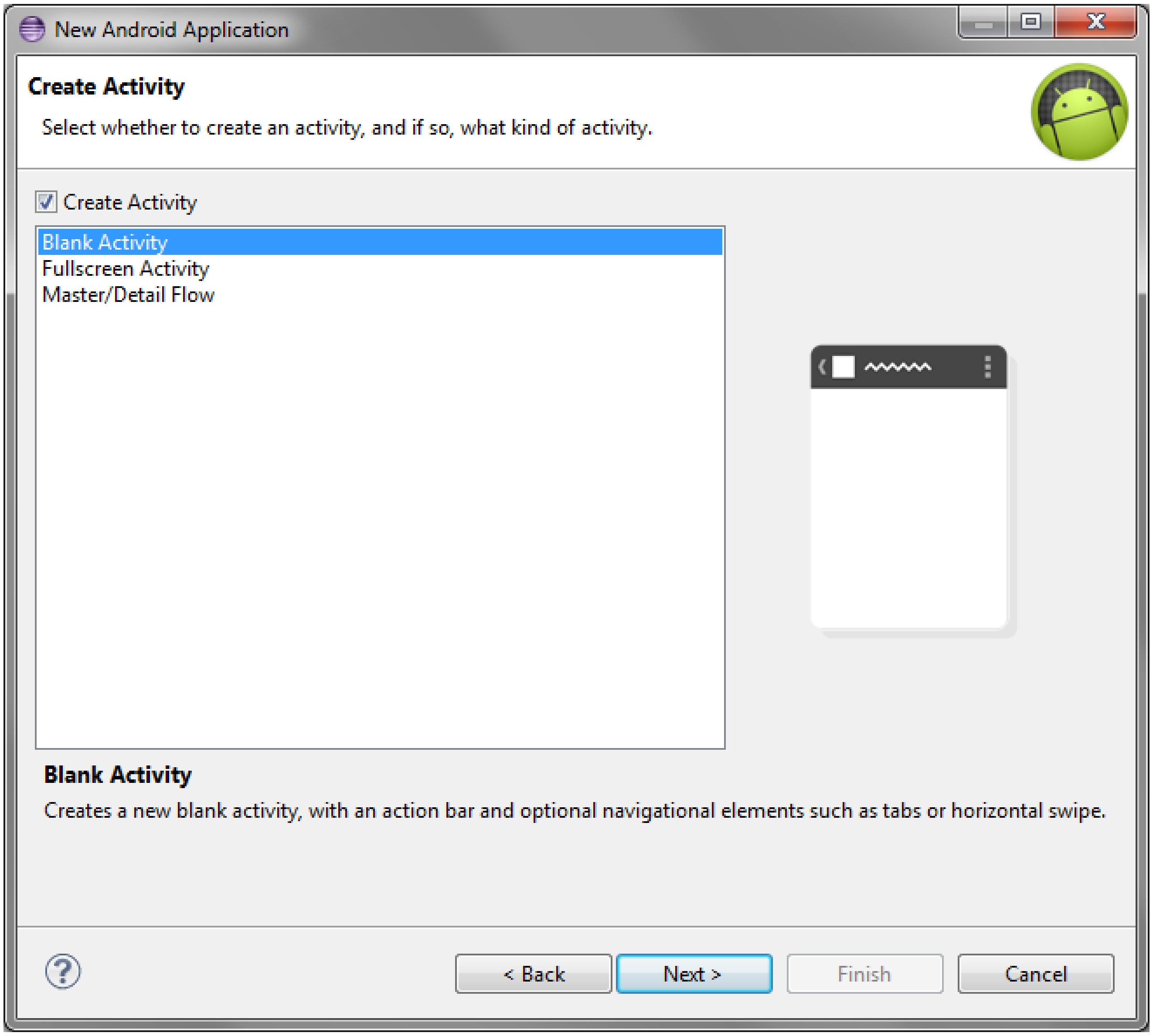
Additional Padding: -49

Foreground Scaling: [✓ Crop] Center

Shape [✓ None] Square Circle

Background Color:

6. Check *Create Activity* and select **Blank Actitvity**. Then **Next**.


New Android Application

Create Activity

Select whether to create an activity, and if so, what kind of activity.

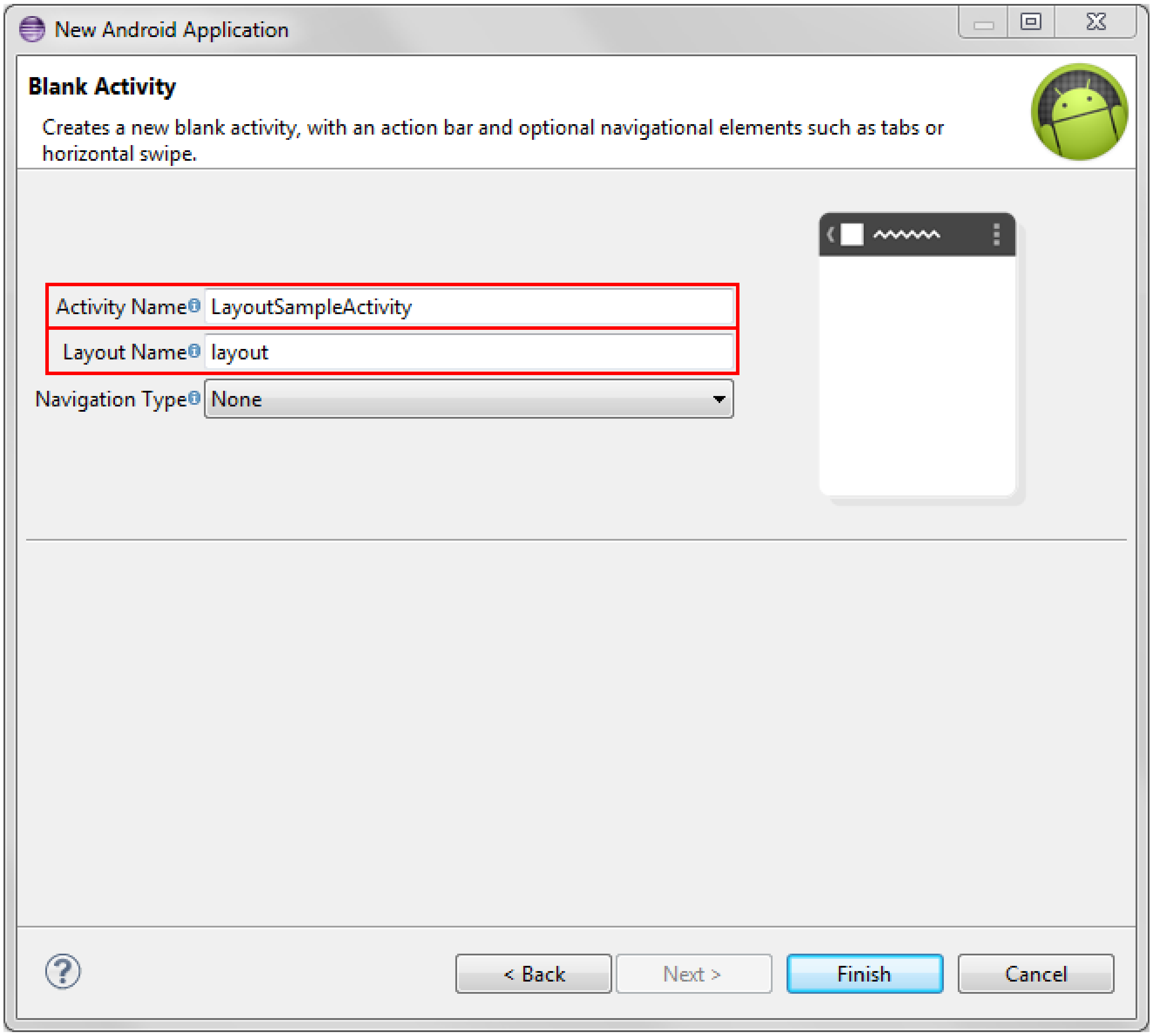
☑ Create Activity

- Blank Activity
- Fullscreen Activity
- Master/Detail Flow

Blank Activity

Creates a new blank activity, with an action bar and optional navigational elements such as tabs or horizontal swipe.

7. Name the activity and layout name. Finally, click **Finish**.


New Android Application

Blank Activity

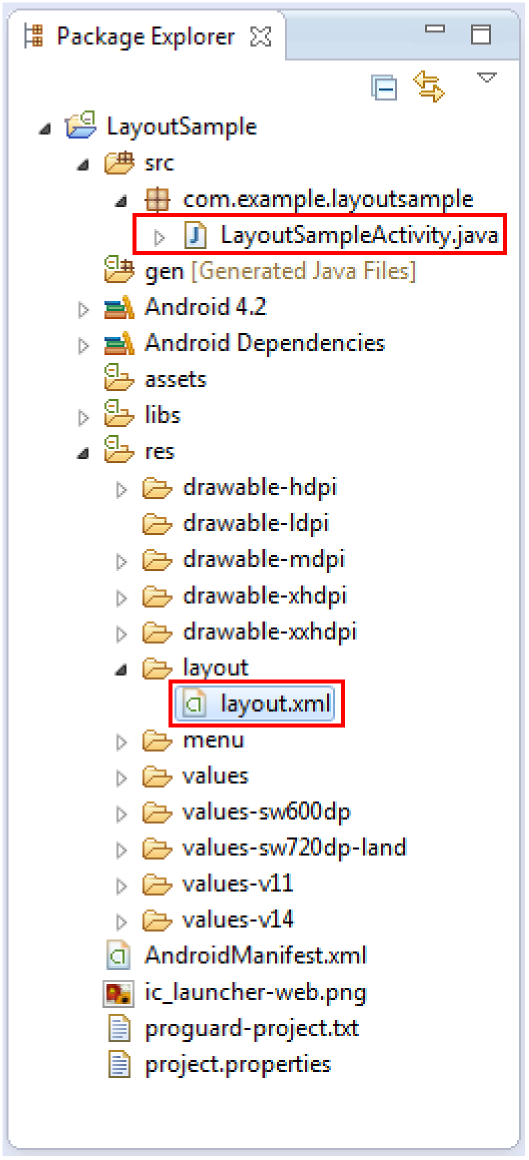
Creates a new blank activity, with an action bar and optional navigational elements such as tabs or horizontal swipe.

Activity Name: LayoutSampleActivity

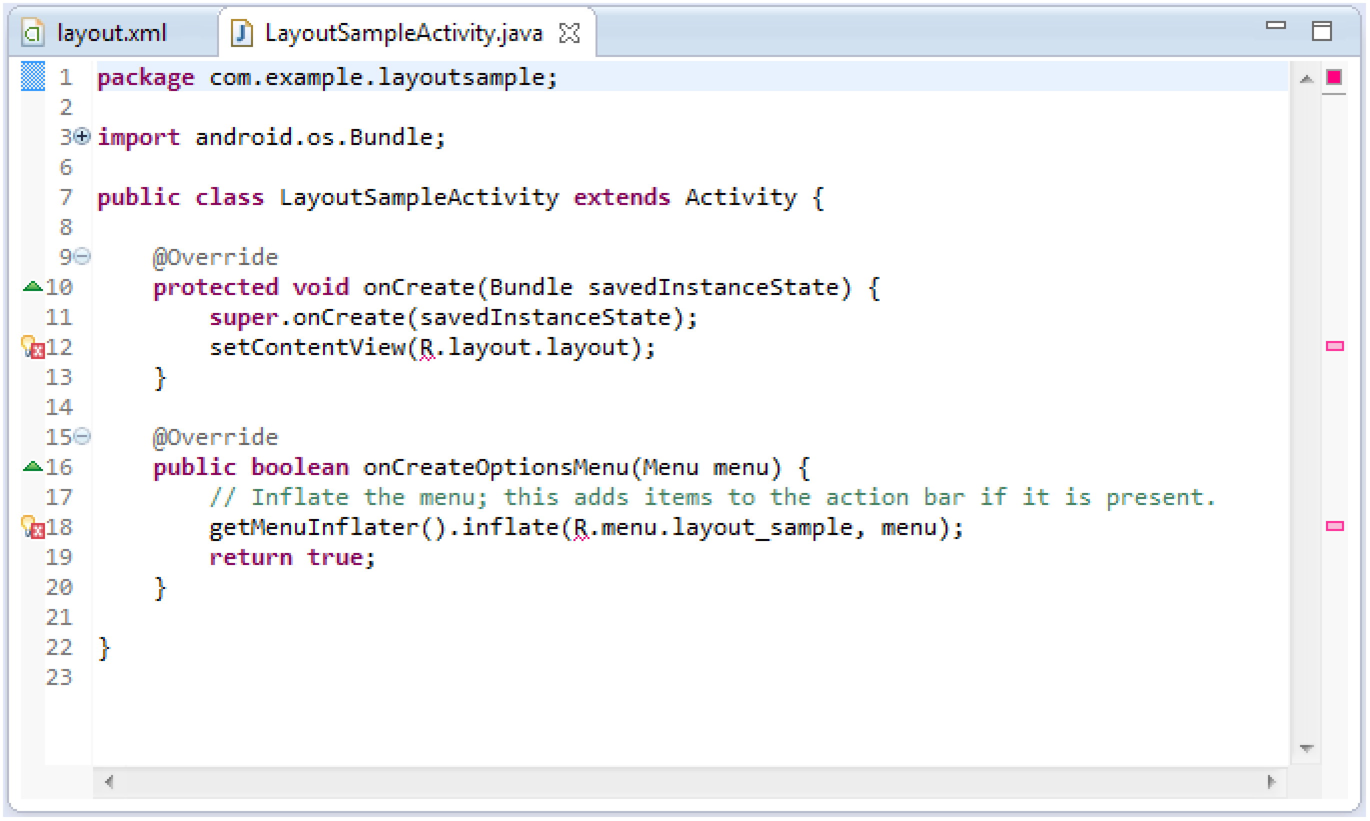
Layout Name: layout

Navigation Type: None

8. Then it would automatically generate a **Java file** and a **XML layout file**.

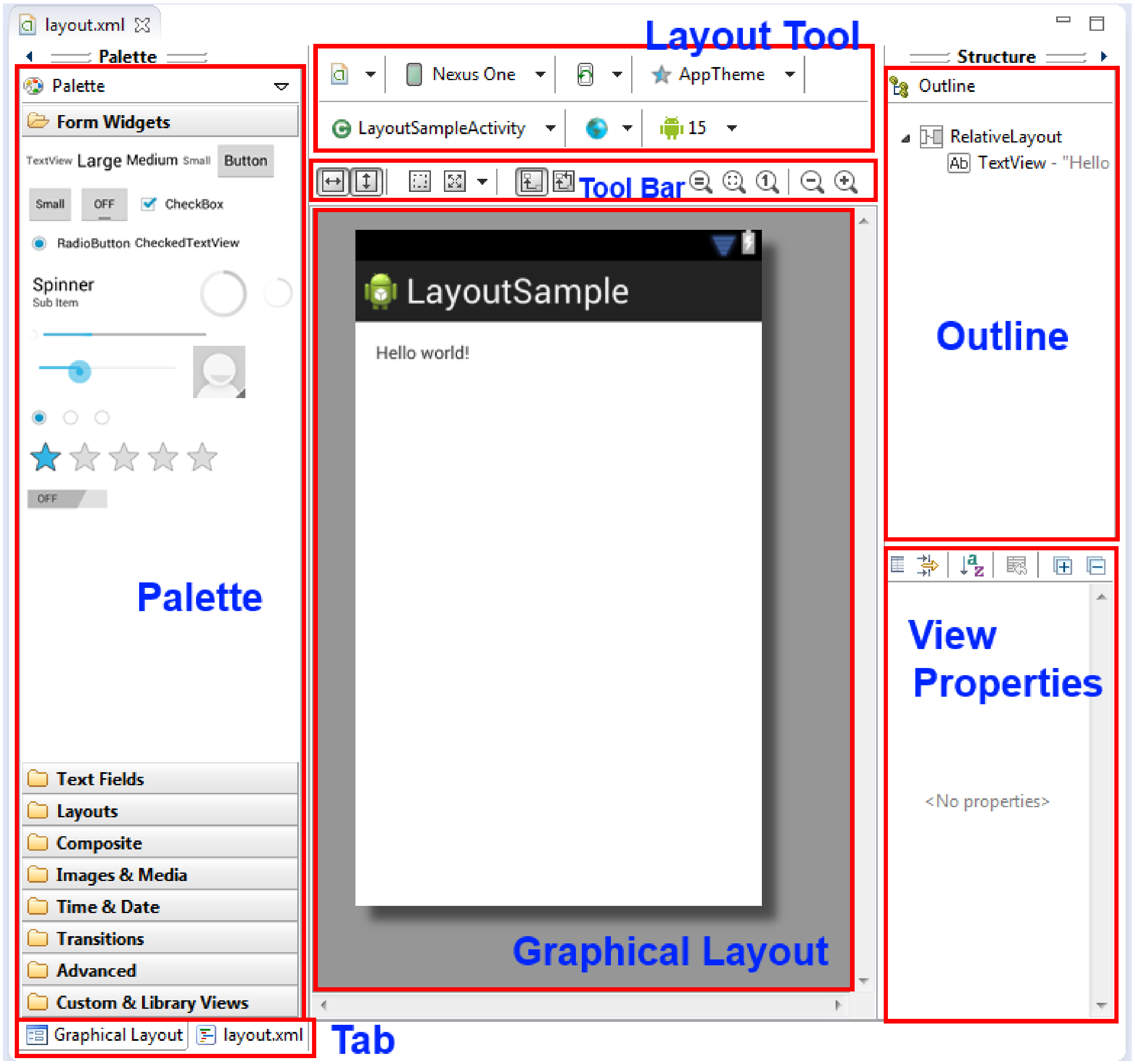


9. Java file is the main file you program in Android app development. It associates with XML layout file.



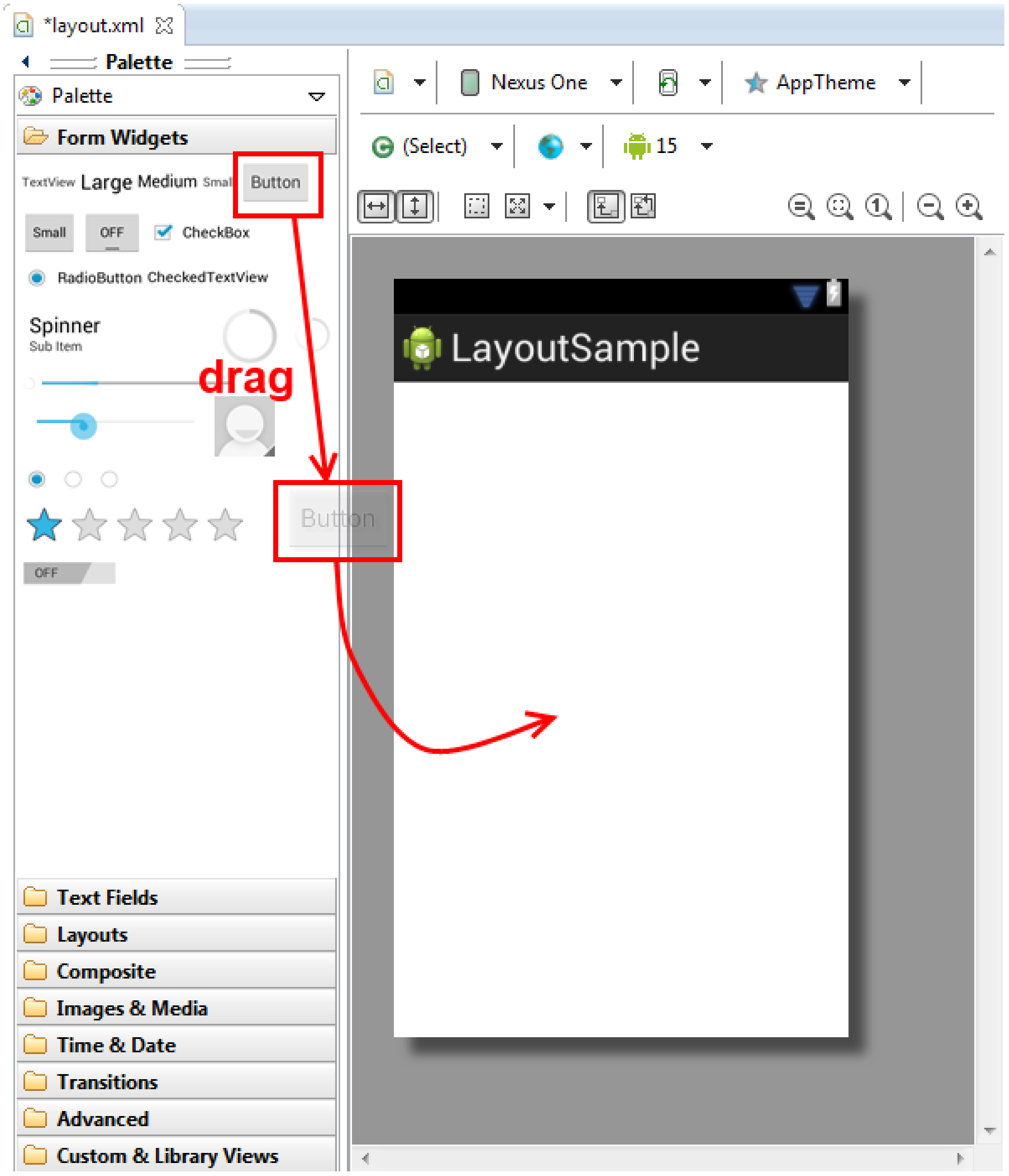
```
package com.example.layoutsample;

import android.os.Bundle;

public class LayoutSampleActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.layout);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.layout_sample, menu);
        return true;
    }

}
```

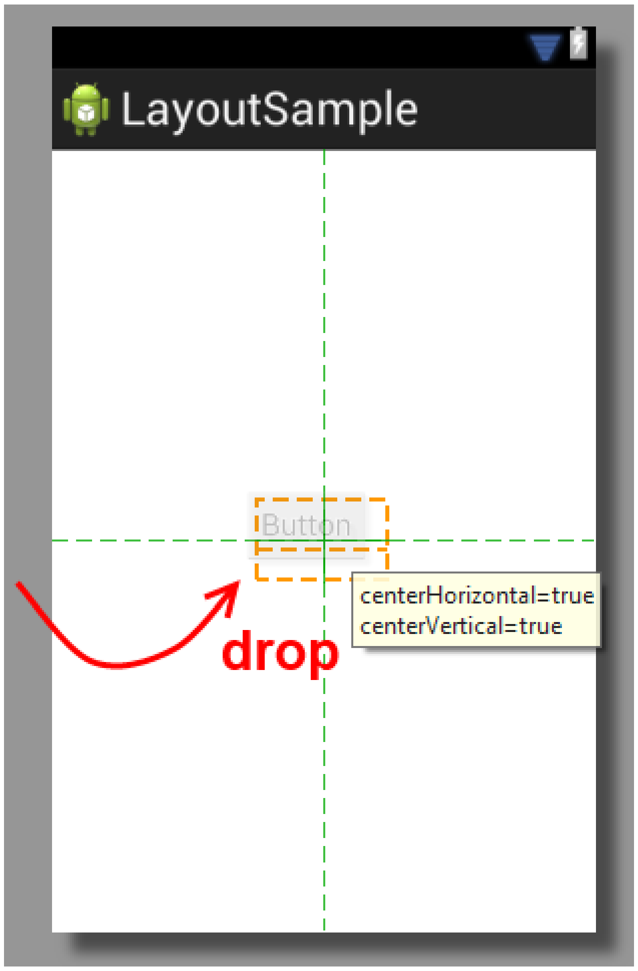
10. XML layout file is where toy could do UI design.

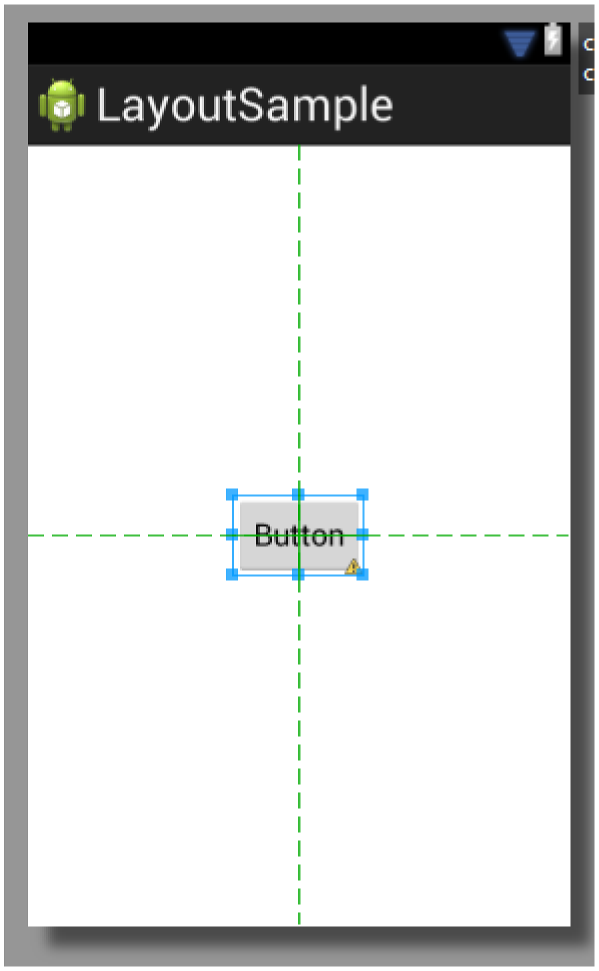


- **Palette:** This contains the different categories of views that are available with the Android SDK for inclusion in user interface designs.
- **Graphical Layout:** A display view that could see how the APP will display directly.
- **Outline:** Show all views you add in layout.
- **View Properties:** View properties sheet displays a list of all the properties available for the currently selected view object.
- **Layout Tool:** Let you do some settings for your APP. Such as screen orientation, App theme, device…
- **Tool Bar:** Some shortcuts of typesettings and zoom.
- **Tab:** Switch between graphical and programmatic design.

11. To add a view to the user interface, simply locate it in the palette and drag and drop it onto the visual representation of the device display anywhere you want.





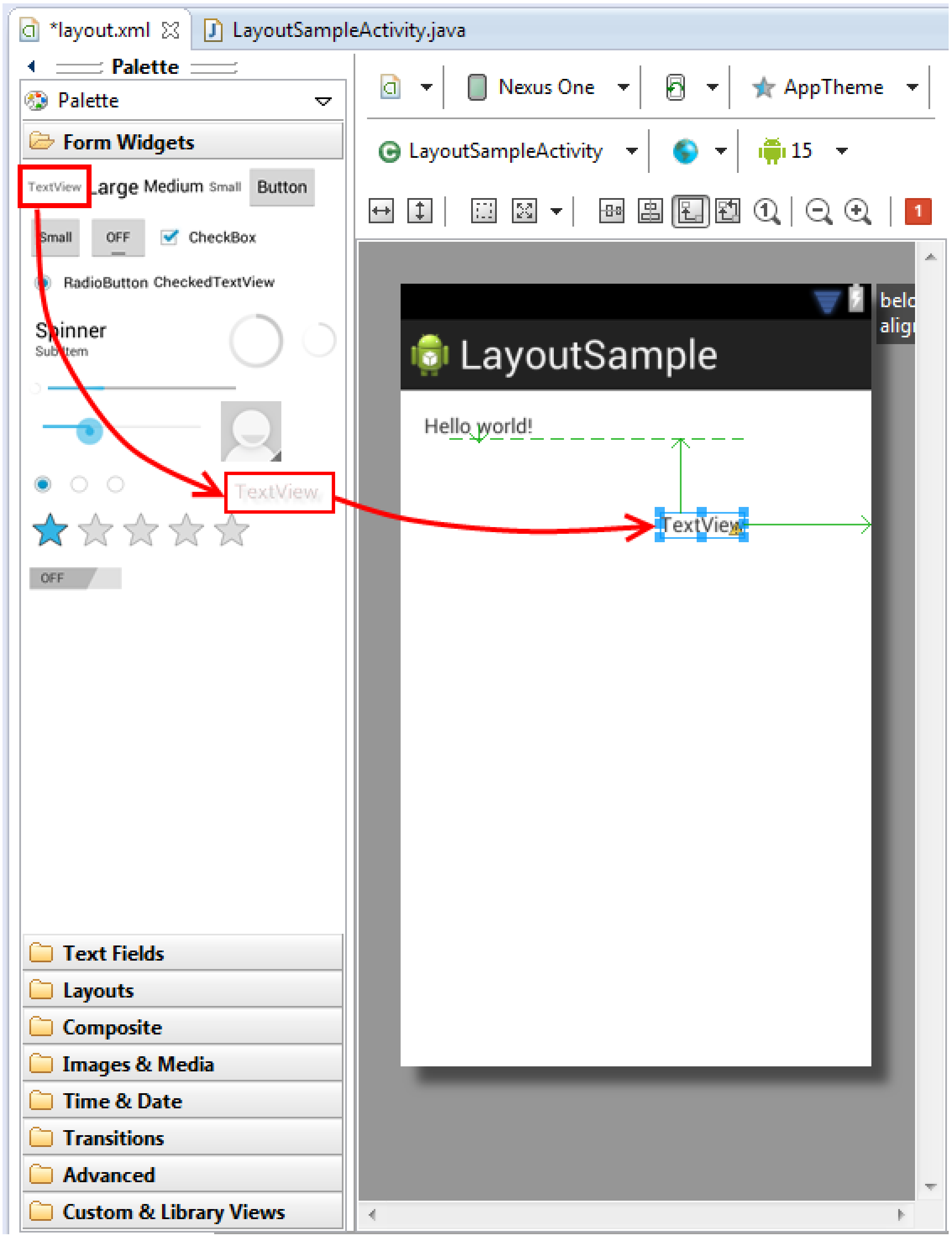


- ☐ You could find many resources on web about UI design. Keyword: “*eclipse*”, “*android*”, “*ui design*”, “*layout*”. Above refer to:
  - ➢ http://www.techotopia.com/index.php/Designing_an_Android_User_Interface_using_the_Graphical_Layout_Tool

  XML:
  - ➢ http://mobile.tutsplus.com/tutorials/android/android-layout/

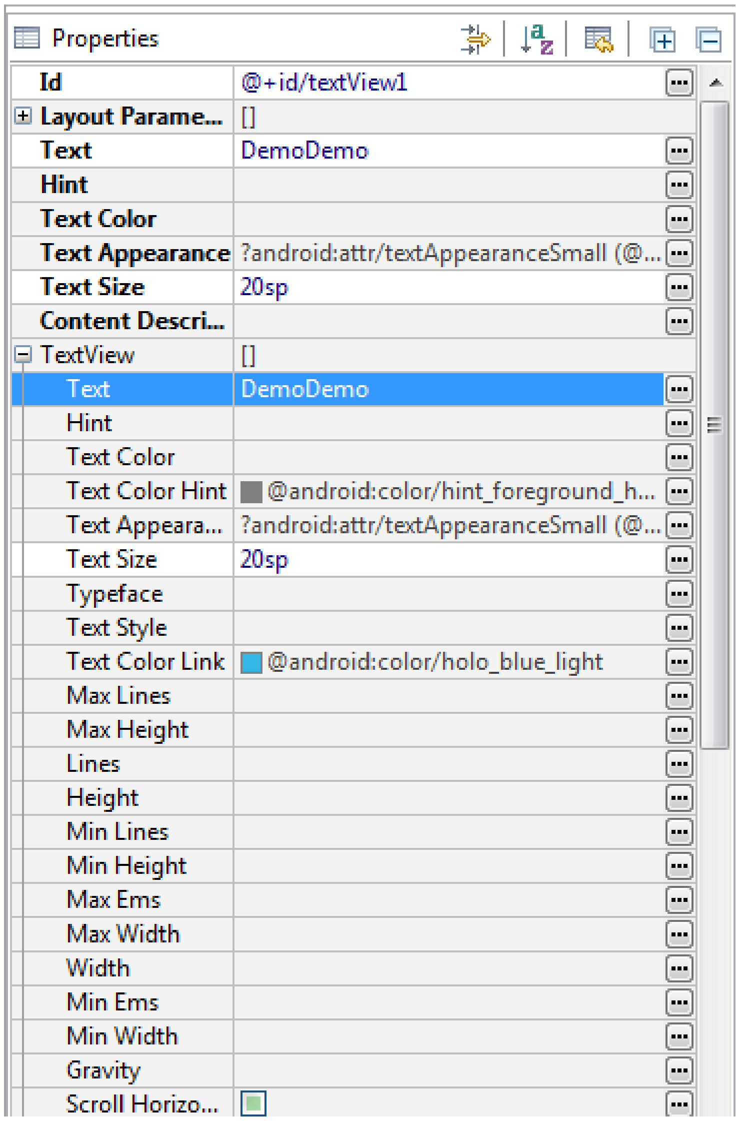
3.2 Add a Text View (Add resources file)

1. Drag a **TextView** directly to the display view.



2. Change text content to “DemoDemo”. You also can try to modify other properties.

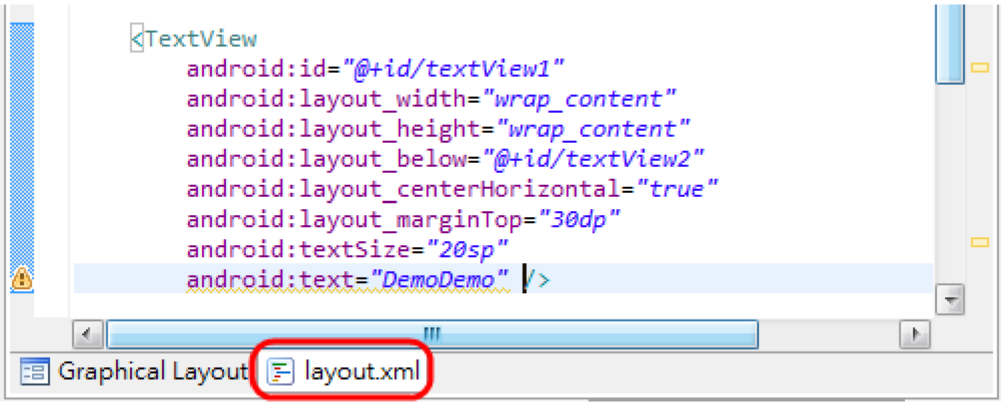
(View Properties Sheet)



Properties

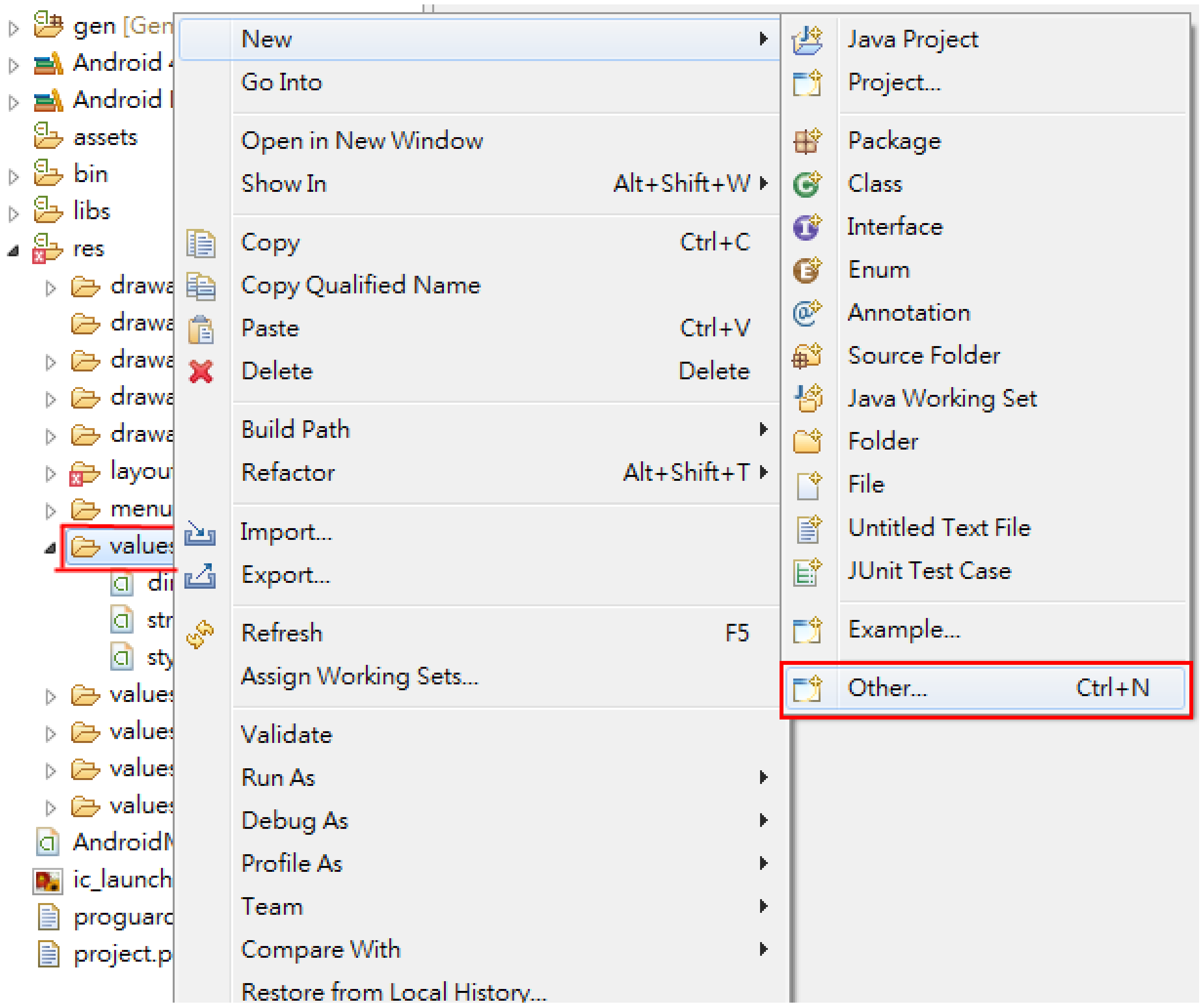
Id	@+id/textView1
⊞ **Layout Parame...**	[]
Text	DemoDemo
Hint	
Text Color	
Text Appearance	?android:attr/textAppearanceSmall (@...
Text Size	20sp
Content Descri...	
⊟ TextView	[]
Text	DemoDemo
Hint	
Text Color	
Text Color Hint	■ @android:color/hint_foreground_h...
Text Appeara...	?android:attr/textAppearanceSmall (@...
Text Size	20sp
Typeface	
Text Style	
Text Color Link	🟦 @android:color/holo_blue_light
Max Lines	
Max Height	
Lines	
Height	
Min Lines	
Min Height	
Max Ems	
Max Width	
Width	
Min Ems	
Min Width	
Gravity	
Scroll Horizo...	☐

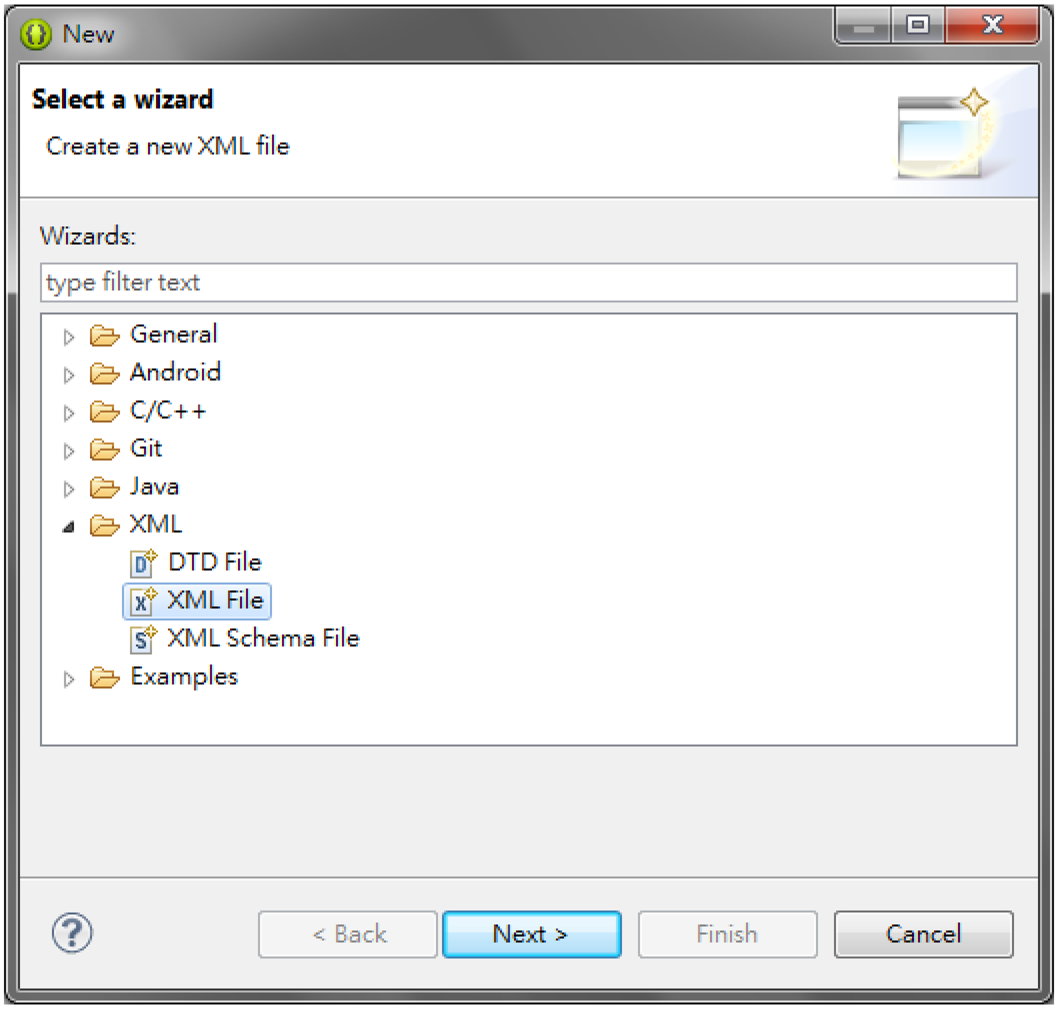
(XML)

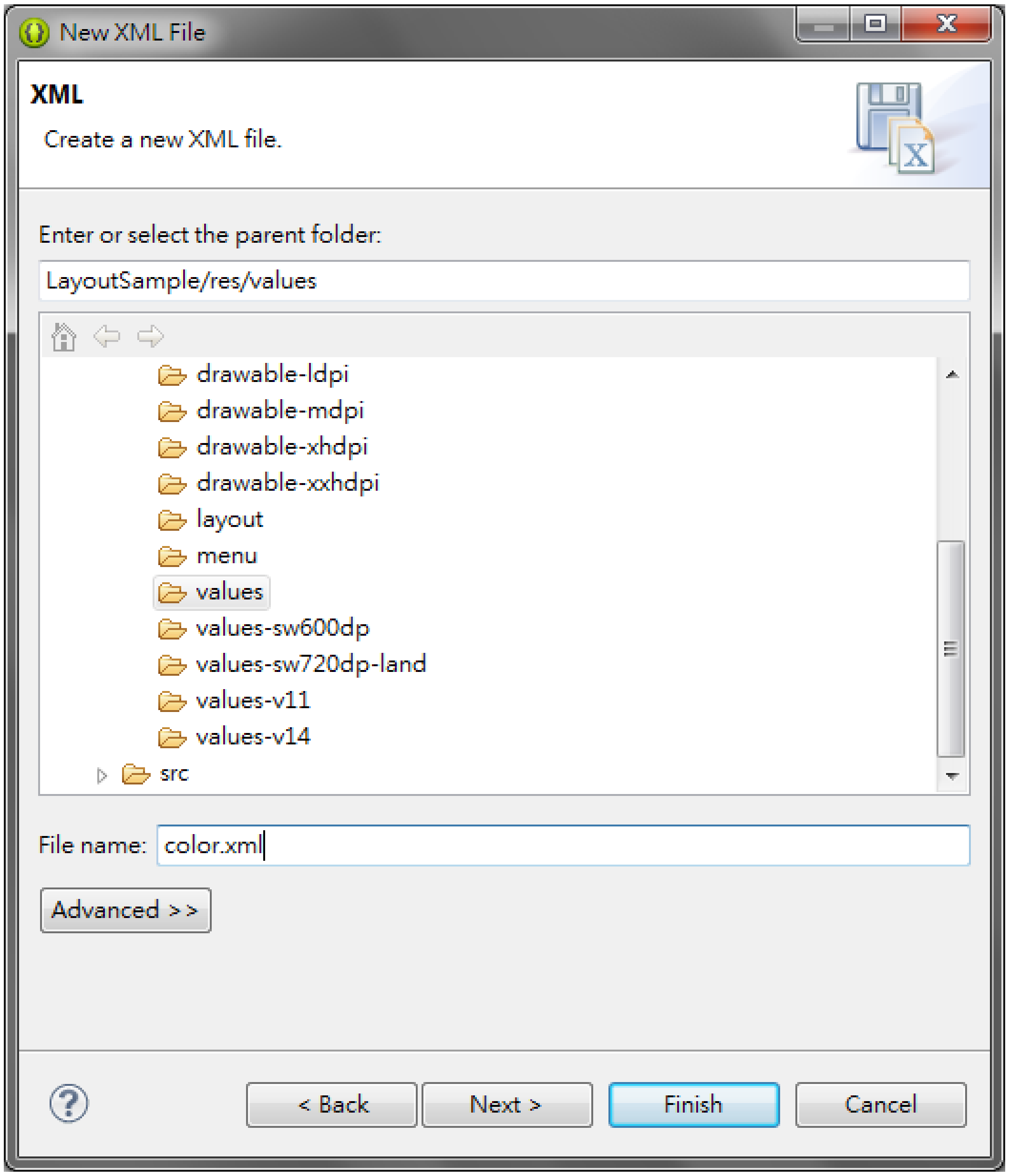


```
<TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textView2"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="30dp"
    android:textSize="20sp"
    android:text="DemoDemo" />
```

3. Then, you need to add some resources such as color by yourself for setting properties. Create a XML file in `res/values` called `color.xml`.




New
Select a wizard
Create a new XML file
Wizards:
type filter text
General
Android
C/C++
Git
Java
XML
DTD File
XML File
XML Schema File
Examples
< Back
Next >
Finish
Cancel



XML

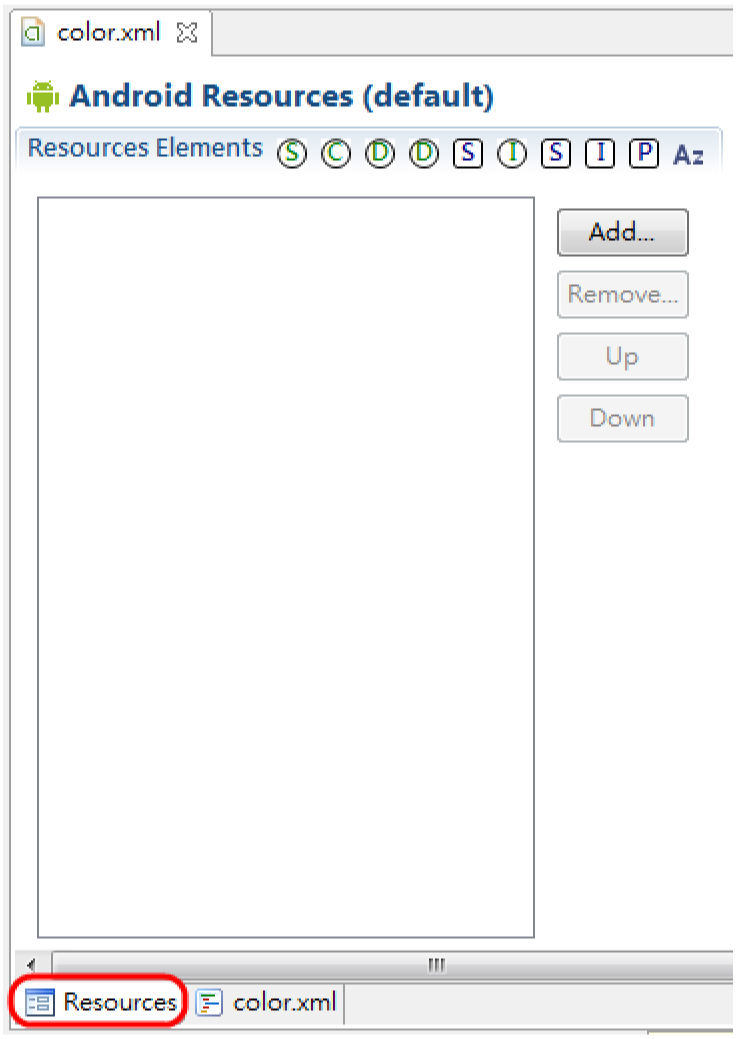
Create a new XML file.

Enter or select the parent folder:

LayoutSample/res/values

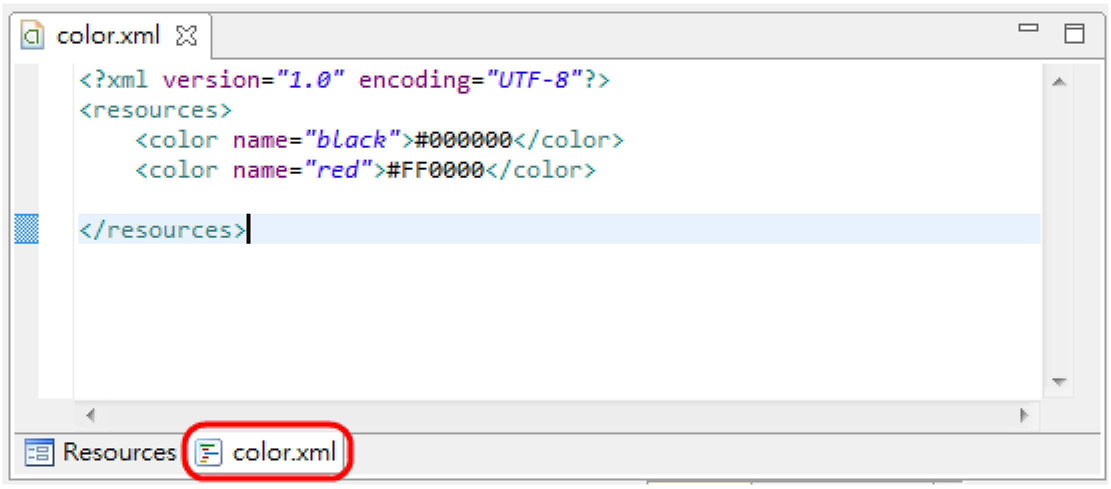
- drawable-ldpi
- drawable-mdpi
- drawable-xhdpi
- drawable-xxhdpi
- layout
- menu
- values
- values-sw600dp
- values-sw720dp-land
- values-v11
- values-v14
- src

File name: color.xml



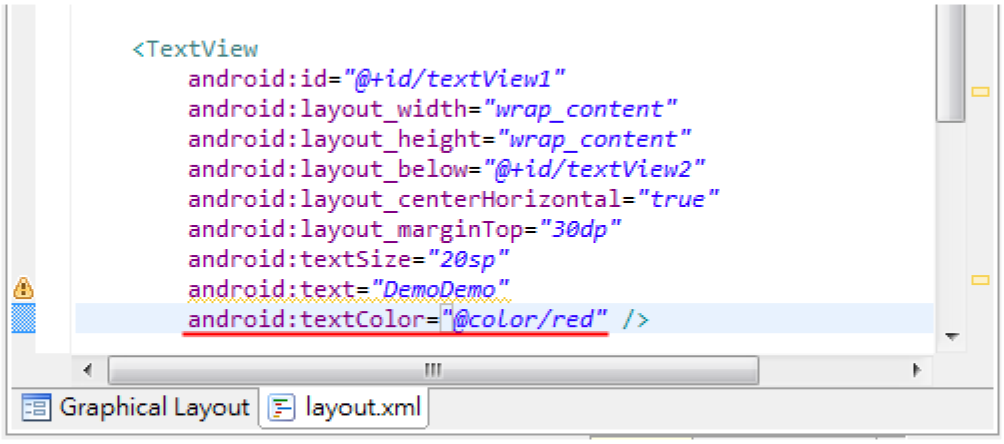
4. Add some color resources to your application so that we can use them in our app.

- `<color> Name="black" Value="#000000"`
- `<color> Name="red" Value="#FF0000"`

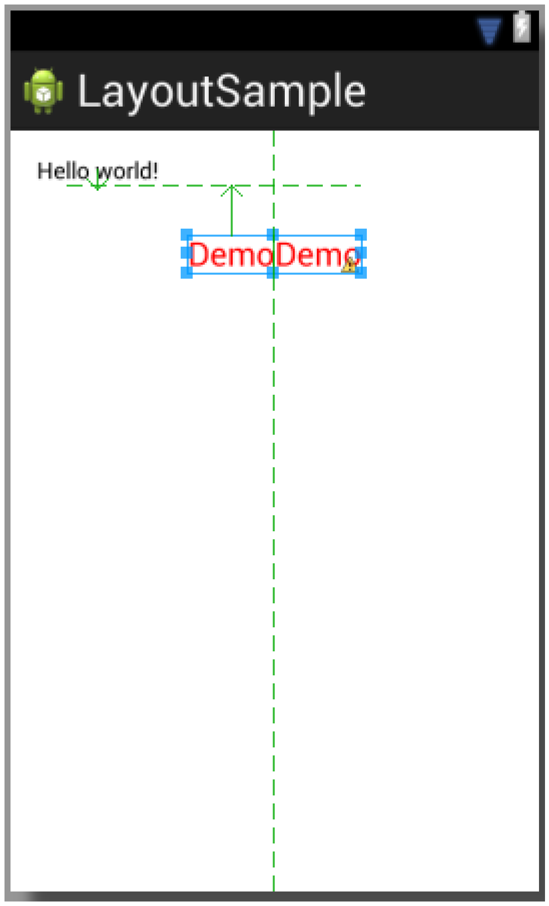


```
<?xml version="1.0" encoding="UTF-8"?>
<resources>
    <color name="black">#000000</color>
    <color name="red">#FF0000</color>

</resources>
```

5. Now you can use the color you define.



```
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView2"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="30dp"
        android:textSize="20sp"
        android:text="DemoDemo"
        android:textColor="@color/red" />
```

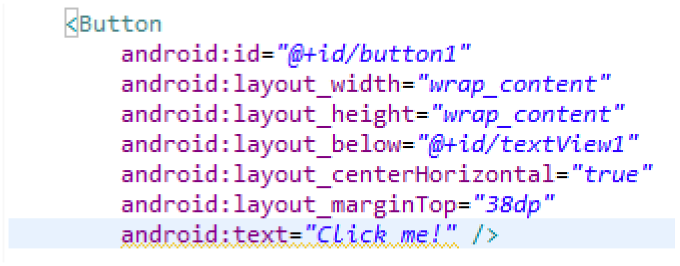


3.3 Add a Button (Add Activity)

With your button in layout, you need to handle clicks on this button. Otherwise, it would no activity when you click.

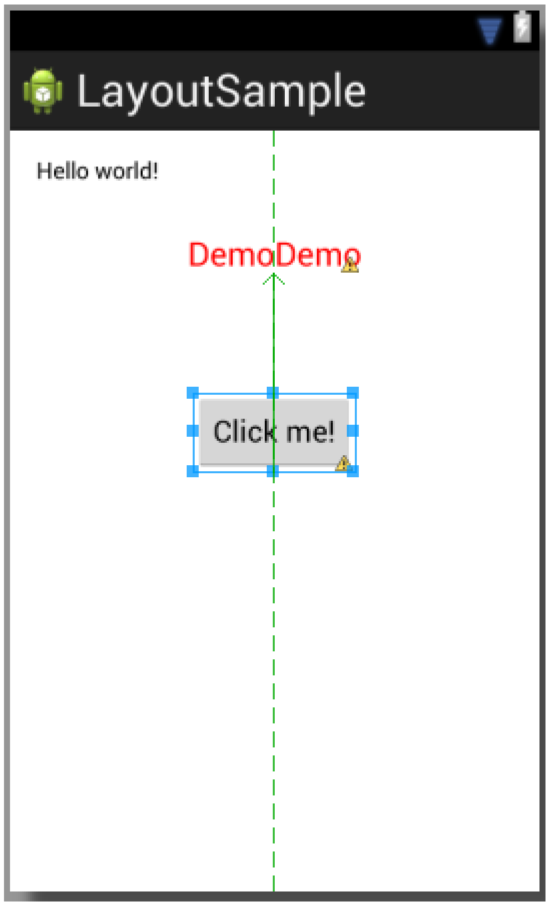
1. Change the text content.

(XML)



```
<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textView1"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="38dp"
    android:text="Click me!" />
```

(*Properties Sheet*)

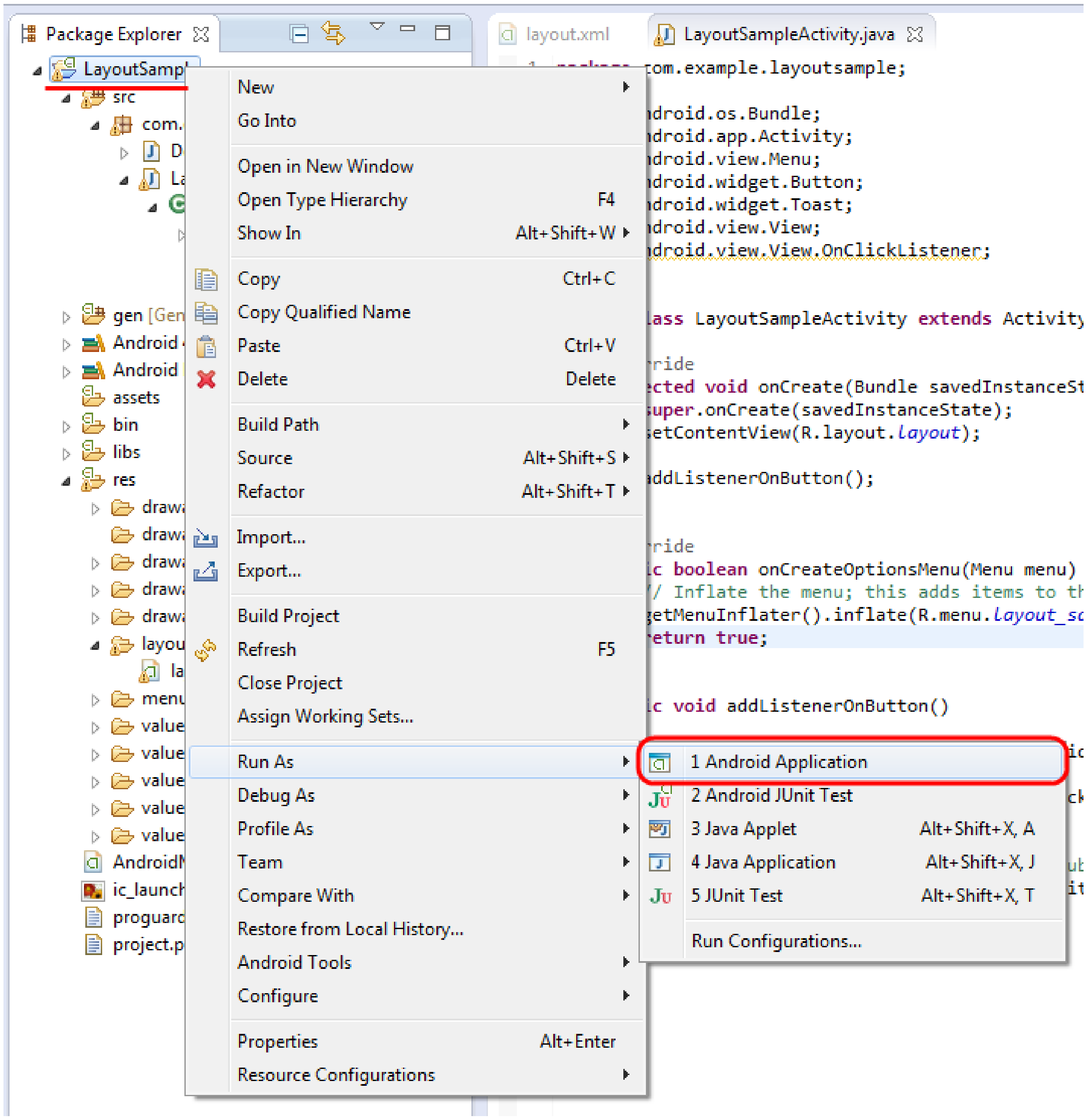


2. Go to the java file of the activity and set an OnClickListener for the button.

```
package com.example.layoutsample;

import android.os.Bundle;
import android.app.Activity;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class LayoutSampleActivity extends Activity{

        private Button mBtn1;
        private TextView mTxt1;

        /* Called when the activity is first created */
        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.layout);

                addButtonClickListener();
        }

        public void addButtonClickListener()
        {
                mBtn1 = (Button) findViewById(R.id.button1);
                mTxt1 = (TextView) findViewById(R.id.textView1);

                mBtn1.setOnClickListener(new View.OnClickListener()
                {
                        @Override
                        public void onClick(View v)
                        {
                                // TODO Auto-generated method stub
                                mTxt1.setText("Hi, welcome!!");
                                Toast.makeText(LayoutSampleActivity.this, "Button
clicked!", Toast.LENGTH_SHORT).show();
                        }
                });
        }
}
```

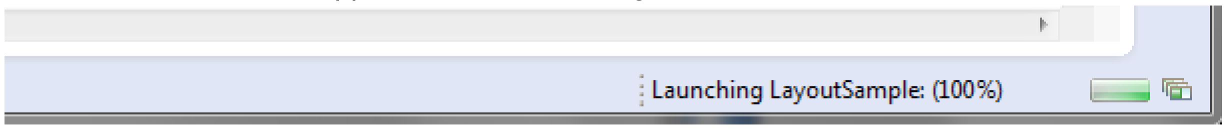
The **`Toast`** component in the code display short pop up messages to the user that automatically disappear in few seconds. This activity is when you click the button, the text “DemoDemo” will change to “Hi, welcome!!” and display a pop up message “Button clicked!”

3. Now you can run as Android application on your device. Connect your device to PC. When it’s attached, right-click on the project name and select **Run As** → 1 **Android Application.**

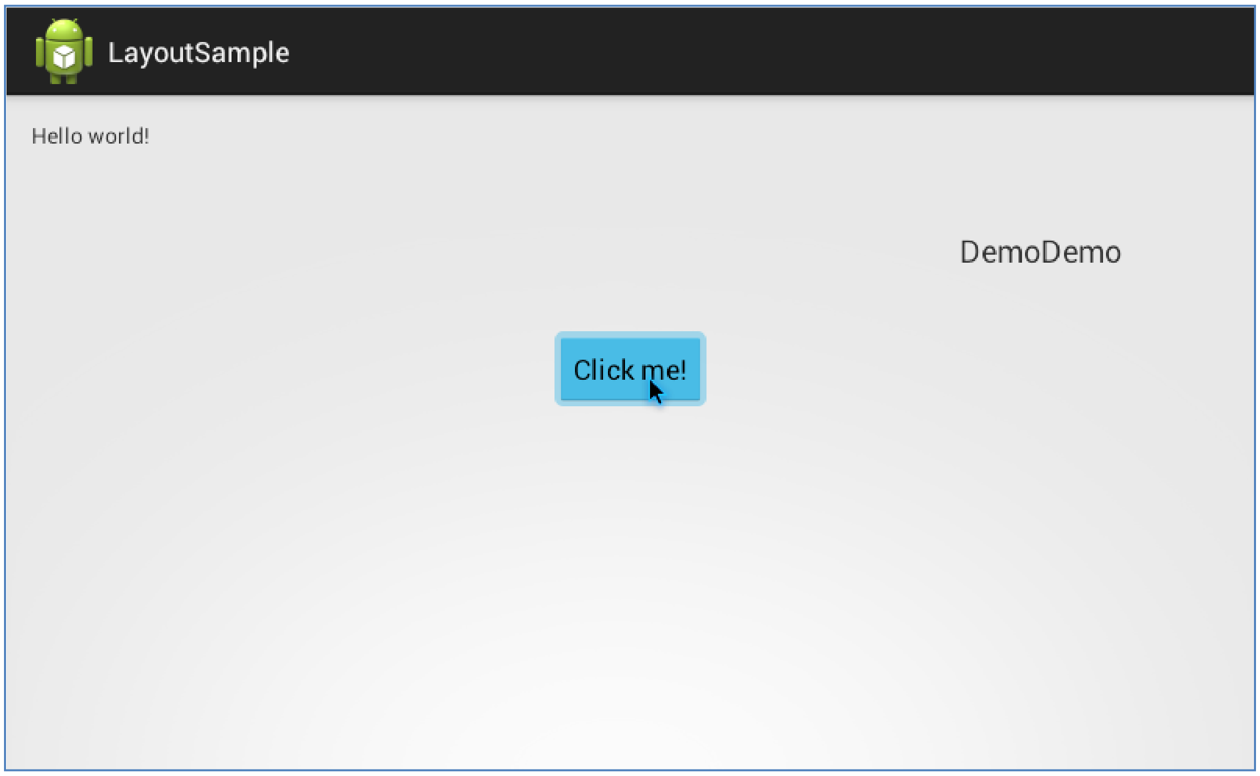


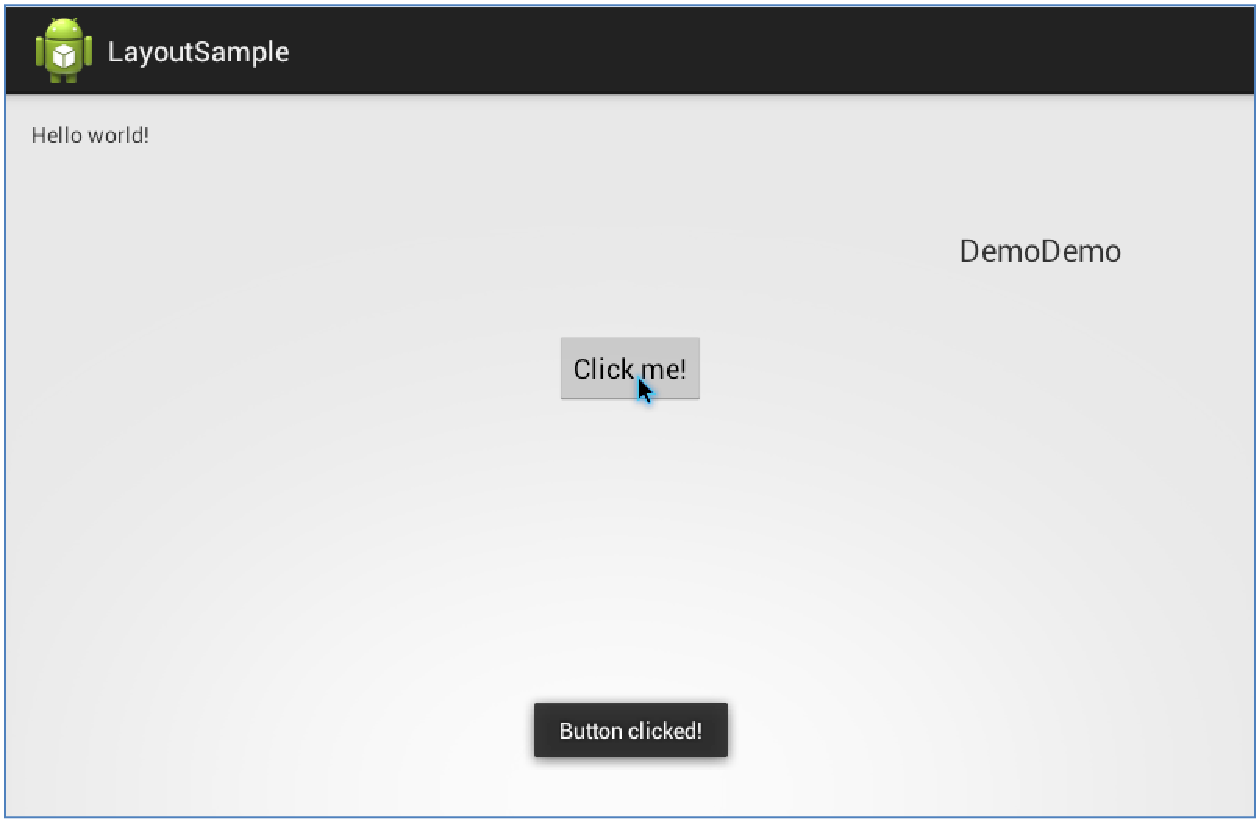
```
com.example.layoutsample;

ndroid.os.Bundle;
ndroid.app.Activity;
ndroid.view.Menu;
ndroid.widget.Button;
ndroid.widget.Toast;
ndroid.view.View;
ndroid.view.View.OnClickListener;

lass LayoutSampleActivity extends Activity

ride
ected void onCreate(Bundle savedInstanceSt
uper.onCreate(savedInstanceState);
etContentView(R.layout.layout);

addListenerOnButton();


ride
ic boolean onCreateOptionsMenu(Menu menu)
/ Inflate the menu; this adds items to th
etMenuInflater().inflate(R.menu.layout_sc
eturn true;

ic void addListenerOnButton()
```

Wait for a while until App launch. The lower right corner will show the status.



4. Then on device, it will show the application. Click the button!

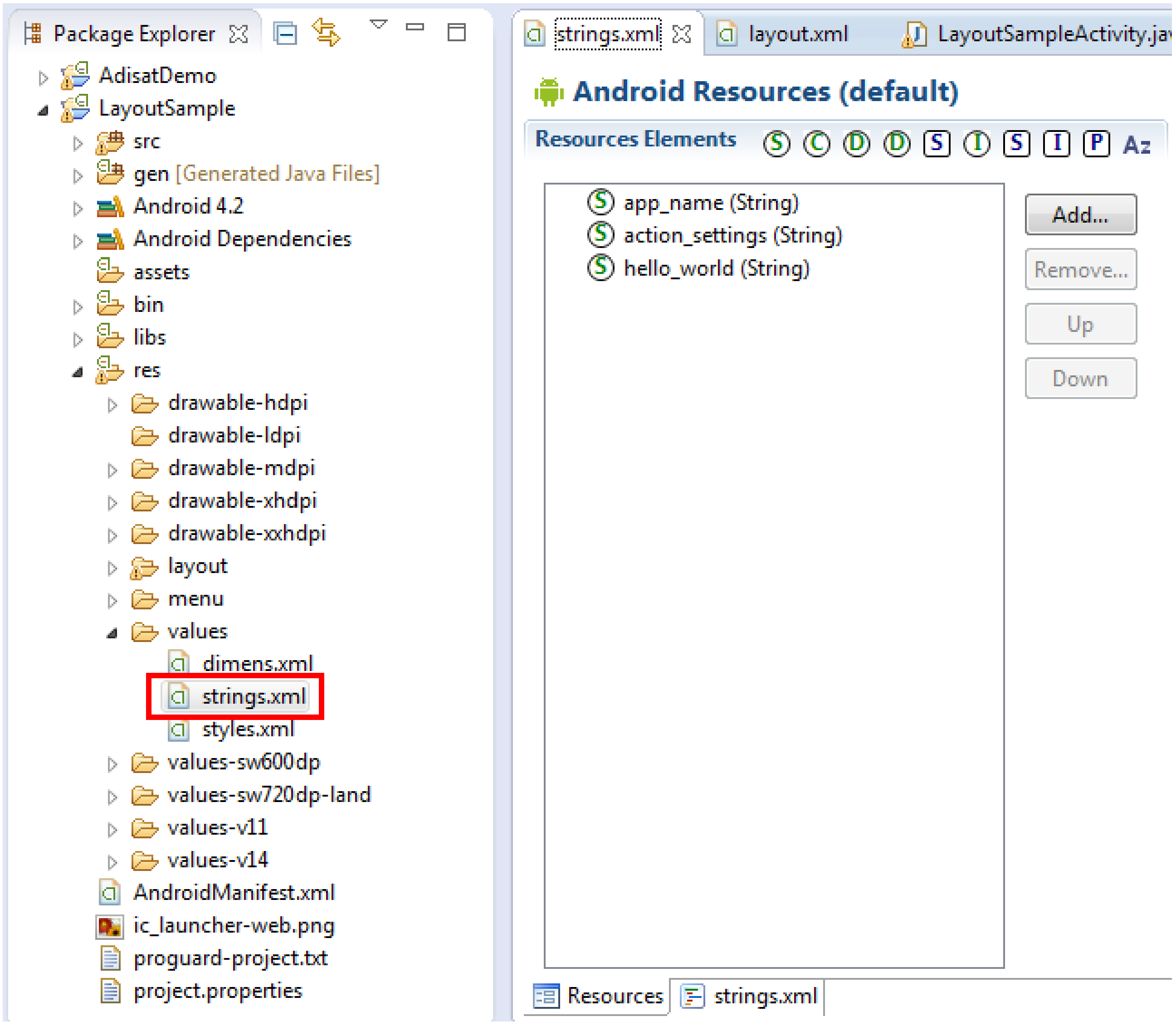




3.4 Add a Image

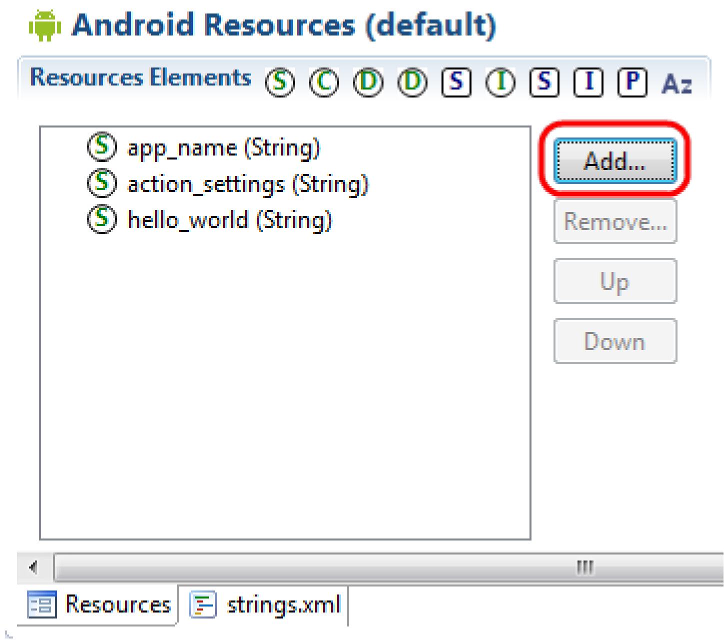
You can use the `ImageView` component to display images in your Application.

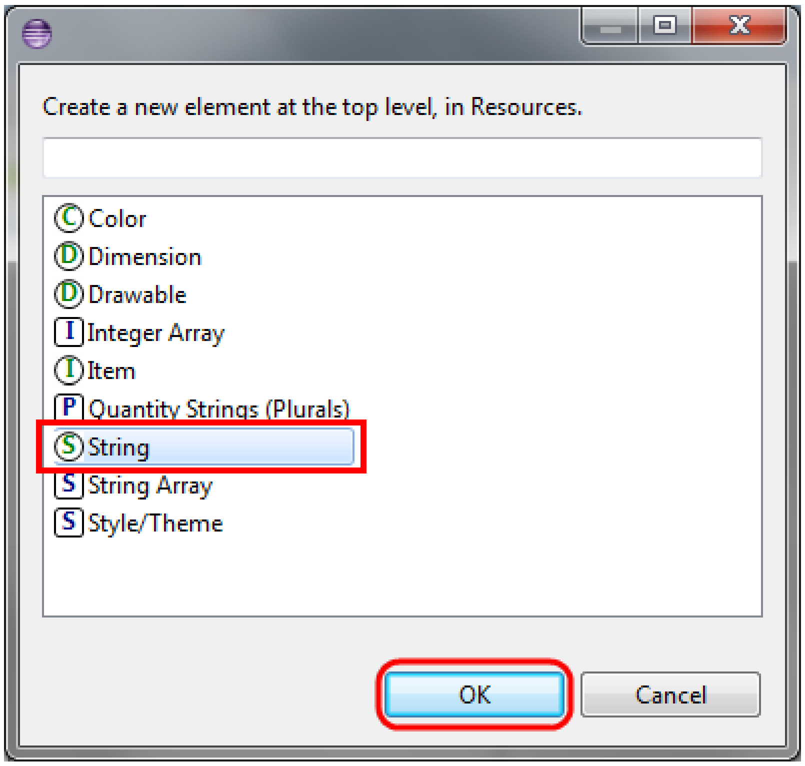
1. First, adding resources. Open the `res/values/strings.xml` file and you will see the graphical Resources View editor:

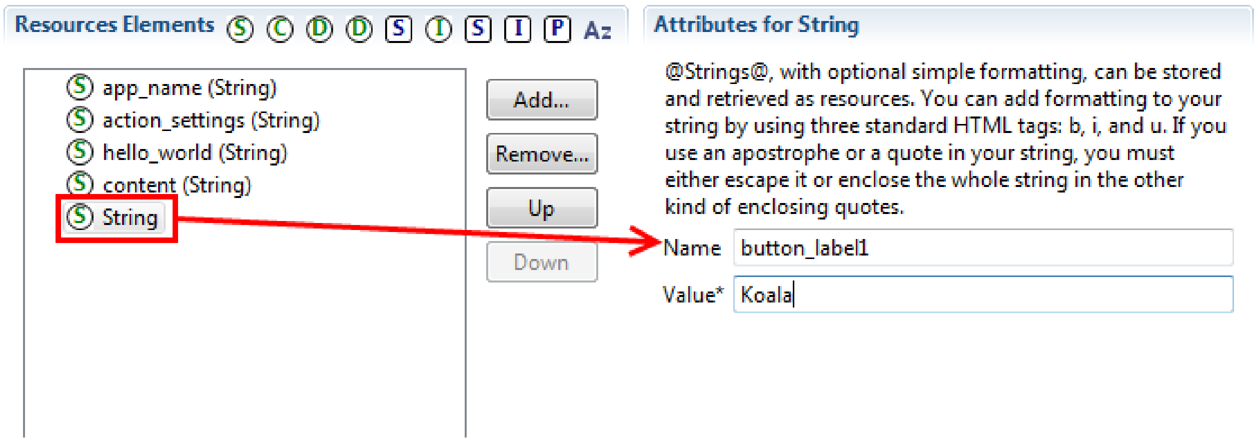


2. Add some string resources to your application so that we can use in many ways and in many places in our app.
   - `<string> Name="button_label1" Value="Koala"`
   - `<string> Name="content" Value="image"`

   Here we show how to add a string resource and you can do another by yourself.




Create a new element at the top level, in Resources.



Resources Elements

app_name (String)
action_settings (String)
hello_world (String)
content (String)
String

Add... Remove... Up Down

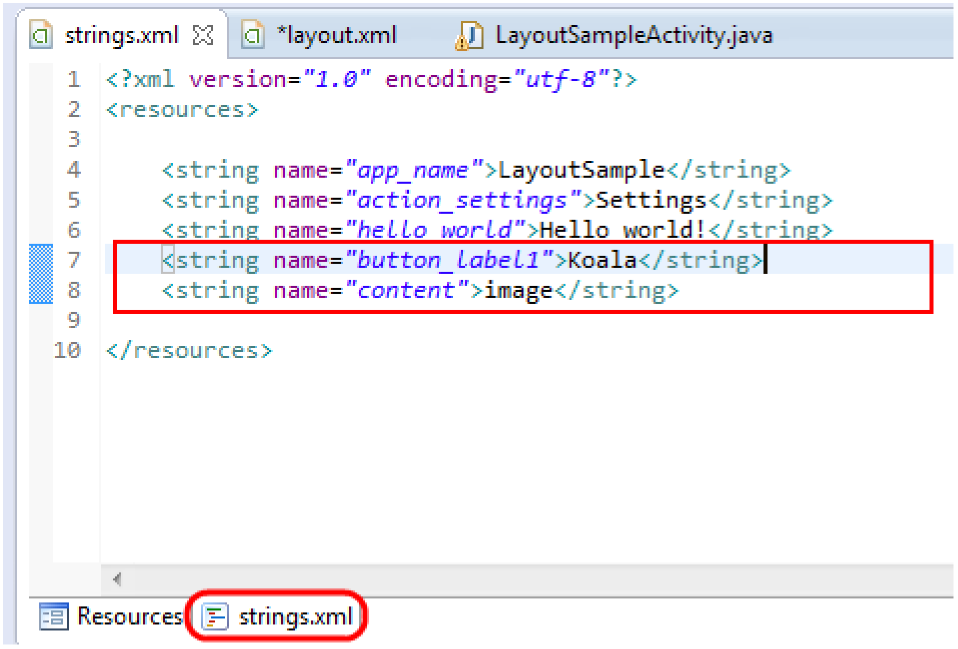
Attributes for String

@Strings@, with optional simple formatting, can be stored and retrieved as resources. You can add formatting to your string by using three standard HTML tags: b, i, and u. If you use an apostrophe or a quote in your string, you must either escape it or enclose the whole string in the other kind of enclosing quotes.

Name: button_label1

Value*: Koala

Press the `string.xml` tab and see the code in XML.

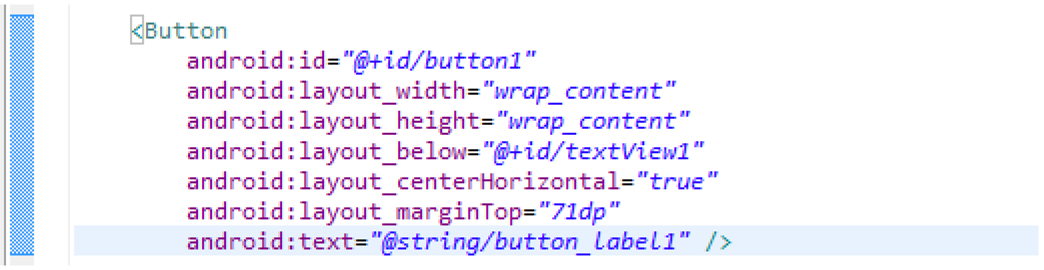


strings.xml | *layout.xml | LayoutSampleActivity.java

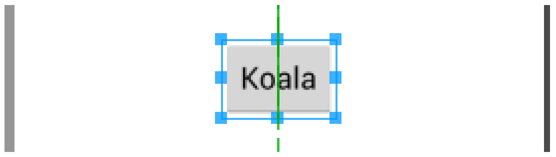
```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">LayoutSample</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string name="button_label1">Koala</string>
    <string name="content">image</string>

</resources>
```

Resources | strings.xml

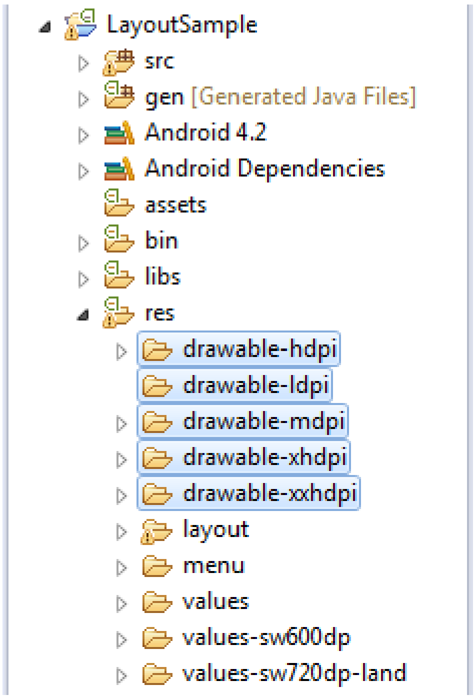
Change the button label.



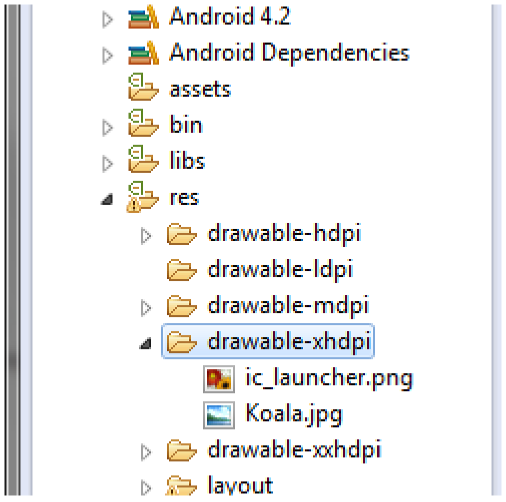
```
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView1"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="71dp"
        android:text="@string/button_label1" />
```



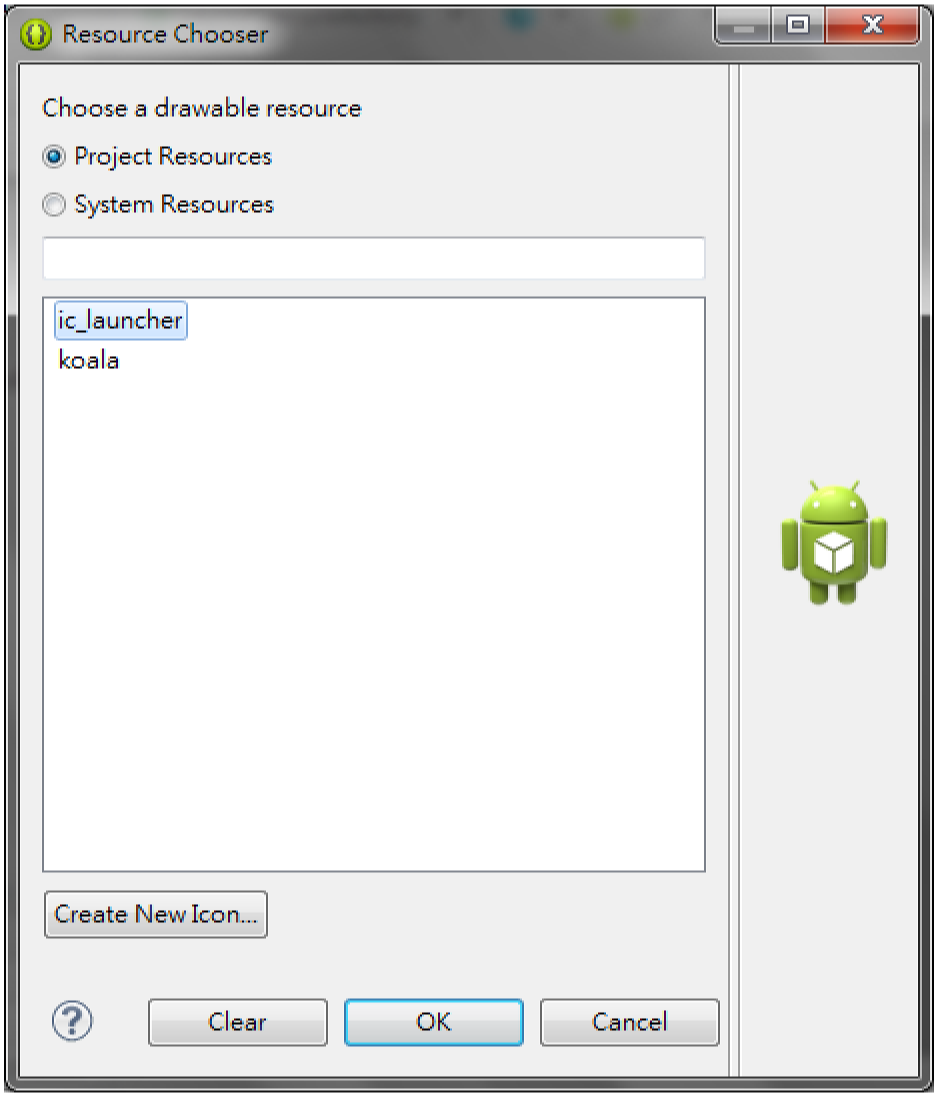
3. As you would have noticed in the Package explorer, there are 4 or more folders that contain the image resources of the project:
   - `res/drawable-hdpi`
   - `res/drawable-ldpi`
   - `res/drawable-mdpi`
   - `res/drawable-hdpi`

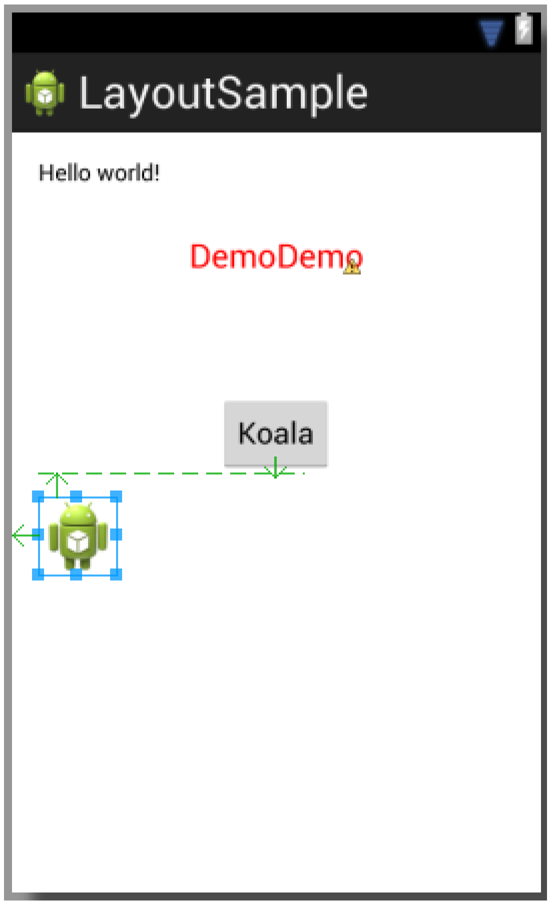


4. You can copy the images you want to put in your application in any one of these folders you want. Android SDK will automatically recognize any images you put on any one of these folders as resources. The image names better start with a lowercase letter. If the image does not appear in the Package Explorer under the folder you've copied it into, right-click on the project name and select **Refresh**. Now the image should be under the correct folder.



5. Now create the image view to your application. Open `layout.xml` and drag an **ImageView** to layout.





6. Open the java file that contains the code of the activity and add following code:

```
package com.example.layoutsample;

import android.os.Bundle;
import android.app.Activity;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.ImageView;

public class LayoutSampleActivity extends Activity{

        private Button mBtn1;
        private TextView mTxt1;
        private ImageView img1;

        /* Called when the activity is first created */
        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.layout);

                addButtonClickListener();
        }

        public void addButtonClickListener()
        {
                mBtn1 = (Button) findViewById(R.id.button1);
                mTxt1 = (TextView) findViewById(R.id.textView1);
                img1 = (ImageView) findViewById(R.id.imageView1);

                mBtn1.setOnClickListener(new View.OnClickListener()
                {
                        @Override
                        public void onClick(View v)
                        {
                                // TODO Auto-generated method stub
                                mTxt1.setText("Change the image!!");
                                img1.setImageResource(R.drawable.koala);
                        }
                });
        }
}
```

7. Then run the application.

Chapter 4: Sample Application

This chapter contains with descriptions of the sample applications product attached.

Sample1: Elevator

The `ICNElevator` Java class is a simple elevator **activity** in the Android application. It puts you to a role of manager so that you can manage the `ICNImageButton` class you create. You can extend your code from the `ICNElevator` Java class and then rewrite your code using the `InitUserButtonSampleCode()` function which can be founded in the the **`SampleUserElevator.java`** file from the `ICNMain` project.

- The library projects: **ICNService, ICNImageButtonMgr**

=== InitUserButtonSampleCode() Function ===

```
public void InitUserButtonSampleCode() {

        rlv = new RelativeLayout( this );
        int count = 4;
        IBInfo[] ibinfolist = new IBInfo[count];

        DisplayMetrics dm = detScreenSize();

        final int button_width = 500;
        final int button_height = 300;
        int xpos = 0;
        int ypos = -1 * button_height;
        for (int i = 1; i <= count; i++){
                ypos += button_height;

                if (ypos > dm.heightPixels || ypos + button_height >
dm.heightPixels){
                        ypos = 0;
                        xpos += button_width;
                }

                if (xpos > dm.widthPixels || xpos + button_width >
dm.widthPixels)
                        break;

                addButton(new ICNImageButton(this, i);
                ibinfolist[i - 1] = new IBInfo(xpos, ypos,
                        button_width, button_height, "",
                        R.drawable.bule_square_button, Color.BLUE,
                        ICNImageButton.FONT_ID.SANSUS, "");
                getButton(i - 1).setClickedImageId(R.drawable.red_square_button);
                getButton(i - 1).setFontSize( 160 );
                getButton(i - 1).createButton(ibinfolist[i - 1]);

                rlv.addView(getButton(i - 1));
        }
        init(ICNService.ICN_COMMAND_ID.ICD_ELEVATOR_DEMO, rlv);

        setContentView( rlv );
}
```

=== Function & Parameter Explanation ===

1 `DisplayMetrics ICNService::getScreenSize()`
It's a function in ICNService. Call it to get your current screen resolution size. Once you get the Display Metrics information, you can adjust the position of your screen position according to the button size specificed.

2 `ICNImageButton Object`
Every ICNImageButton object is corresponding to each button you see in the GUI. To use **`new`** to construct it, you need the current activity and *button id*. The button id is an integer number. Every button must have a unique number. When you click one button, the system service will use it to make and send bottom-down commands to your smart display

3 `void ICNElevator::addButton (ICNImageButton)`
It adds an image button to the ICNElevator manager.

4 `ICNImageButton ICNElevator::getButton(int i)`
Integer i starts from 0. This is just an index which the image button has been added to. This function gets the ICNImageButton object.

5 `void ICNImageButton::setClickedImageId(int iid)`
Integer iid is a drawable id stored in the res folder. You assign your button with a specified picture. When you click the button, it will display that picture.

6 `void ICNImageButton::setFontSize(int size)`
The integer size is your font size. You should use a good font size for your text or number so that it would look compatible with your button.

7 `void ICNImageButton::createButton(IBInfo ibinfo)`
Using createButton with the IBInfo object, you can generate a button body. The IBInfo object constructor needs the following input parameters:

7.1 `int x`：the x-coordinate

7.2 `int y`：the y-coordinate

7.3 `int w`：the width of button

7.4 `int h`：the height of button

7.5 `String img`：specify the absolute file path of your background picture. (For example, /sdcard/imag/xxxx.bmp) This is an optional input. If you did not use this, you can just set it as “nul”l string. Once you set it, its priority will be higher than the next parameter.

7.6 `int did`：it is a drawable id in your res folder of the library project. If you did not set the previous parameter, the button will use this id as its background picture.

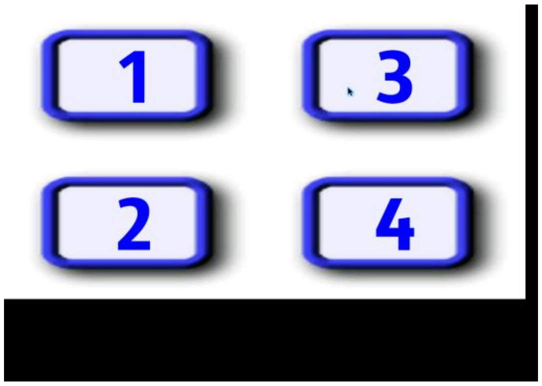
7.7 `int textcolor`：This describes the color for your output.

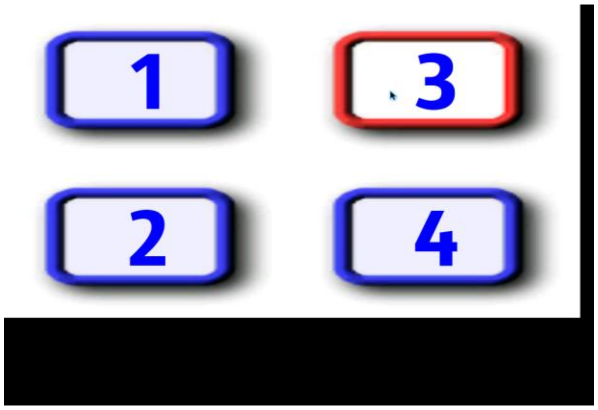
7.8 `ICNImageButton.`*`FONT_ID fontid`*: you can choose the font here. We provide three fonts: ACENS, MINIS and SANSUS for you to choose from. The default font is /system/fonts/DroidSerif-Bold.ttf

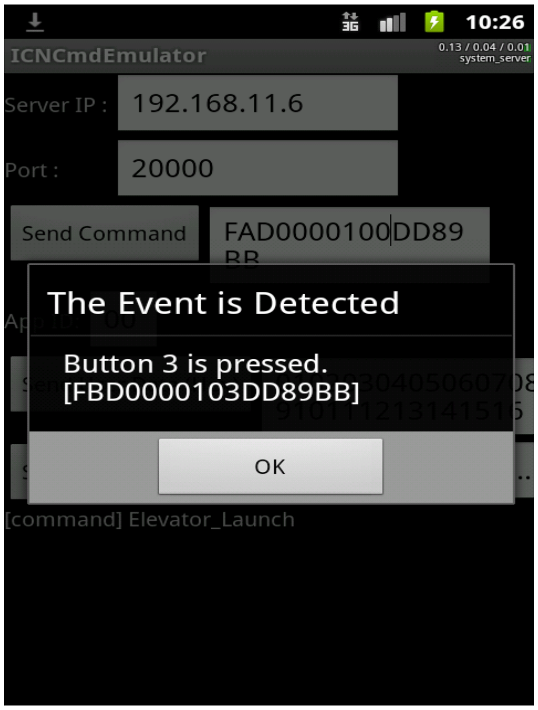
7.9 `String text`：Your output text on ICNImageButton. You can keep this as a null string and then ICNImageButton will use the *button id* you set in the constructor time. Once you set this parameter, ICNImageButton will replace the *button id* with the text and this will appear in the output.

=== Demo ===

A. If you use ICNCmdEmulator tool to launch Elevator, when you click the button *3*, you will receive a command in the ICNCmdEmulator







Server IP : 192.168.11.6

Port : 20000

FAD0000100DD89BB

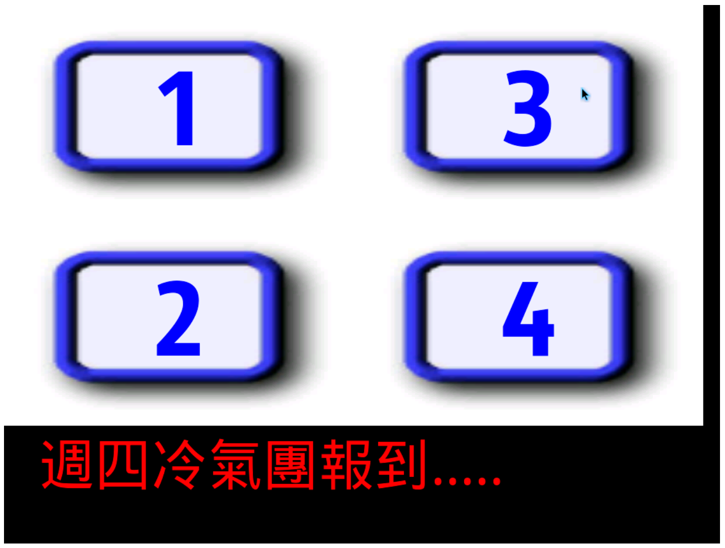
The Event is Detected

Button 3 is pressed.
[FBD0000103DD89BB]

[command] Elevator_Launch

B. If you use ICNCmdEmulator tool to send a button-up command like *FAD0000203DD89BB* to the elevator, pressing the button *3* will restore the colour to blue again.

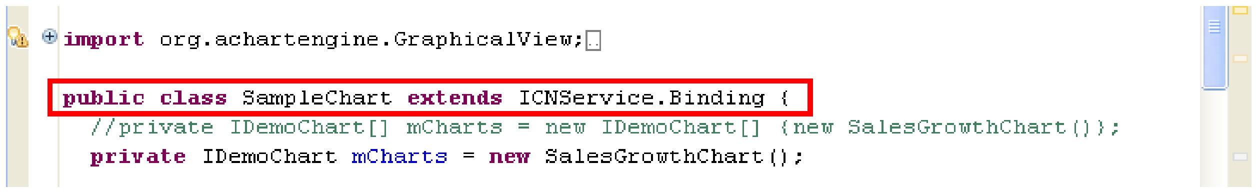
c. If you click the button *Send User Text (UTF-8)* to send a user text to the elevator, it will show you the text you set previously in the ICNCmdEmulator.



Sample2: Chart

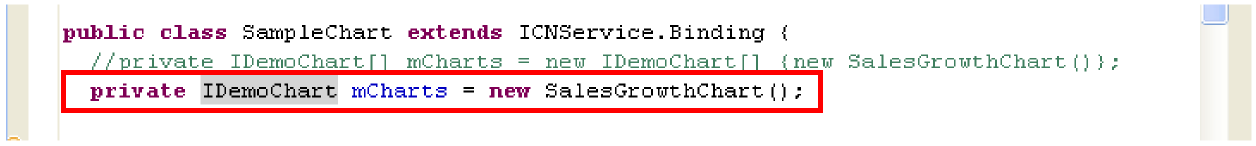
This section describes how to add an application to your `Chart` Project.

1. Declare a new project named `SampleChart` that extends the `ICNService.Binding` class.



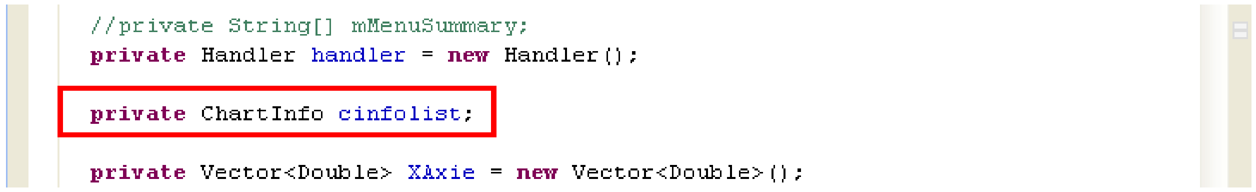
```
import org.achartengine.GraphicalView;

public class SampleChart extends ICNService.Binding {
    //private IDemoChart[] mCharts = new IDemoChart[] {new SalesGrowthChart()};
    private IDemoChart mCharts = new SalesGrowthChart();
```

2. Use `IDemoChart` to add a new class object named `mCharts`. Assign `mCharts` to `SalesGrowthChart()`.



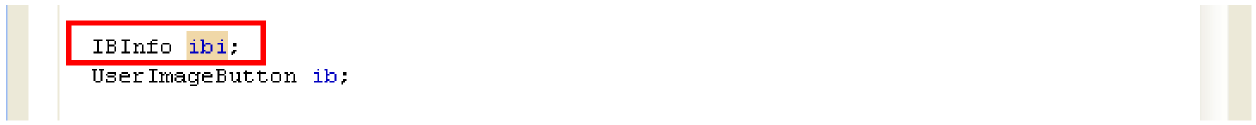
```
public class SampleChart extends ICNService.Binding {
    //private IDemoChart[] mCharts = new IDemoChart[] {new SalesGrowthChart()};
    private IDemoChart mCharts = new SalesGrowthChart();
```

3. Use `ChartInfo` to establish a parameter object for Chart named `cinfolist`.



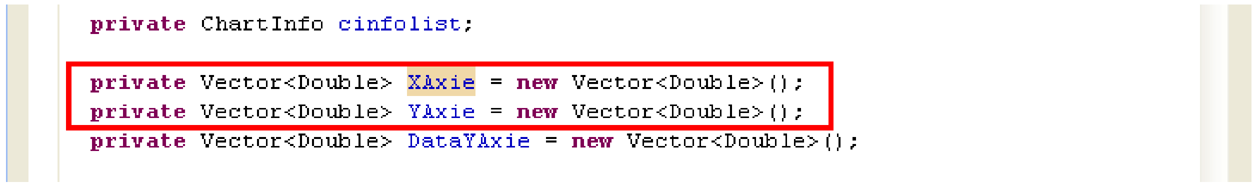
```
    //private String[] mMenuSummary;
    private Handler handler = new Handler();

    private ChartInfo cinfolist;

    private Vector<Double> XAxie = new Vector<Double>();
```

4. Use `IBInfo` to establish a parameter object for ImageButtonn named `ibi`.



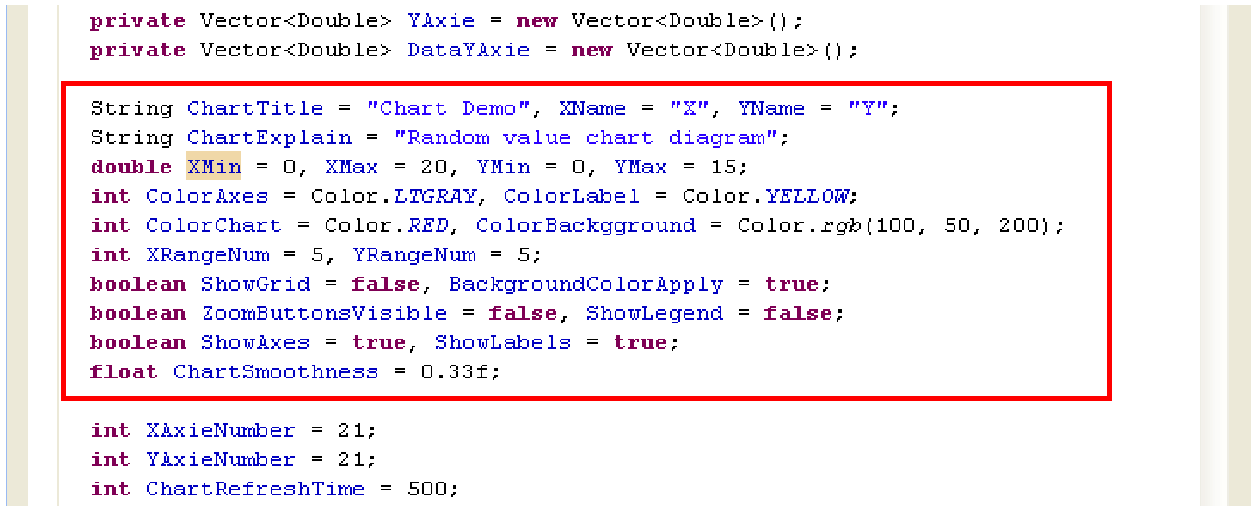
```
    IBInfo ibi;
    UserImageButton ib;
```

5. Add two `Double` data types for each x-axis and y-axis respectively.



```
    private ChartInfo cinfolist;

    private Vector<Double> XAxie = new Vector<Double>();
    private Vector<Double> YAxie = new Vector<Double>();
    private Vector<Double> DataYAxie = new Vector<Double>();
```

6. Declare the variables that will be used in `ChartInfo()`.



```
private Vector<Double> YAxie = new Vector<Double>();
private Vector<Double> DataYAxie = new Vector<Double>();

String ChartTitle = "Chart Demo", XName = "X", YName = "Y";
String ChartExplain = "Random value chart diagram";
double XMin = 0, XMax = 20, YMin = 0, YMax = 15;
int ColorAxes = Color.LTGRAY, ColorLabel = Color.YELLOW;
int ColorChart = Color.RED, ColorBackgground = Color.rgb(100, 50, 200);
int XRangeNum = 5, YRangeNum = 5;
boolean ShowGrid = false, BackgroundColorApply = true;
boolean ZoomButtonsVisible = false, ShowLegend = false;
boolean ShowAxes = true, ShowLabels = true;
float ChartSmoothness = 0.33f;

int XAxieNumber = 21;
int YAxieNumber = 21;
int ChartRefreshTime = 500;
```

--- Variable definition---

`String ChartTitle` : the Boolean value to specify whether Chart's title name is displayed

`String XName` : the name of Chart's x-axis

`String YName` : the name of Chart's y-axis

`String ChartExplain` : Chart's text description at the bottom left corner

`double XMin` : the minimum Chart's x-axis value

`double XMax` : the maximum Chart's x-axis value

`double YMin` : the minimum Chart's y-axis value

`double YMax` : the maximum Chart's y-axis value

`int ColorAxes` : Chart's axes color

`int ColorLabel` : Chart's label color

`int ColorChart` : Chart's line color

`int ColorBackgground` : Chart's background color

`int XRangeNum` : the number of Chart's x-axis intervals

`int YRangeNum` : the number of Chart's y-axis intervals

`boolean ShowGrid` : the Boolean value to specify whether to display or hide gridlines in Chart

`boolean BackgroundColorApply` : the Boolean value to specify whether the background color is applied to Chart. A false means 'transparent' background and a true will change the ColorBackgground to the background color.

`boolean ZoomButtonsVisible` : the Boolean value to specify whether the zoom interface is displayed

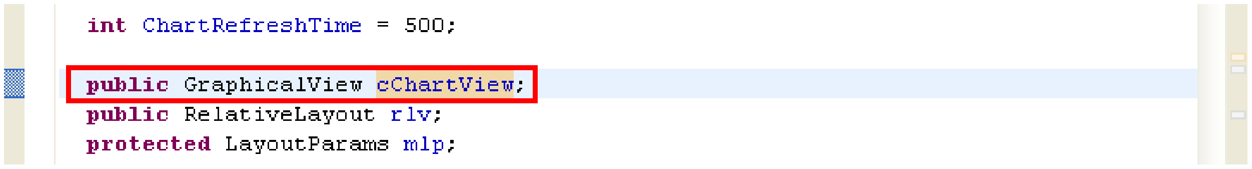
`boolean ShowLegend` : the Boolean value to specify whether Chart's text description at the bottom left corner is displayed

`boolean ShowAxes` : the Boolean value to specify whether Chart's axes are displayed

`boolean ShowLabels` : the Boolean value to specify whether the label names of Chart's axes are displayed

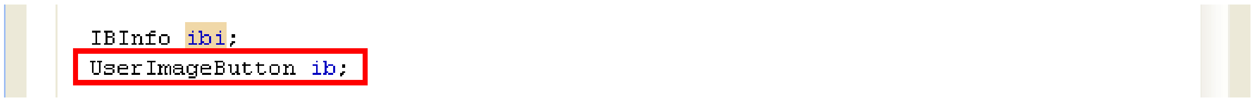
`float ChartSmoothness` : Chart's line smoothness

7. Add a `GraphicalView` object named `cChartView`.



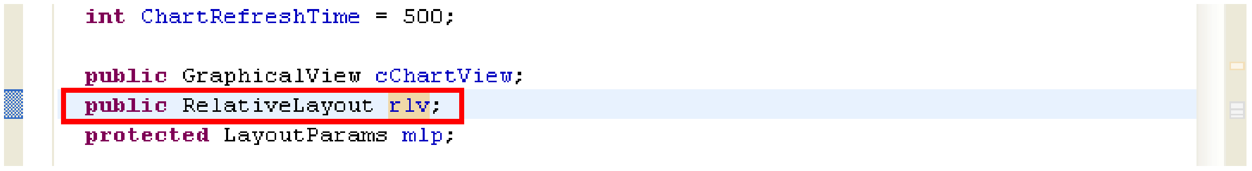
```
int ChartRefreshTime = 500;

public GraphicalView cChartView;
public RelativeLayout rlv;
protected LayoutParams mlp;
```

8. Add a `UserImageButton` object named `ib`.



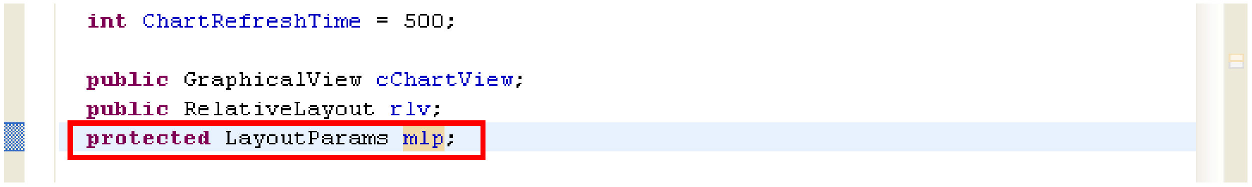
```
IBInfo ibi;
UserImageButton ib;
```

9. Add a new relative layout.



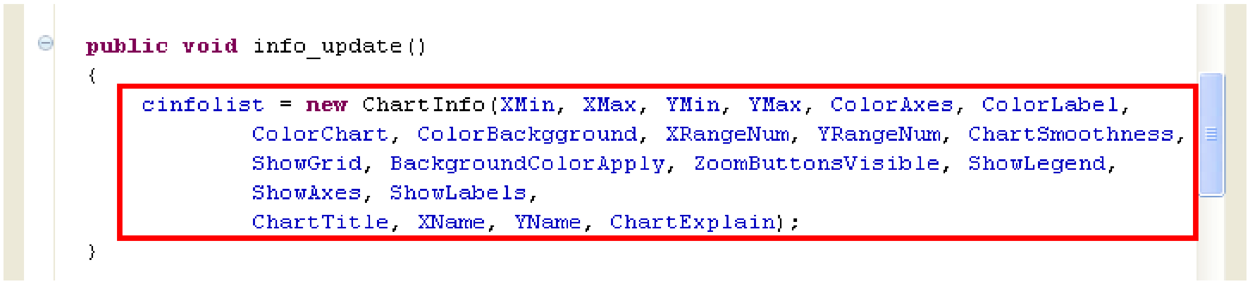
```
int ChartRefreshTime = 500;

public GraphicalView cChartView;
public RelativeLayout rlv;
protected LayoutParams mlp;
```

10. Add a `LayoutParams` named `mlp`.



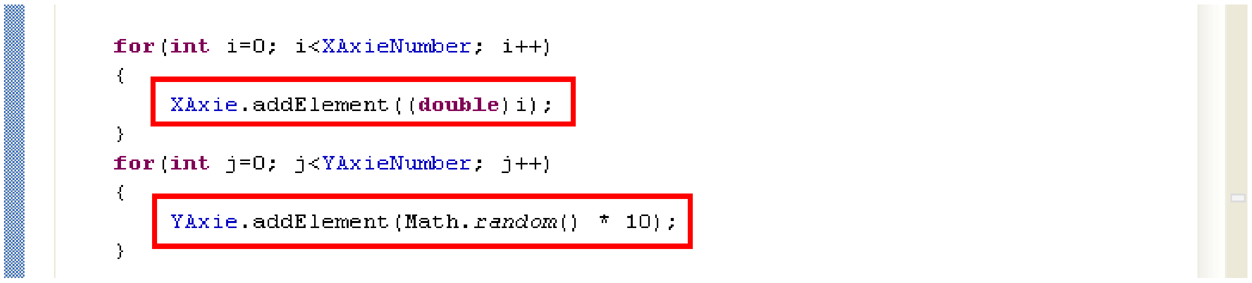
```
int ChartRefreshTime = 500;

public GraphicalView cChartView;
public RelativeLayout rlv;
protected LayoutParams mlp;
```

11. In the function `ChartInfo()`, assign all the parameters that have already established in Step 6 to `cinfolist`.



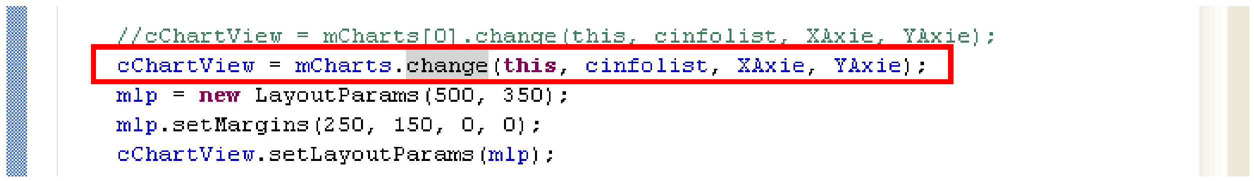
```
public void info_update()
{
    cinfolist = new ChartInfo(XMin, XMax, YMin, YMax, ColorAxes, ColorLabel,
            ColorChart, ColorBackground, XRangeNum, YRangeNum, ChartSmoothness,
            ShowGrid, BackgroundColorApply, ZoomButtonsVisible, ShowLegend,
            ShowAxes, ShowLabels,
            ChartTitle, XName, YName, ChartExplain);
}
```

12. For the range of values in X-axis and Y-axis that have already defined in Step 6, add a double data type to `XAixe` and `YAxis` respectively.



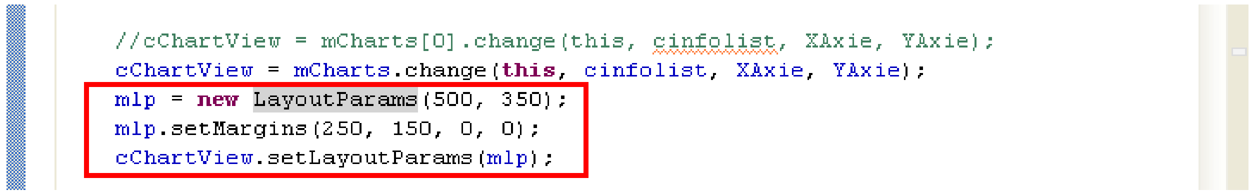
```
for(int i=0; i<XAxieNumber; i++)
{
    XAxie.addElement((double)i);
}
for(int j=0; j<YAxieNumber; j++)
{
    YAxie.addElement(Math.random() * 10);
}
```

13. In the function `change()`, enter the parameter value for `mCharts`.



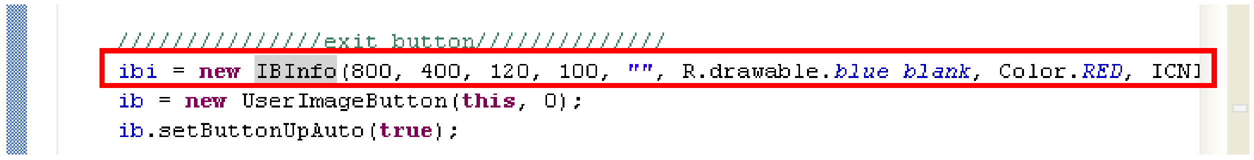
```
//cChartView = mCharts[0].change(this, cinfolist, XAxie, YAxie);
cChartView = mCharts.change(this, cinfolist, XAxie, YAxie);
mlp = new LayoutParams(500, 350);
mlp.setMargins(250, 150, 0, 0);
cChartView.setLayoutParams(mlp);
```

14. Assign a new `LayoutParams()` parameter to `mlp`. Then assign two values to `LayoutParams()` to represent the width and height of the layout. Use `setMargins()` to position layout where the first two parameters represent the x-axis coordinate and the y-axis coordinate.
Assign the height and width of `LayoutParams()` to `mlp` and add `mlp` to `setLayoutParams()` before `cChartView`.



```
//cChartView = mCharts[0].change(this, cinfolist, XAxie, YAxie);
cChartView = mCharts.change(this, cinfolist, XAxie, YAxie);
mlp = new LayoutParams(500, 350);
mlp.setMargins(250, 150, 0, 0);
cChartView.setLayoutParams(mlp);
```

15. You can add the parameters of the image button to `ibi`.

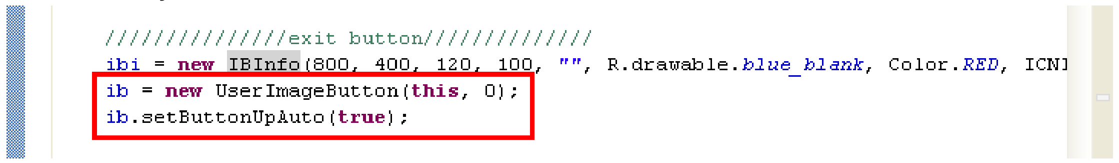


```
///////////////exit button//////////////
ibi = new IBInfo(800, 400, 120, 100, "", R.drawable.blue_blank, Color.RED, ICN]
ib = new UserImageButton(this, 0);
ib.setButtonUpAuto(true);
```

---Parameter definition---

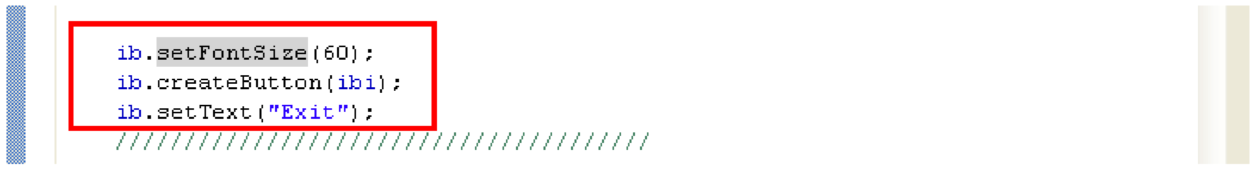
1st (`int x`) : the x-coordinate of the image button
2nd (`int y`) : the y-coordinate of the image button
3rd (`int w`) : the width of the image button
4th (`int h`) : the height of the image button
5th (`String img`) : to use the image name as the image button
6th (`int did`) : the numerical value in the image button
7th (`int textcolor`) : the text color of the image button
8th (`FONT_ID fontid`) : the font type of the image button
9th (`String text`) : the text content of the image button

16. The parameters of `UserImageButton()` are the id serial numbers for Activity and image button. You may use `setButtonUpAuto()` to decide whether image button will be automatically turned on.



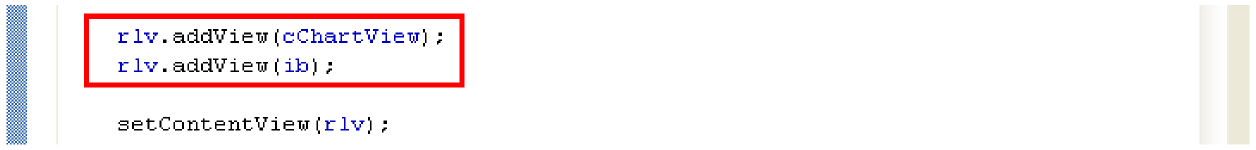
```
///////////////exit button//////////////
ibi = new IBInfo(800, 400, 120, 100, "", R.drawable.blue_blank, Color.RED, ICN]
ib = new UserImageButton(this, 0);
ib.setButtonUpAuto(true);
```

17. Use `setFontSize()` to specify the font size of the image button. `createButton()` will create a new image button based on the parameters of `IBInfo`. The `setText()` will specify the text content of the image button.



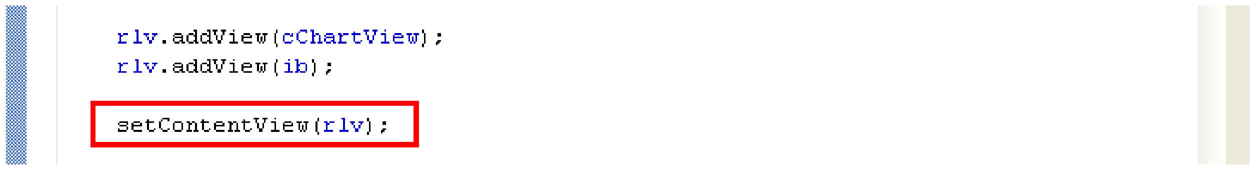
```
ib.setFontSize(60);
ib.createButton(ibi);
ib.setText("Exit");
////////////////////////////////////////////////
```

18. In the `onCreate()` method, put the view objects of `cChartView` and `ib` inside the `addView()` function. Then add view objects to the already established `rlv` (relative layout).



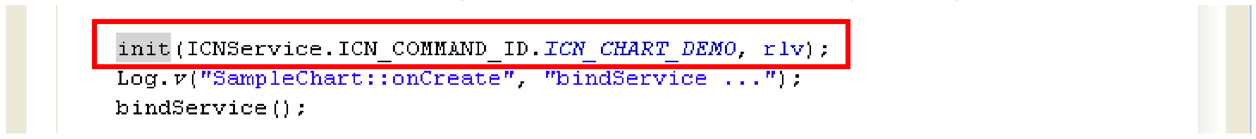
```
rlv.addView(cChartView);
rlv.addView(ib);

setContentView(rlv);
```

19. In the `onCreate()` method, add `setContentView(rlv)`.



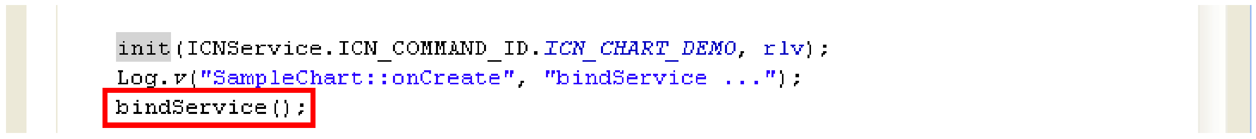
```
rlv.addView(cChartView);
rlv.addView(ib);

setContentView(rlv);
```

20. In the `onCreate()` method, add `init(parameter1, parameter2)`. Fill the Command ID of ICNService in parameter1 and the relative layout in parameter2.



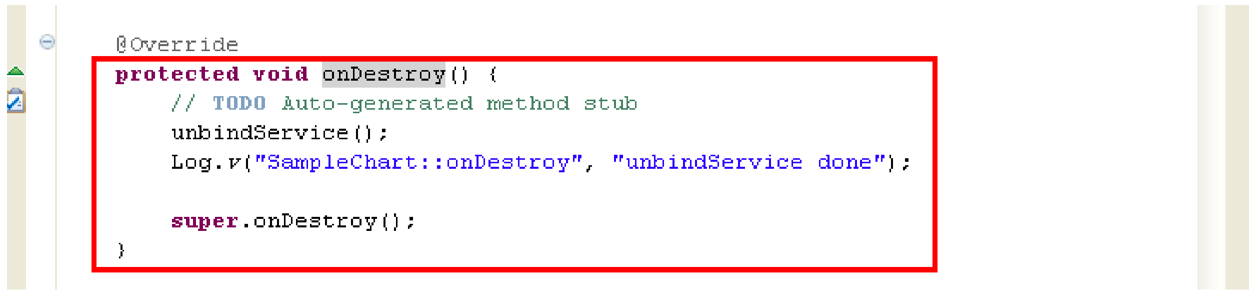
```
init(ICNService.ICN_COMMAND_ID.ICN_CHART_DEMO, rlv);
Log.v("SampleChart::onCreate", "bindService ...");
bindService();
```

21. In the `onCreate()` method, add `bindService()`.



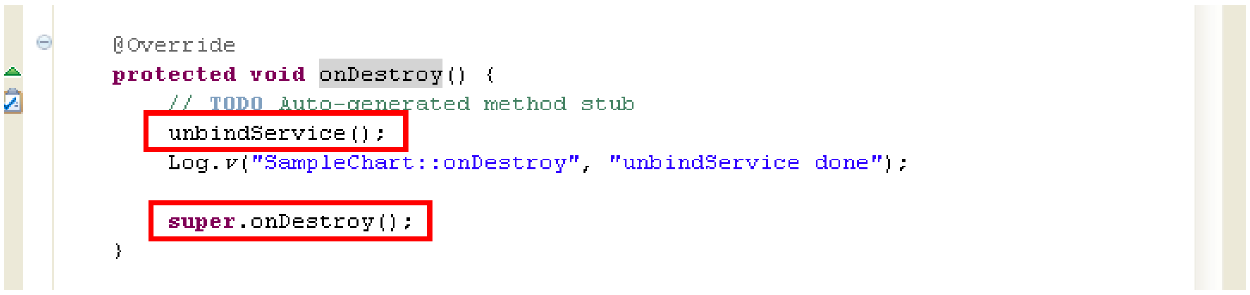
```
init(ICNService.ICN_COMMAND_ID.ICN_CHART_DEMO, rlv);
Log.v("SampleChart::onCreate", "bindService ...");
bindService();
```

22. Replace function name `SampleDigitalSignage()` with `onDestroy()`.



```
@Override
protected void onDestroy() {
    // TODO Auto-generated method stub
    unbindService();
    Log.v("SampleChart::onDestroy", "unbindService done");

    super.onDestroy();
}
```

23. In the `onDestroy()`, add `unbindService()` and `super.onDestroy()`.

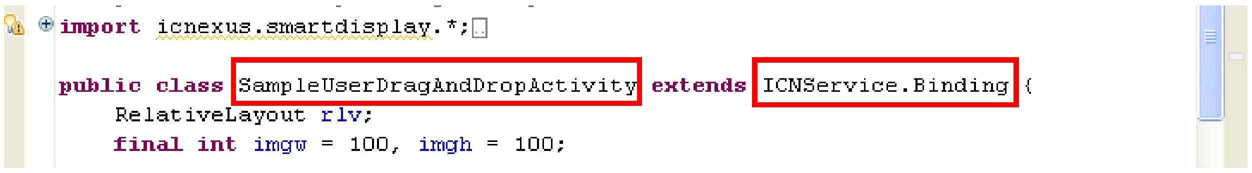


```
@Override
protected void onDestroy() {
    // TODO Auto-generated method stub
    unbindService();
    Log.v("SampleChart::onDestroy", "unbindService done");

    super.onDestroy();
}
```

Sample3: User Drag and Drop

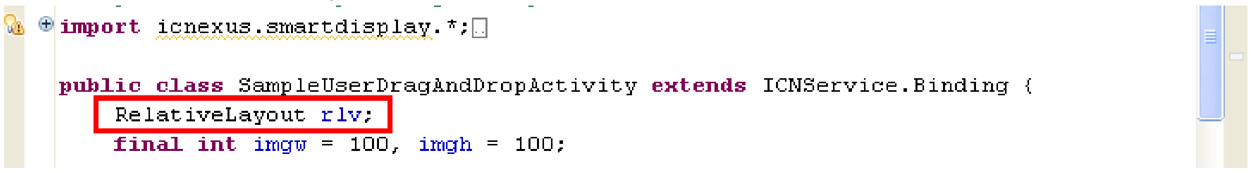
This section will introduce you how to add applications in the `UserDragAndDrop` Project.

1. Create a new project named `SampleUserDragAndDropActivity` and extends it to `ICNService.Binding`



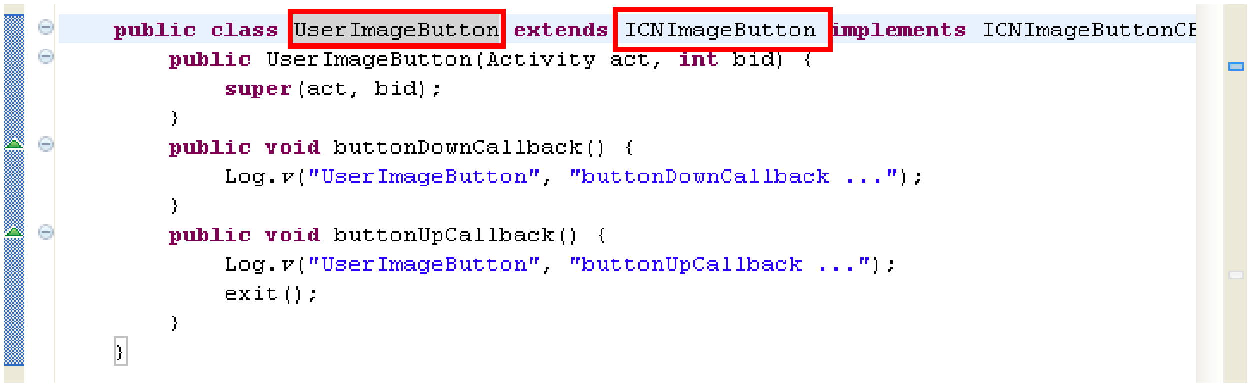
```
import icnexus.smartdisplay.*;

public class SampleUserDragAndDropActivity extends ICNService.Binding {
    RelativeLayout rlv;
    final int imgw = 100, imgh = 100;
```

2. Add a new relative layout



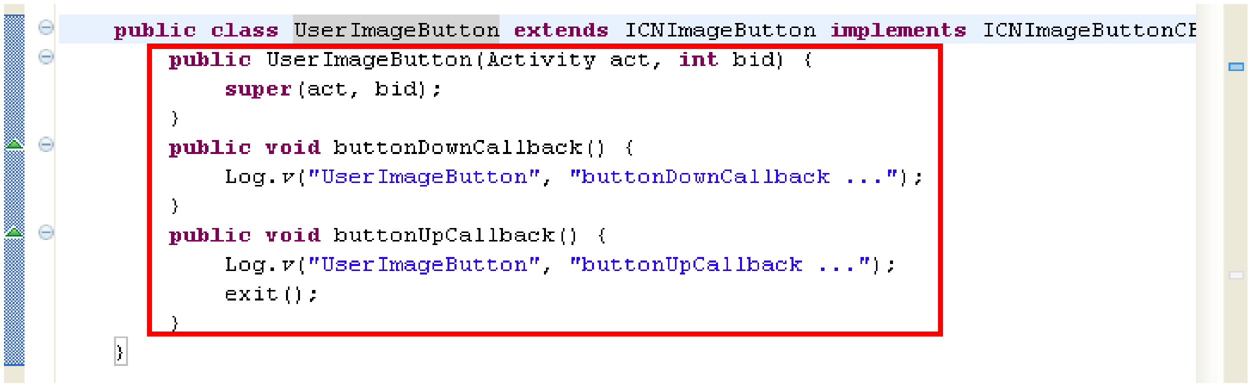
```
import icnexus.smartdisplay.*;

public class SampleUserDragAndDropActivity extends ICNService.Binding {
    RelativeLayout rlv;
    final int imgw = 100, imgh = 100;
```

3. Create a class named `UserImageButton` and extends it to `ICNImageButton`



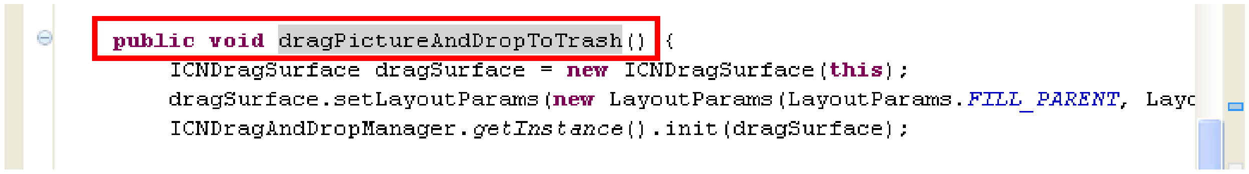
```
    public class UserImageButton extends ICNImageButton implements ICNImageButtonCE
        public UserImageButton(Activity act, int bid) {
            super(act, bid);
        }
        public void buttonDownCallback() {
            Log.v("UserImageButton", "buttonDownCallback ...");
        }
        public void buttonUpCallback() {
            Log.v("UserImageButton", "buttonUpCallback ...");
            exit();
        }
    }
```

4. In Class `UserImageButton`, create a `UserImageButton` function with a `super()`, and create a `buttonDownCallback` function. In Class `UserImageButton`，add `buttonUpcallback` function and `exit()`.



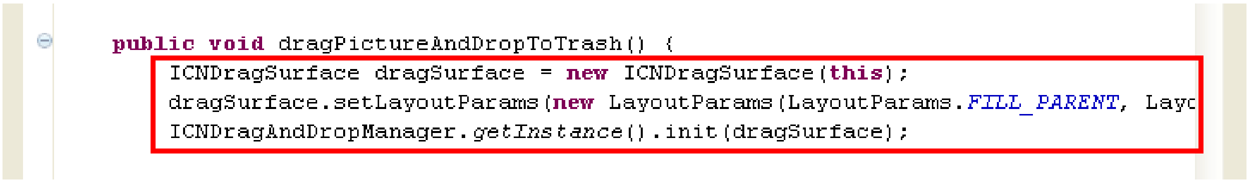
```
    public class UserImageButton extends ICNImageButton implements ICNImageButtonCE
        public UserImageButton(Activity act, int bid) {
            super(act, bid);
        }
        public void buttonDownCallback() {
            Log.v("UserImageButton", "buttonDownCallback ...");
        }
        public void buttonUpCallback() {
            Log.v("UserImageButton", "buttonUpCallback ...");
            exit();
        }
    }
```

5. Create a new function named `dragPictureAndDropToTrash`. This will initialize a draggable item, a dragged zone and `ICNDragSurface`.



```
    public void dragPictureAndDropToTrash() {
        ICNDragSurface dragSurface = new ICNDragSurface(this);
        dragSurface.setLayoutParams(new LayoutParams(LayoutParams.FILL_PARENT, Layo
        ICNDragAndDropManager.getInstance().init(dragSurface);
```

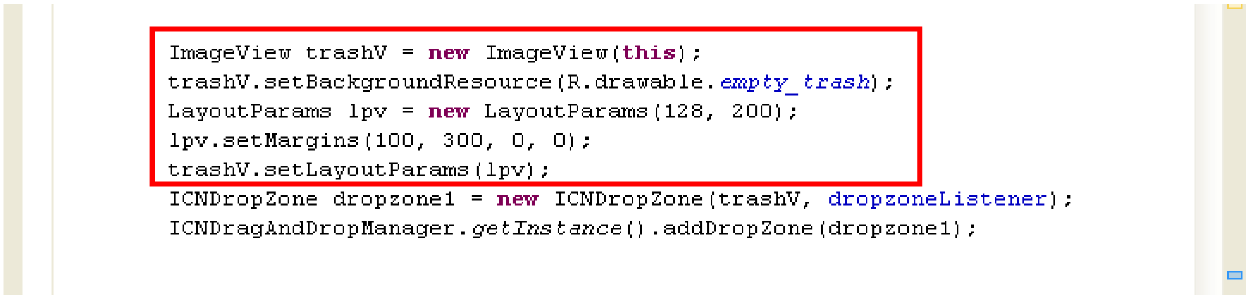
6. In the `dragPictureAndDropToTrash` function, add a `dragSurface` next to `ICNDragSurface` and give it a property.



```
    public void dragPictureAndDropToTrash() {
        ICNDragSurface dragSurface = new ICNDragSurface(this);
        dragSurface.setLayoutParams(new LayoutParams(LayoutParams.FILL_PARENT, Layo
        ICNDragAndDropManager.getInstance().init(dragSurface);
```

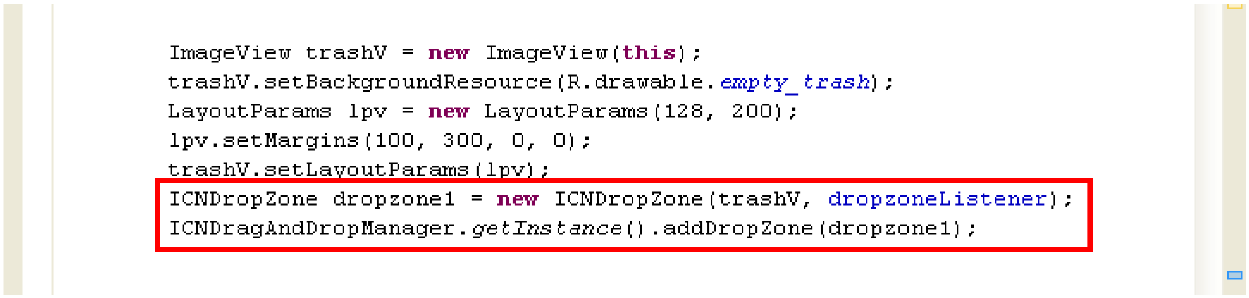
7. In the `dragPictureAndDropToTrash` function:

7.1. Add a new trash icon for ImageView named `trashV`. Remember to give it a location, height and width values.



```
        ImageView trashV = new ImageView(this);
        trashV.setBackgroundResource(R.drawable.empty_trash);
        LayoutParams lpv = new LayoutParams(128, 200);
        lpv.setMargins(100, 300, 0, 0);
        trashV.setLayoutParams(lpv);
        ICNDropZone dropzone1 = new ICNDropZone(trashV, dropzoneListener);
        ICNDragAndDropManager.getInstance().addDropZone(dropzone1);
```

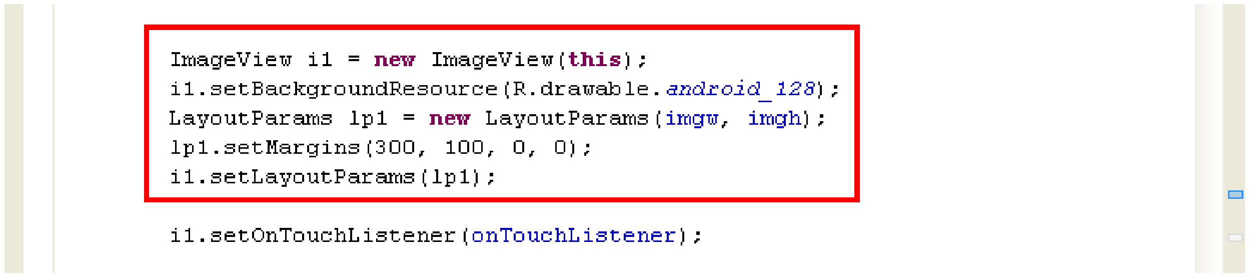
7.2. Add an `ICNDropArea` named `dropzone1`. Make the region of `trashV` as the place where one can drops an item.



```
        ImageView trashV = new ImageView(this);
        trashV.setBackgroundResource(R.drawable.empty_trash);
        LayoutParams lpv = new LayoutParams(128, 200);
        lpv.setMargins(100, 300, 0, 0);
        trashV.setLayoutParams(lpv);
        ICNDropZone dropzone1 = new ICNDropZone(trashV, dropzoneListener);
        ICNDragAndDropManager.getInstance().addDropZone(dropzone1);
```

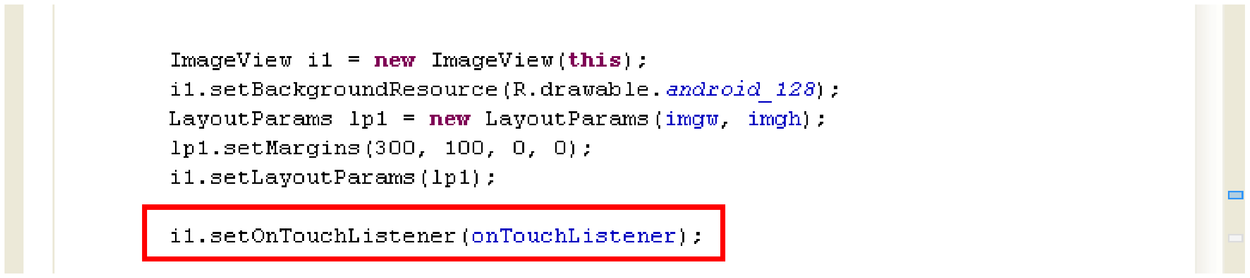
8. In the `dragPictureAndDropToTrash` function:

8.1. Add a draggable object: ImageView and named it `i1`. Specify the source of the image, its height, width and position.



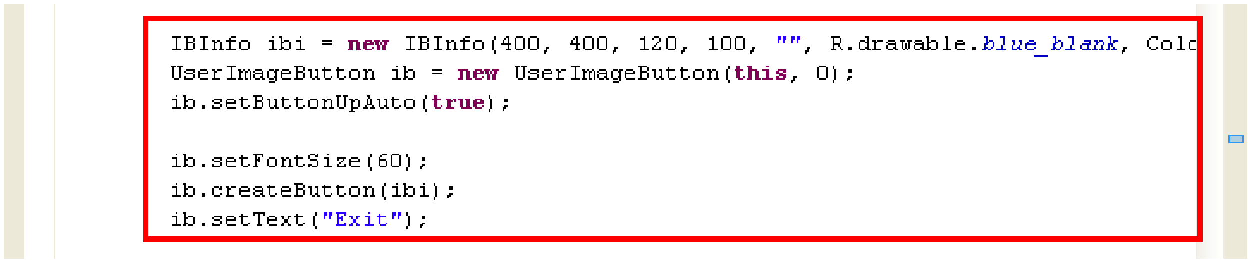
```
ImageView i1 = new ImageView(this);
i1.setBackgroundResource(R.drawable.android_128);
LayoutParams lp1 = new LayoutParams(imgw, imgh);
lp1.setMargins(300, 100, 0, 0);
i1.setLayoutParams(lp1);

i1.setOnTouchListener(onTouchListener);
```

8.2. Add a draggable object `i1` to TouchListener.



```
ImageView i1 = new ImageView(this);
i1.setBackgroundResource(R.drawable.android_128);
LayoutParams lp1 = new LayoutParams(imgw, imgh);
lp1.setMargins(300, 100, 0, 0);
i1.setLayoutParams(lp1);

i1.setOnTouchListener(onTouchListener);
```

9. In the `dragPictureAndDropToTrash` function, add an exit object: UserImageButton named `ib`. Then set the ButtonUpAuto as true.



```
IBInfo ibi = new IBInfo(400, 400, 120, 100, "", R.drawable.blue_blank, Colc
UserImageButton ib = new UserImageButton(this, 0);
ib.setButtonUpAuto(true);

ib.setFontSize(60);
ib.createButton(ibi);
ib.setText("Exit");
```

---Parameter definition---

1st (`int x`) : the x-coordinate of the image button

2nd (`int y`) : the y-coordinate of the image button

3rd (`int w`) : the width of the image button

4th (`int h`) : the height of the image button

5th (`String img`) : To use the image name as the image button

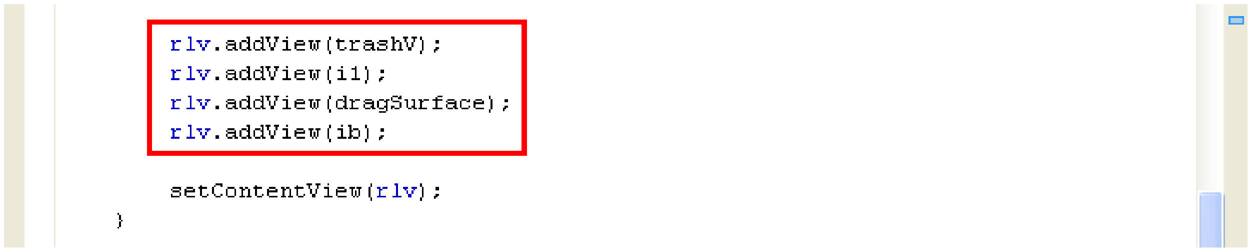
6th (`int did`) : the numerical value in the image button

7th (`int textcolor`) : the text color of the image button

8th (`FONT_ID fontid`) : the font type of the image button

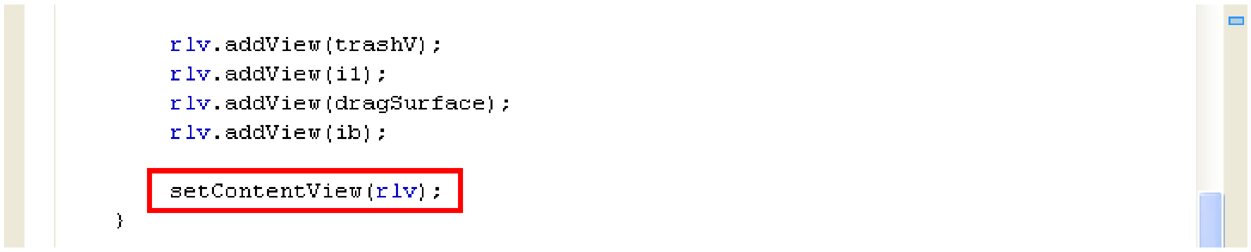
9th (`String text`) : the text content of the image button

10. In the `dragPictureAndDropToTrash` function, add a few view objects named `trashV`, `i1`, `dragsurface` and `ib` in the `addView` functionto the already established `rlv (`relative layout).



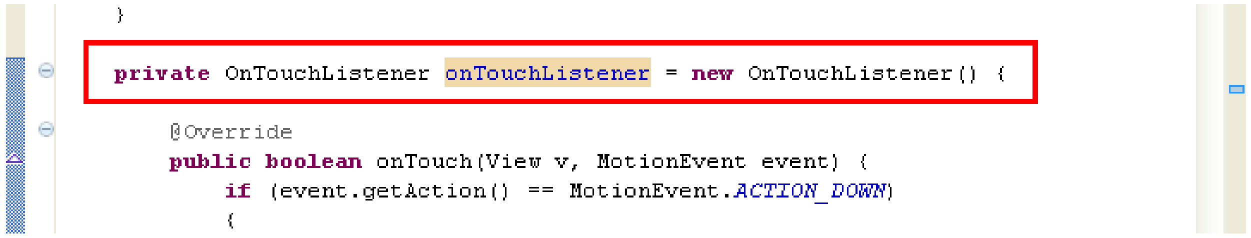
```
        rlv.addView(trashV);
        rlv.addView(i1);
        rlv.addView(dragSurface);
        rlv.addView(ib);

        setContentView(rlv);
    }
```

11. In the `dragPictureAndDropToTrash` function, add `setContentView(rlv)`



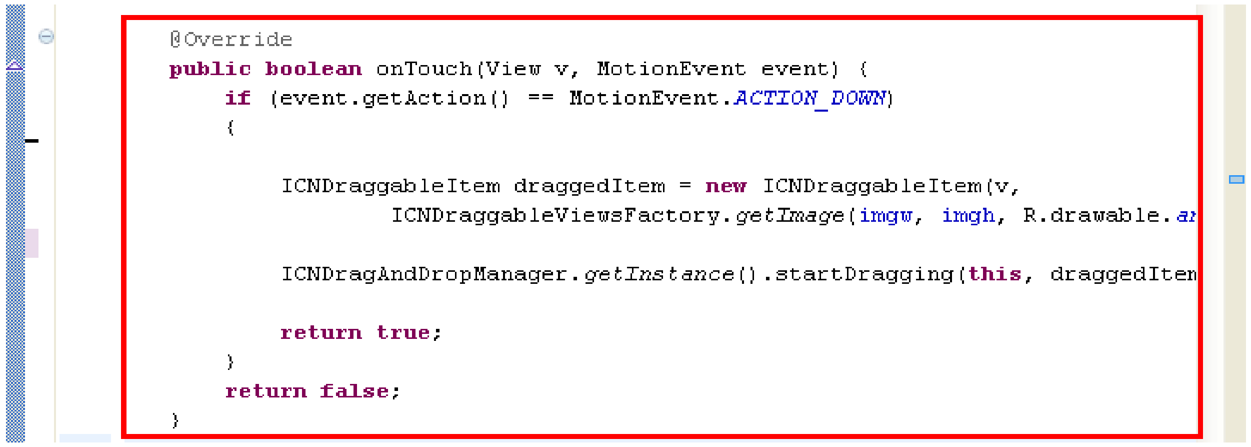
```
        rlv.addView(trashV);
        rlv.addView(i1);
        rlv.addView(dragSurface);
        rlv.addView(ib);

        setContentView(rlv);
    }
```

12. Add onTouchListener for **8.2** to use:



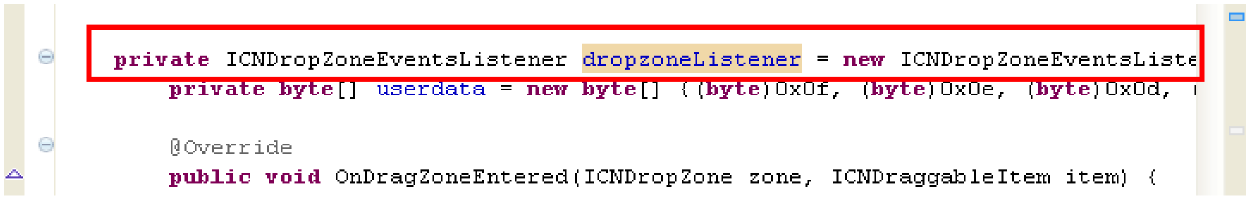
```
    }

    private OnTouchListener onTouchListener = new OnTouchListener() {

        @Override
        public boolean onTouch(View v, MotionEvent event) {
            if (event.getAction() == MotionEvent.ACTION_DOWN)
            {
```

13. In the onTouchListener, implement an draggedItem



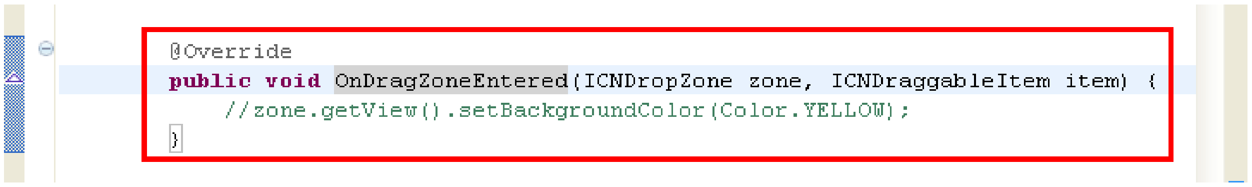
```
        @Override
        public boolean onTouch(View v, MotionEvent event) {
            if (event.getAction() == MotionEvent.ACTION_DOWN)
            {

                ICNDraggableItem draggedItem = new ICNDraggableItem(v,
                        ICNDraggableViewsFactory.getImage(imgw, imgh, R.drawable.ar

                ICNDragAndDropManager.getInstance().startDragging(this, draggedIten

                return true;
            }
            return false;
        }
```

14. Add a new event listener named dropzoneListener for **7.2** to use.



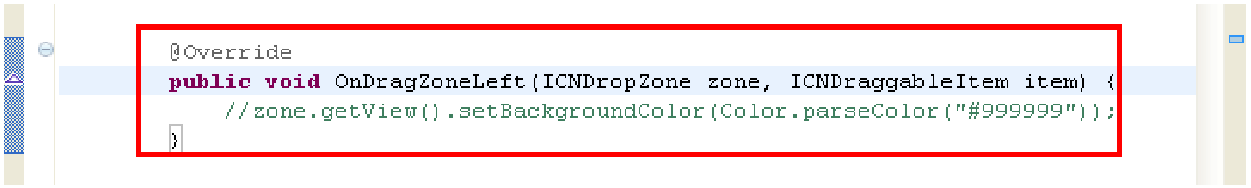
```
private ICNDropZoneEventsListener dropzoneListener = new ICNDropZoneEventsListe
    private byte[] userdata = new byte[] {(byte)0x0f, (byte)0x0e, (byte)0x0d,

    @Override
    public void OnDragZoneEntered(ICNDropZone zone, ICNDraggableItem item) {
```

15. In the dropzoneListener, implement `OnDragZoneEntered`



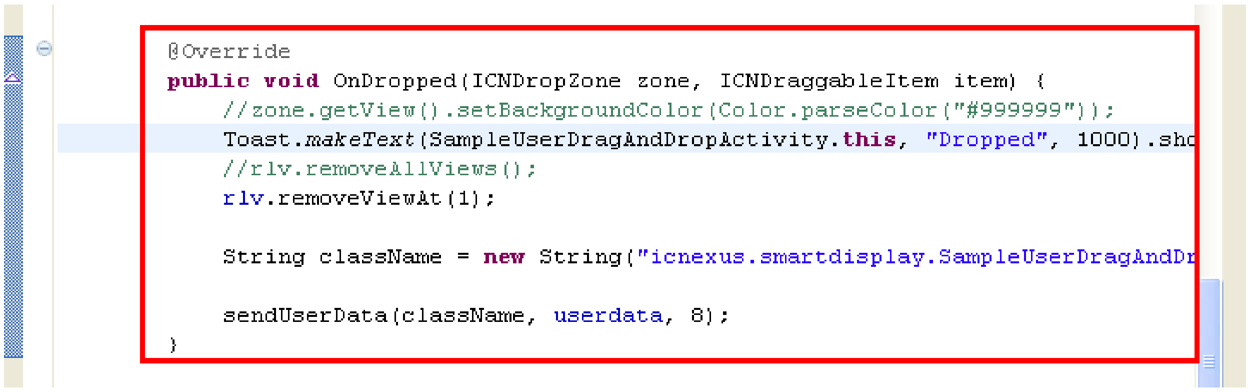
```
@Override
public void OnDragZoneEntered(ICNDropZone zone, ICNDraggableItem item) {
    //zone.getView().setBackgroundColor(Color.YELLOW);
}
```

16. In the dropzoneListener, implement `OnDragZoneLeft`



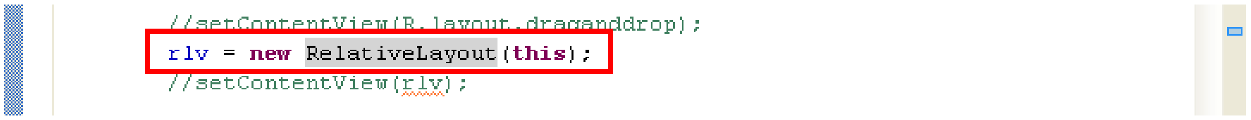
```
@Override
public void OnDragZoneLeft(ICNDropZone zone, ICNDraggableItem item) {
    //zone.getView().setBackgroundColor(Color.parseColor("#999999"));
}
```

17. In the dropzoneListener, implement `OnDropped`



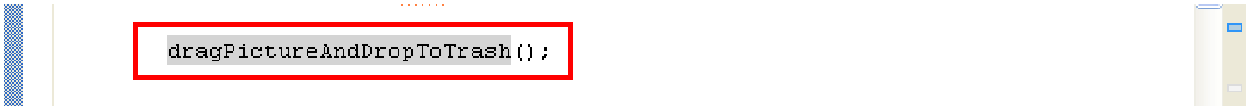
```
@Override
public void OnDropped(ICNDropZone zone, ICNDraggableItem item) {
    //zone.getView().setBackgroundColor(Color.parseColor("#999999"));
    Toast.makeText(SampleUserDragAndDropActivity.this, "Dropped", 1000).sho
    //rlv.removeAllViews();
    rlv.removeViewAt(1);

    String className = new String("icnexus.smartdisplay.SampleUserDragAndDr

    sendUserData(className, userdata, 8);
}
```

18. In the onCreate add `rlv = new RelativeLayout(this)`



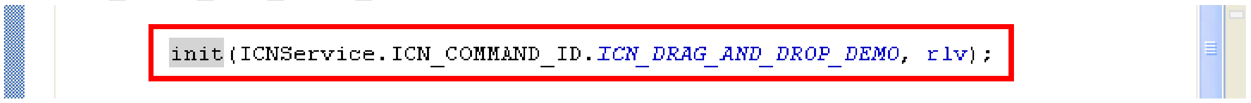
```
//setContentView(R.layout.draganddrop);
rlv = new RelativeLayout(this);
//setContentView(rlv);
```

19. In the onCreate, call the `dragPictureAndDropToTrash` function that has already established in step 5.



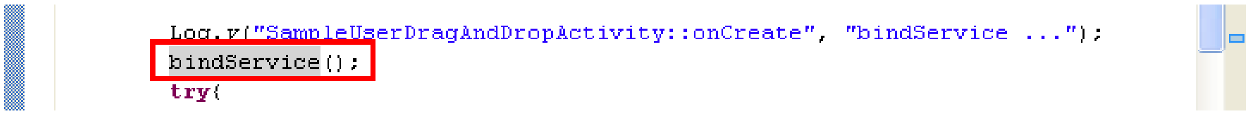
```
dragPictureAndDropToTrash();
```

20. In the onCreate, add `init(parameter1, parameter2)`. Fill the `rlv` (relative layout) in parameter2 and the Command ID of ICNService in parameter 1. The demo app as `ICN_DRAG_AND_DROP_DEMO`.



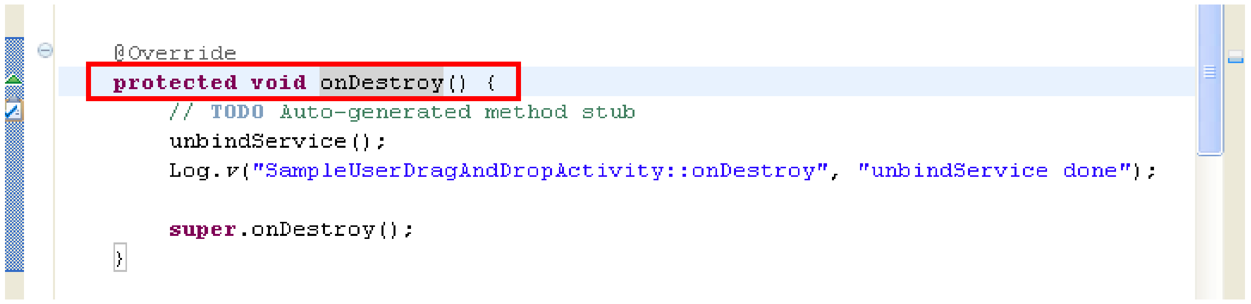
```
        init(ICNService.ICN_COMMAND_ID.ICN_DRAG_AND_DROP_DEMO, rlv);
```

21. In the onCreate, add `bindService()`.



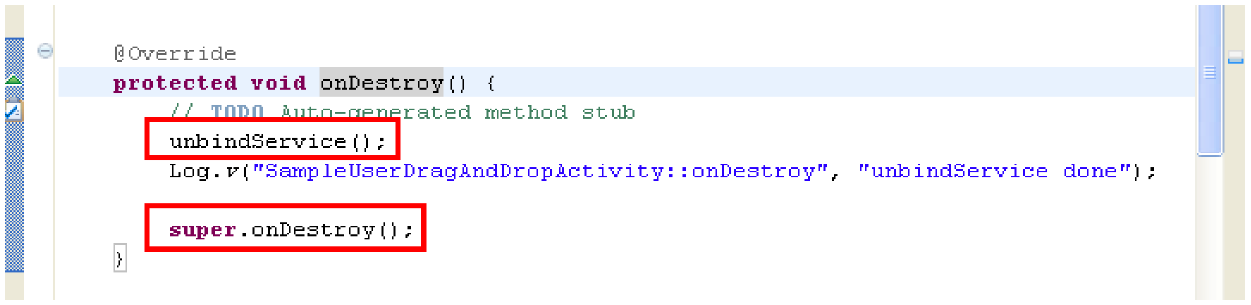
```
        Log.v("SampleUserDragAndDropActivity::onCreate", "bindService ...");
        bindService();
        try{
```

22. In the `SampleUserDragAndDropActivity`, override `onDestroy` function.



```
    @Override
    protected void onDestroy() {
        // TODO Auto-generated method stub
        unbindService();
        Log.v("SampleUserDragAndDropActivity::onDestroy", "unbindService done");

        super.onDestroy();
    }
```

23. In the onDestroy function, add `unbindService()` and `super.onDestroy()`.

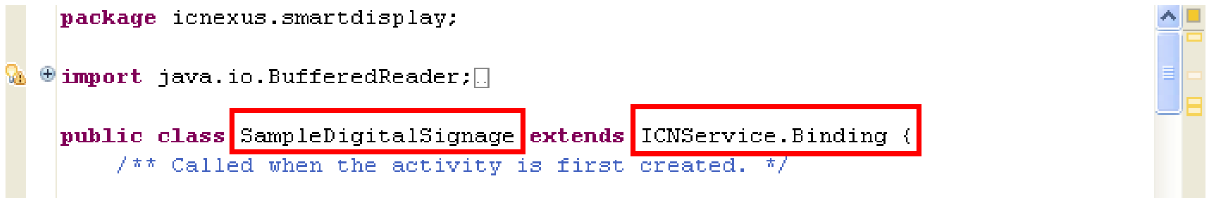


```
    @Override
    protected void onDestroy() {
        // TODO Auto-generated method stub
        unbindService();
        Log.v("SampleUserDragAndDropActivity::onDestroy", "unbindService done");

        super.onDestroy();
    }
```

Sample4: Digital Signage

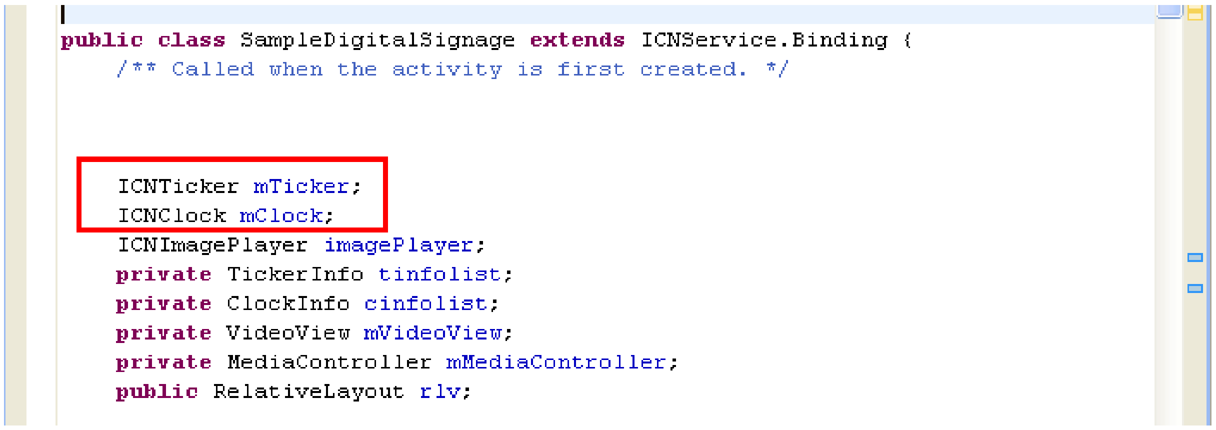
Here we will introduce how `Clock` and `Ticker` could be added to the `DigitalSignage` Project.

1. Declare a public class named `SampleDigitalSignage` that extends the `ICNService.Binding` class.



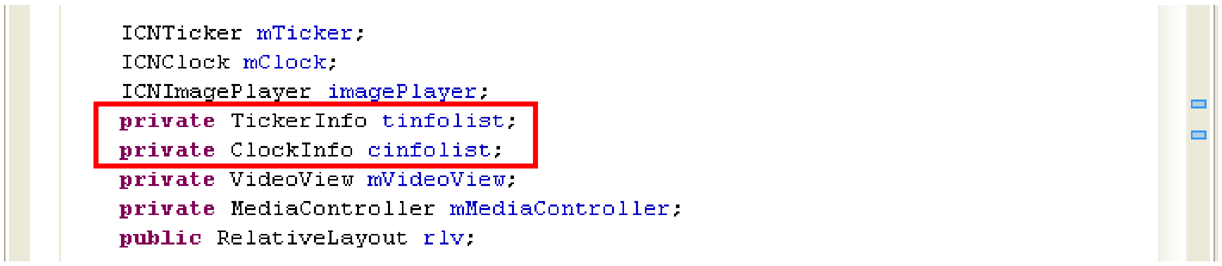
```
package icnexus.smartdisplay;

import java.io.BufferedReader;

public class SampleDigitalSignage extends ICNService.Binding {
    /** Called when the activity is first created. */
```

2. Apply `ICNTicker` and `ICNClock` to create two classes named `mTicker` and `mClock` respectively.



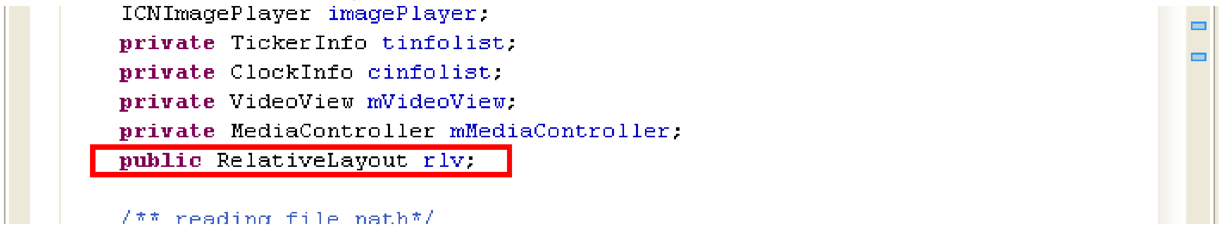
```
public class SampleDigitalSignage extends ICNService.Binding {
    /** Called when the activity is first created. */


    ICNTicker mTicker;
    ICNClock mClock;
    ICNImagePlayer imagePlayer;
    private TickerInfo tinfolist;
    private ClockInfo cinfolist;
    private VideoView mVideoView;
    private MediaController mMediaController;
    public RelativeLayout rlv;
```

3. Use `TickerInfo` and `ClockInfo` to create two parameter objects for Ticker and Clock. Let's name them as `tinfolist` and `cinfolist` respectively.



```
    ICNTicker mTicker;
    ICNClock mClock;
    ICNImagePlayer imagePlayer;
    private TickerInfo tinfolist;
    private ClockInfo cinfolist;
    private VideoView mVideoView;
    private MediaController mMediaController;
    public RelativeLayout rlv;
```

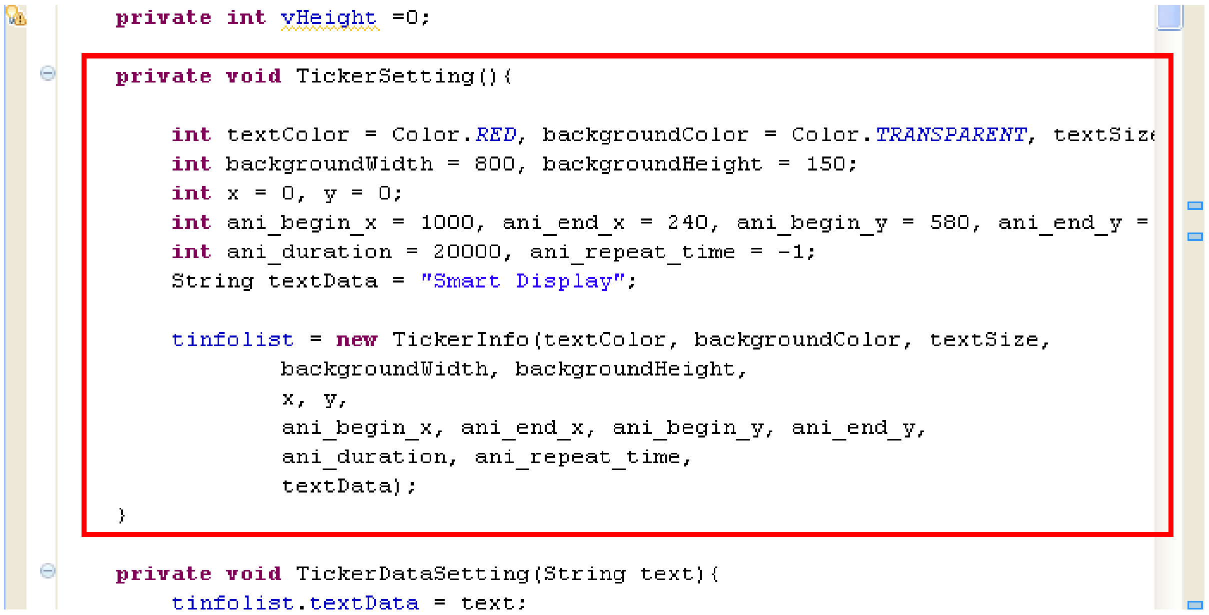
4. Let's create a relative layout.



```
    ICNImagePlayer imagePlayer;
    private TickerInfo tinfolist;
    private ClockInfo cinfolist;
    private VideoView mVideoView;
    private MediaController mMediaController;
    public RelativeLayout rlv;

    /** reading file path*/
```

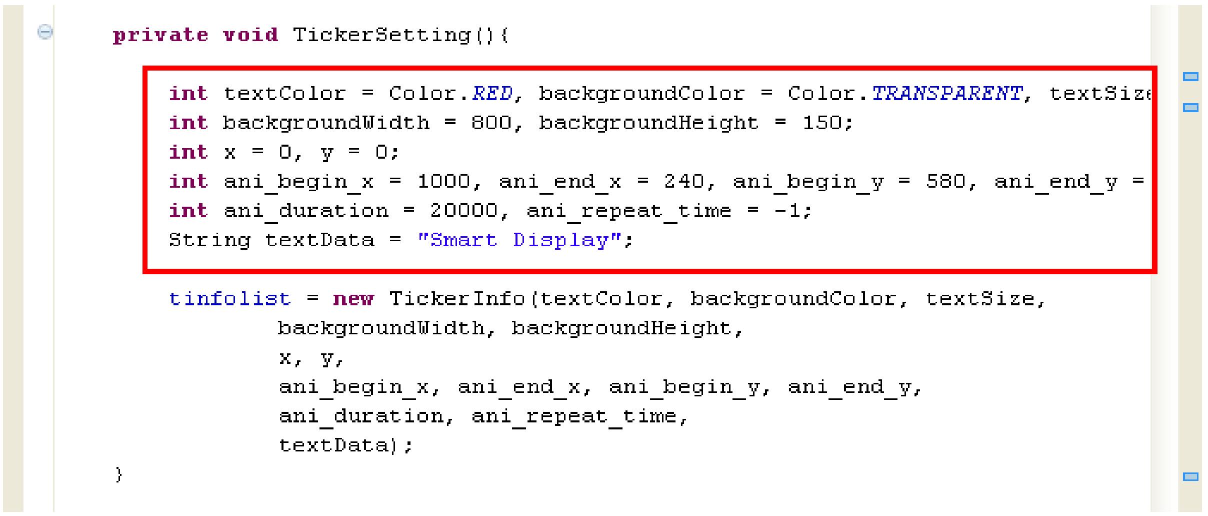
5.

5.1. Add a function parameter named `TickerSetting` for Ticker.



```
    private int vHeight =0;

    private void TickerSetting(){

        int textColor = Color.RED, backgroundColor = Color.TRANSPARENT, textSize
        int backgroundWidth = 800, backgroundHeight = 150;
        int x = 0, y = 0;
        int ani_begin_x = 1000, ani_end_x = 240, ani_begin_y = 580, ani_end_y =
        int ani_duration = 20000, ani_repeat_time = -1;
        String textData = "Smart Display";

        tinfolist = new TickerInfo(textColor, backgroundColor, textSize,
                backgroundWidth, backgroundHeight,
                x, y,
                ani_begin_x, ani_end_x, ani_begin_y, ani_end_y,
                ani_duration, ani_repeat_time,
                textData);
    }

    private void TickerDataSetting(String text){
        tinfolist.textData = text;
```

5.1.1. Set the variables.



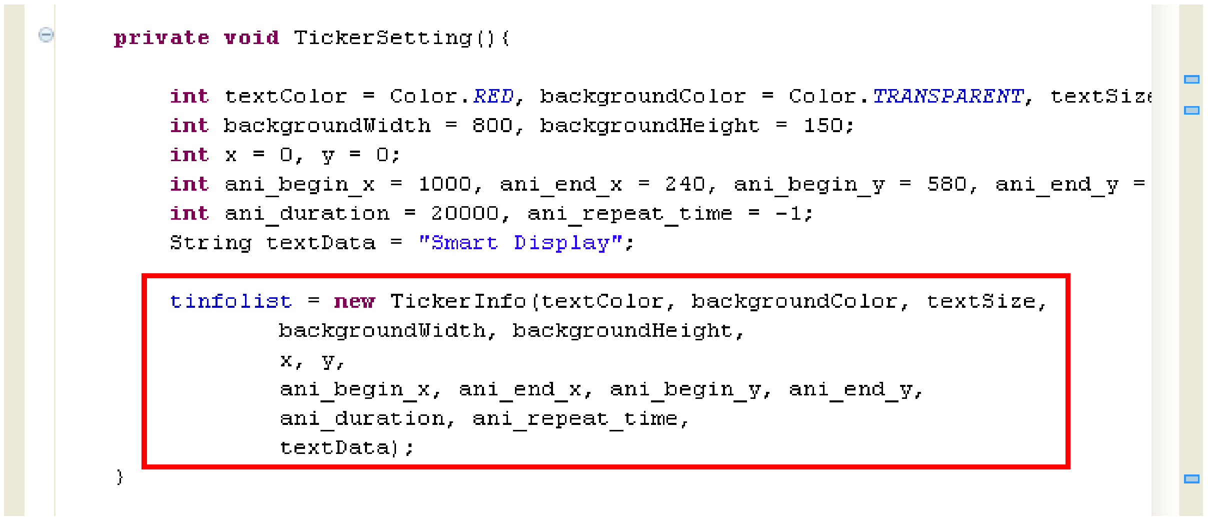
```
    private void TickerSetting(){

        int textColor = Color.RED, backgroundColor = Color.TRANSPARENT, textSize
        int backgroundWidth = 800, backgroundHeight = 150;
        int x = 0, y = 0;
        int ani_begin_x = 1000, ani_end_x = 240, ani_begin_y = 580, ani_end_y =
        int ani_duration = 20000, ani_repeat_time = -1;
        String textData = "Smart Display";

        tinfolist = new TickerInfo(textColor, backgroundColor, textSize,
                backgroundWidth, backgroundHeight,
                x, y,
                ani_begin_x, ani_end_x, ani_begin_y, ani_end_y,
                ani_duration, ani_repeat_time,
                textData);
    }
```

--- Variable definition---

- `int textColor` : Ticker's text color
- `int backgroundColor` : Ticket's background color
- `int textSize` : Ticker's font size
- `int backgroundWidth` : the width of Ticker's background
- `int backgroundHeight` : the height of Ticker's background
- `int x` : Ticker's x coordinate of the upper-left corner
- `int y` : Ticker's y coordinate of the upper-left corner
- `int ani_begin_x` : Ticker's x coordinate of the starting point at the upper-left corner

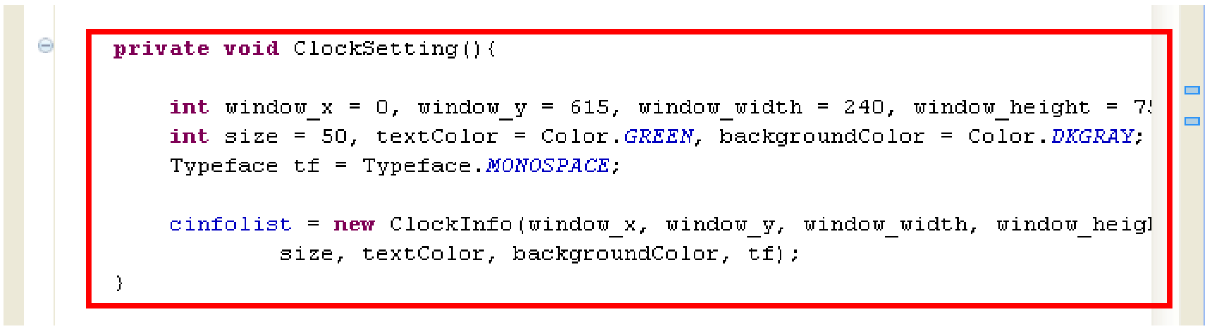
`int ani_end_x` : Ticker's x coordinate of the end point at the upper-left corner
`int ani_begin_y` : Ticker's y coordinate of the starting point at the upper-left corner
`int ani_end_y` : Ticker's y coordinate of the end point at the upper-left corner
`int ani_duration` : the time Ticker takes to complete the whole path (ms)
`int ani_repeat_time` : the number of times Ticker has run. -1 implies infinity
`String textData` : the text content of Ticker

5.1.2. Add parameters to `tinfolist`.



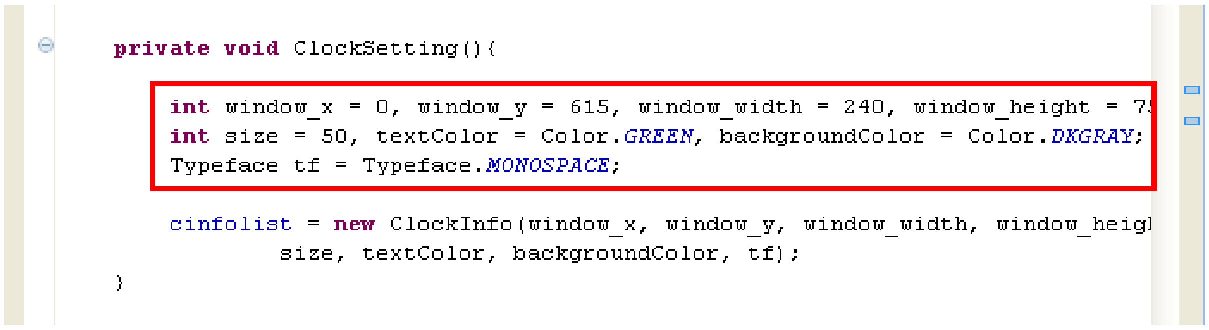
```
private void TickerSetting(){

    int textColor = Color.RED, backgroundColor = Color.TRANSPARENT, textSize
    int backgroundWidth = 800, backgroundHeight = 150;
    int x = 0, y = 0;
    int ani_begin_x = 1000, ani_end_x = 240, ani_begin_y = 580, ani_end_y =
    int ani_duration = 20000, ani_repeat_time = -1;
    String textData = "Smart Display";

    tinfolist = new TickerInfo(textColor, backgroundColor, textSize,
            backgroundWidth, backgroundHeight,
            x, y,
            ani_begin_x, ani_end_x, ani_begin_y, ani_end_y,
            ani_duration, ani_repeat_time,
            textData);
}
```

5.2. Add a new parameter for Clock named `ClockSetting`.



```
private void ClockSetting(){

    int window_x = 0, window_y = 615, window_width = 240, window_height = 7
    int size = 50, textColor = Color.GREEN, backgroundColor = Color.DKGRAY;
    Typeface tf = Typeface.MONOSPACE;

    cinfolist = new ClockInfo(window_x, window_y, window_width, window_heig
            size, textColor, backgroundColor, tf);
}
```

5.2.1. Set the variables.



```
private void ClockSetting(){

    int window_x = 0, window_y = 615, window_width = 240, window_height = 7
    int size = 50, textColor = Color.GREEN, backgroundColor = Color.DKGRAY;
    Typeface tf = Typeface.MONOSPACE;

    cinfolist = new ClockInfo(window_x, window_y, window_width, window_heig
            size, textColor, backgroundColor, tf);
}
```

--- Variable definition---

`int window_x` : Clock's x coordinate of the upper left corner of the background

`int window_y` : Clock's y coordinate of the upper left corner of the background

`int window_width` : the width of Clock's background

`int window_height` : the height of Clock's background

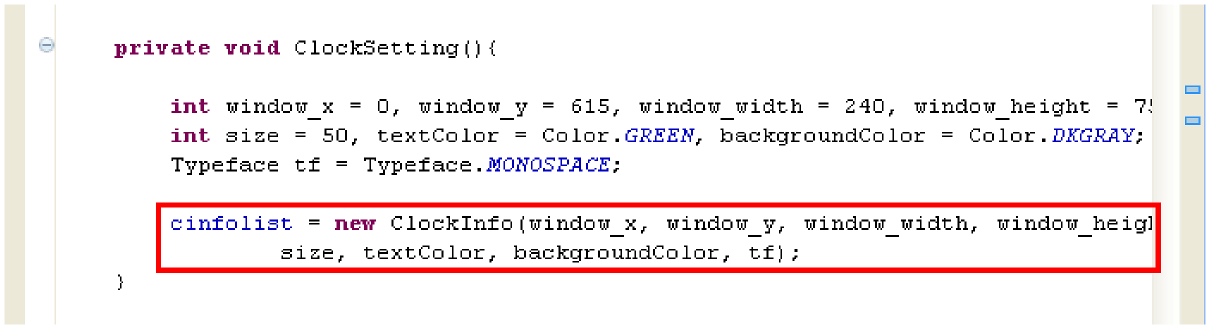
`int size` : Clock's font size

`int textColor` : Clock's font color

`int backgroundColor` : Clock's background color

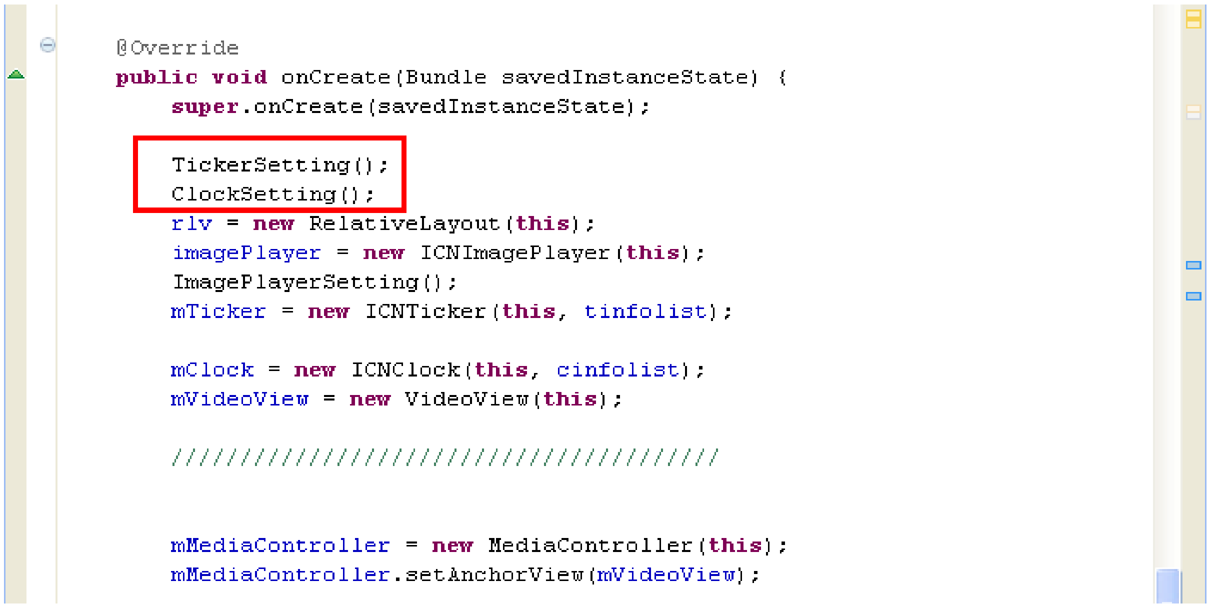
`Typeface tf` : Clock's text font type

5.2.2. Adding parameters to `cinfolist`.



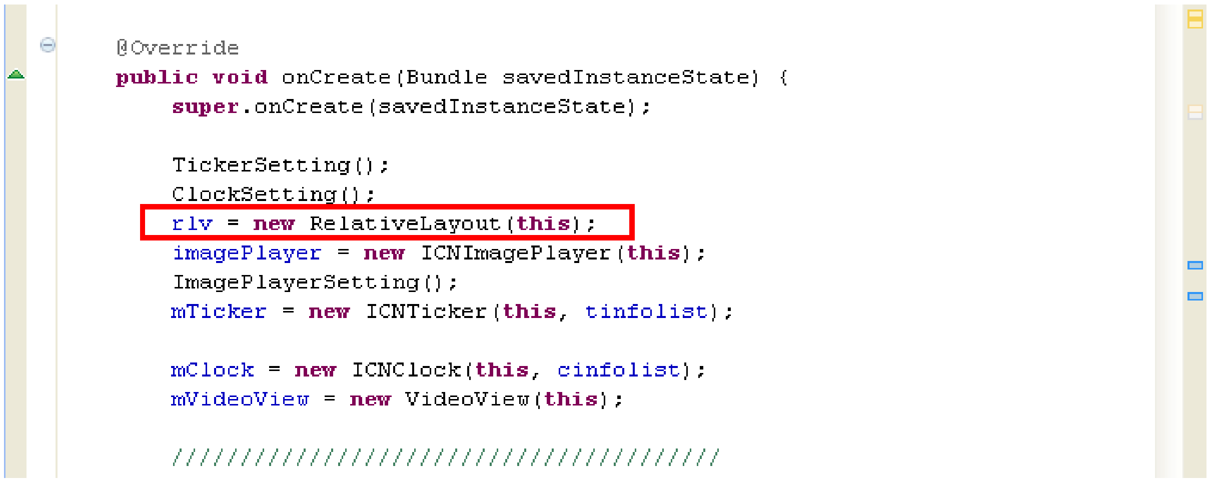
```
private void ClockSetting(){

    int window_x = 0, window_y = 615, window_width = 240, window_height = 7
    int size = 50, textColor = Color.GREEN, backgroundColor = Color.DKGRAY;
    Typeface tf = Typeface.MONOSPACE;

    cinfolist = new ClockInfo(window_x, window_y, window_width, window_heig
            size, textColor, backgroundColor, tf);
}
```

6. In the onCreate method, call function `TickerSetting` and `ClockSetting`.



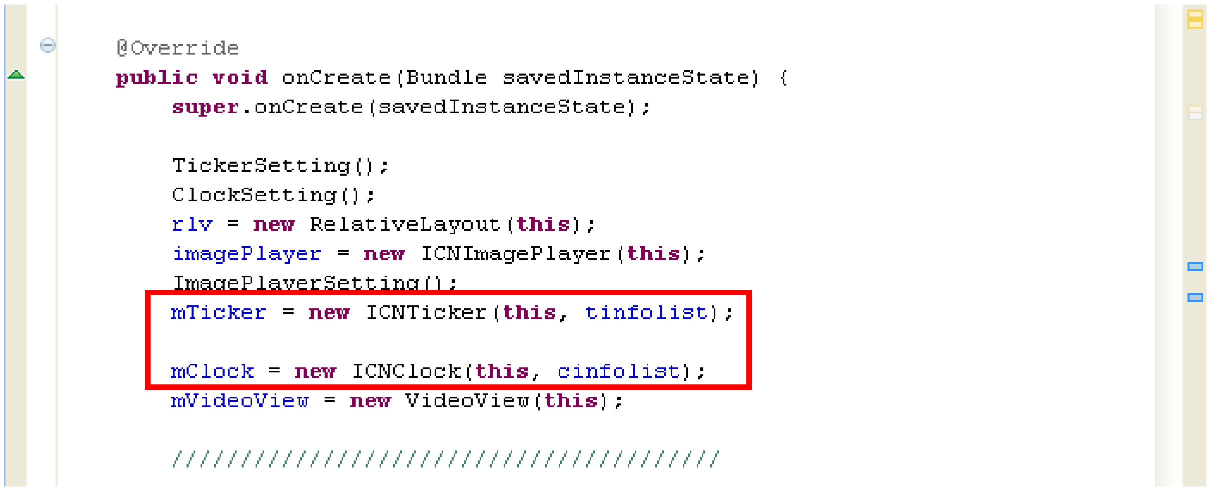
```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    TickerSetting();
    ClockSetting();
    rlv = new RelativeLayout(this);
    imagePlayer = new ICNImagePlayer(this);
    ImagePlayerSetting();
    mTicker = new ICNTicker(this, tinfolist);

    mClock = new ICNClock(this, cinfolist);
    mVideoView = new VideoView(this);

    ////////////////////////////////////////////


    mMediaController = new MediaController(this);
    mMediaController.setAnchorView(mVideoView);
```

7. In the onCreate method, change the variable of relative layout to the current context.



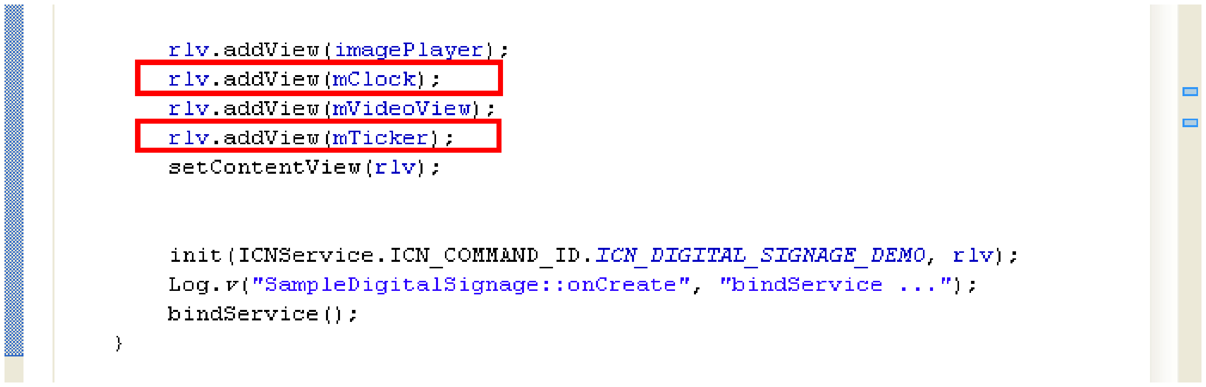
```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    TickerSetting();
    ClockSetting();
    rlv = new RelativeLayout(this);
    imagePlayer = new ICNImagePlayer(this);
    ImagePlayerSetting();
    mTicker = new ICNTicker(this, tinfolist);

    mClock = new ICNClock(this, cinfolist);
    mVideoView = new VideoView(this);

    ///////////////////////////////////////////
```

8. Create two new objects named `ICTTicket` and `ICTClock` respectively and then initialize them with the information in their parameters.



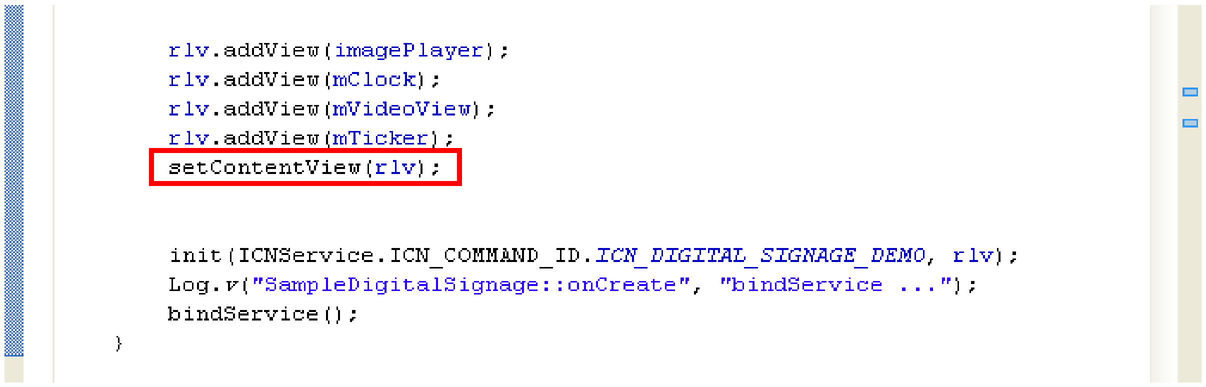
```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    TickerSetting();
    ClockSetting();
    rlv = new RelativeLayout(this);
    imagePlayer = new ICNImagePlayer(this);
    ImagePlayerSetting();
    mTicker = new ICNTicker(this, tinfolist);

    mClock = new ICNClock(this, cinfolist);
    mVideoView = new VideoView(this);

    ///////////////////////////////////////////
```

9. In the onCreate method, use `rlv` (relative layout) object reference to invoke the `addView` method. Then you apply the `mClock` and `mTicket` arguments to the addView method.



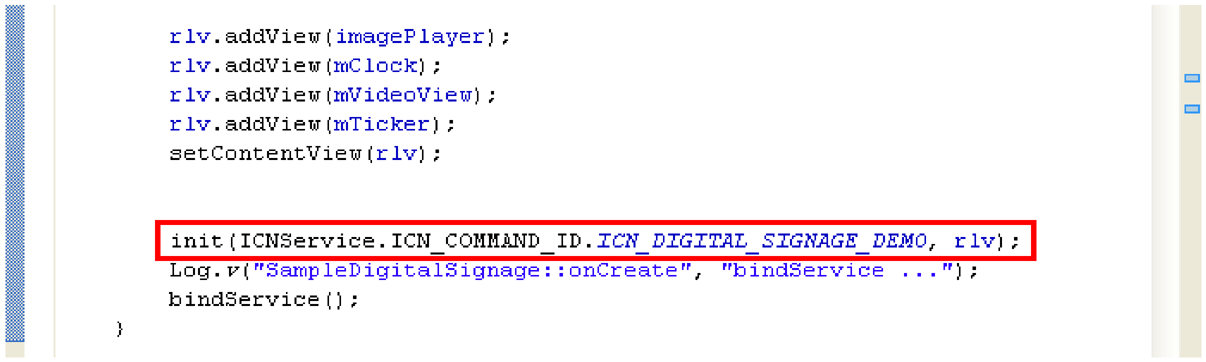
```
    rlv.addView(imagePlayer);
    rlv.addView(mClock);
    rlv.addView(mVideoView);
    rlv.addView(mTicker);
    setContentView(rlv);


    init(ICNService.ICN_COMMAND_ID.ICN_DIGITAL_SIGNAGE_DEMO, rlv);
    Log.v("SampleDigitalSignage::onCreate", "bindService ...");
    bindService();
}
```

10. In the onCreate method, add `setContentView` to the relative layout object reference.



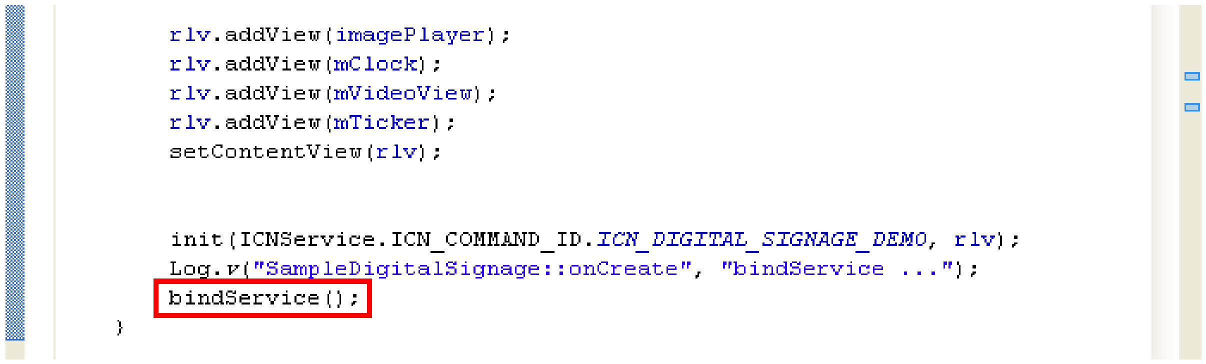
```
        rlv.addView(imagePlayer);
        rlv.addView(mClock);
        rlv.addView(mVideoView);
        rlv.addView(mTicker);
        setContentView(rlv);


        init(ICNService.ICN_COMMAND_ID.ICN_DIGITAL_SIGNAGE_DEMO, rlv);
        Log.v("SampleDigitalSignage::onCreate", "bindService ...");
        bindService();
    }
```

11. In the onCreate method, add `rlv` (relative layout) to the second parameter and the Command ID of ICNService to the first parameter of init.



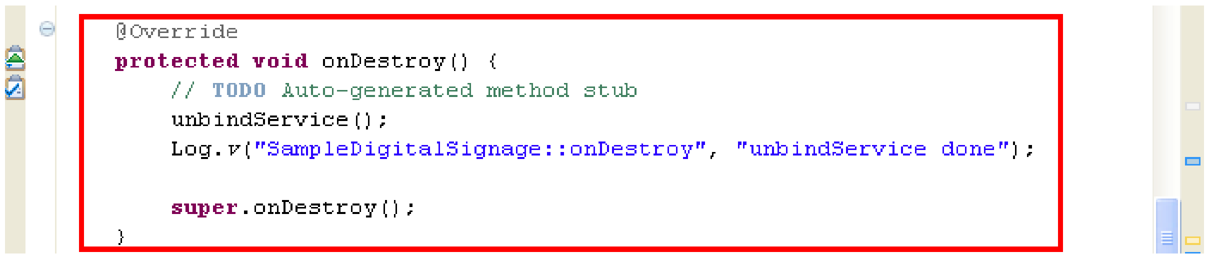
```
        rlv.addView(imagePlayer);
        rlv.addView(mClock);
        rlv.addView(mVideoView);
        rlv.addView(mTicker);
        setContentView(rlv);


        init(ICNService.ICN_COMMAND_ID.ICN_DIGITAL_SIGNAGE_DEMO, rlv);
        Log.v("SampleDigitalSignage::onCreate", "bindService ...");
        bindService();
    }
```

12. In the onCreate method, call the `bindService()` function.



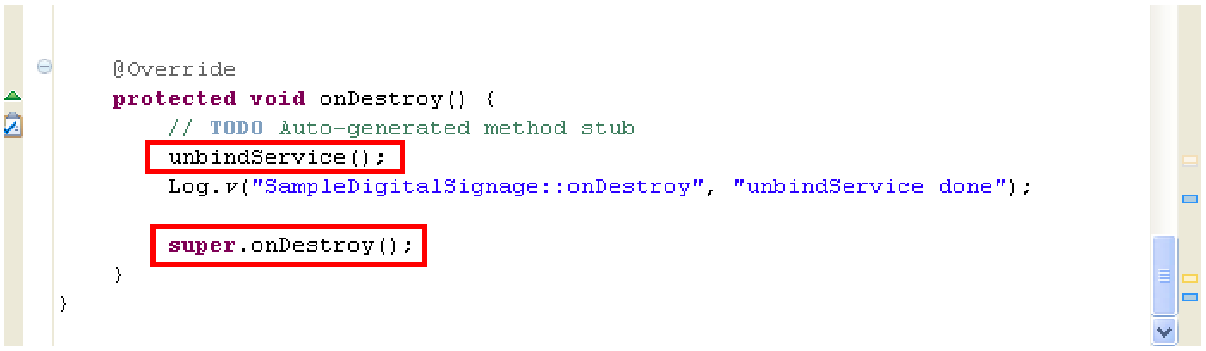
```
        rlv.addView(imagePlayer);
        rlv.addView(mClock);
        rlv.addView(mVideoView);
        rlv.addView(mTicker);
        setContentView(rlv);


        init(ICNService.ICN_COMMAND_ID.ICN_DIGITAL_SIGNAGE_DEMO, rlv);
        Log.v("SampleDigitalSignage::onCreate", "bindService ...");
        bindService();
    }
```

13. In the `SampleDigitalSignage` class, override the `onDestroy` function.



```
    @Override
    protected void onDestroy() {
        // TODO Auto-generated method stub
        unbindService();
        Log.v("SampleDigitalSignage::onDestroy", "unbindService done");

        super.onDestroy();
    }
```

14. In the onDestroy function, add both `unbindService()` and `super.onDestroy()`.

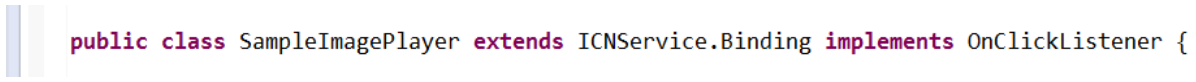


```
    @Override
    protected void onDestroy() {
        // TODO Auto-generated method stub
        unbindService();
        Log.v("SampleDigitalSignage::onDestroy", "unbindService done");

        super.onDestroy();
    }
}
```

Sample5: Image Player

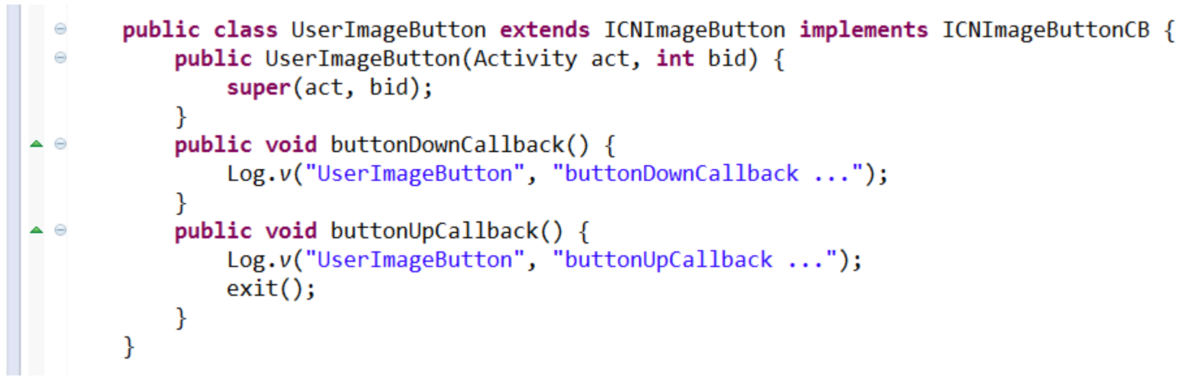
This section illustrates how the `SampleImagePlayer` works, with a simple guide to demonstrate the development of image player

1. Declare a public class named something likes ImagePlayer that extends the `ICNService.Binding` class.



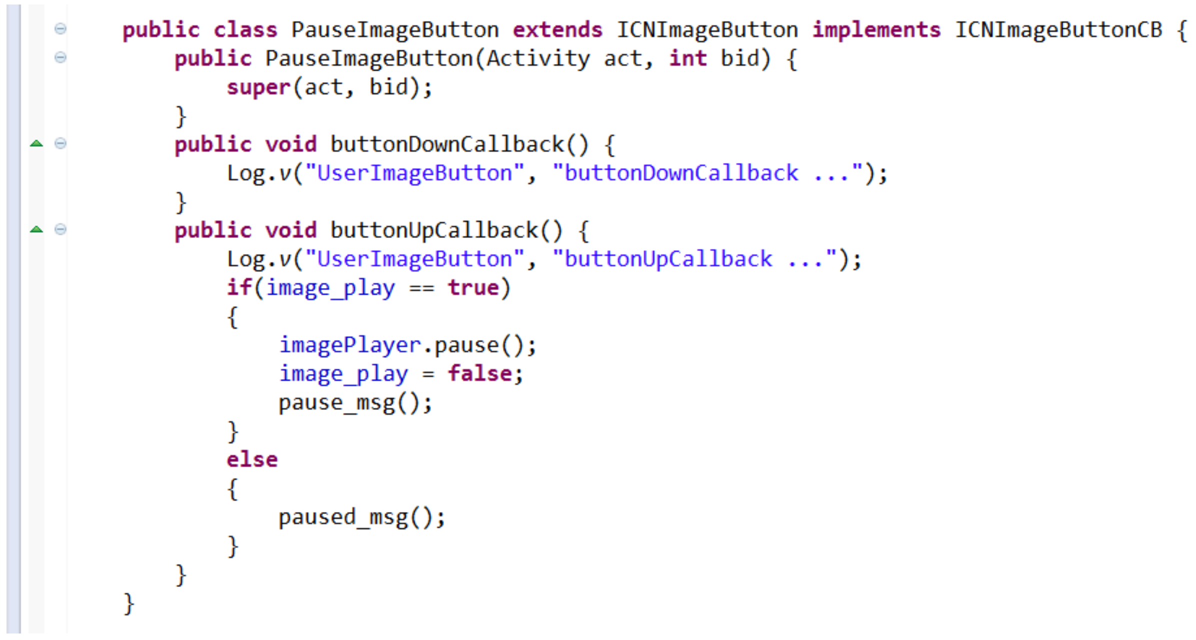
```
public class SampleImagePlayer extends ICNService.Binding implements OnClickListener {
```

2. Declare an ICNImageButton class for the purpose of exit function.



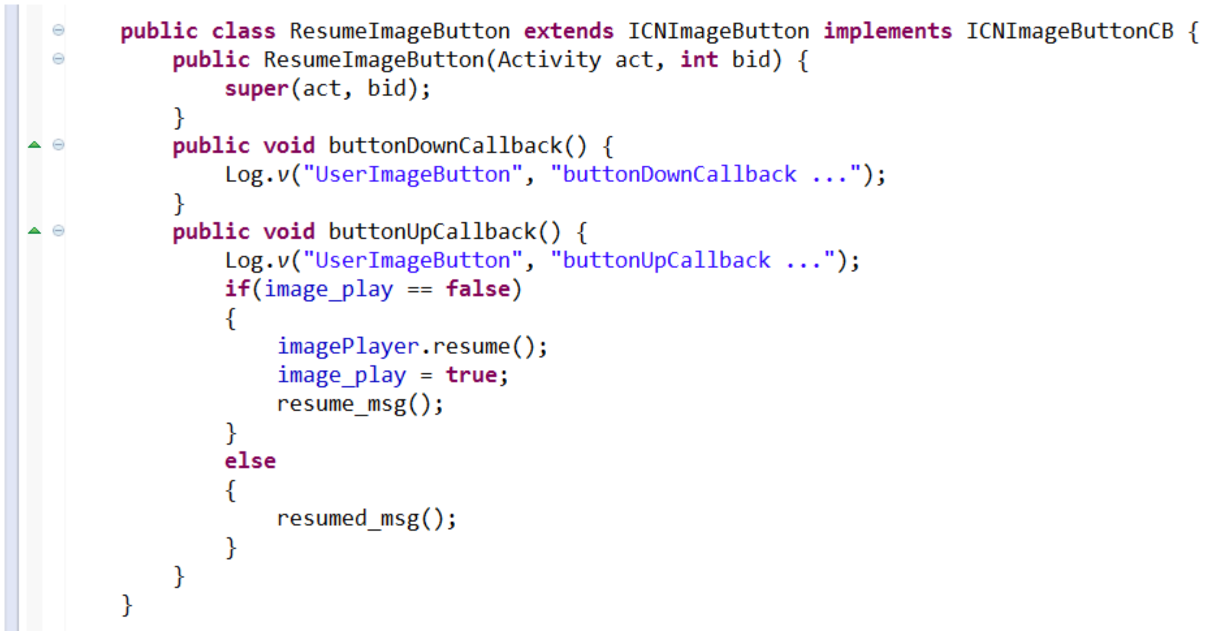
```
    public class UserImageButton extends ICNImageButton implements ICNImageButtonCB {
        public UserImageButton(Activity act, int bid) {
            super(act, bid);
        }
        public void buttonDownCallback() {
            Log.v("UserImageButton", "buttonDownCallback ...");
        }
        public void buttonUpCallback() {
            Log.v("UserImageButton", "buttonUpCallback ...");
            exit();
        }
    }
```

3. Declare an ICNImageButton class for the purpose of pause function.



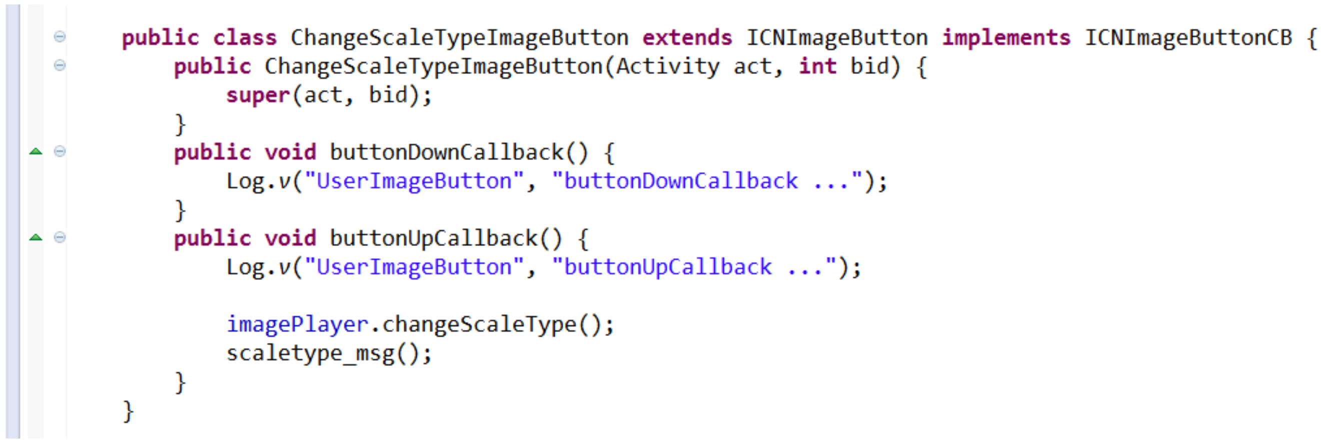
```
    public class PauseImageButton extends ICNImageButton implements ICNImageButtonCB {
        public PauseImageButton(Activity act, int bid) {
            super(act, bid);
        }
        public void buttonDownCallback() {
            Log.v("UserImageButton", "buttonDownCallback ...");
        }
        public void buttonUpCallback() {
            Log.v("UserImageButton", "buttonUpCallback ...");
            if(image_play == true)
            {
                imagePlayer.pause();
                image_play = false;
                pause_msg();
            }
            else
            {
                paused_msg();
            }
        }
    }
```

4. Declare an ICNImageButton class for the purpose of resume function.



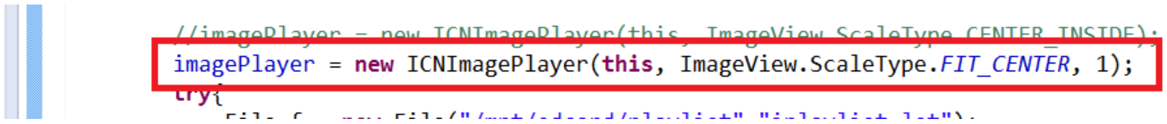
```
public class ResumeImageButton extends ICNImageButton implements ICNImageButtonCB {
    public ResumeImageButton(Activity act, int bid) {
        super(act, bid);
    }
    public void buttonDownCallback() {
        Log.v("UserImageButton", "buttonDownCallback ...");
    }
    public void buttonUpCallback() {
        Log.v("UserImageButton", "buttonUpCallback ...");
        if(image_play == false)
        {
            imagePlayer.resume();
            image_play = true;
            resume_msg();
        }
        else
        {
            resumed_msg();
        }
    }
}
```

5. Declare an ICNImageButton class for the purpose of scale type changing function.



```
public class ChangeScaleTypeImageButton extends ICNImageButton implements ICNImageButtonCB {
    public ChangeScaleTypeImageButton(Activity act, int bid) {
        super(act, bid);
    }
    public void buttonDownCallback() {
        Log.v("UserImageButton", "buttonDownCallback ...");
    }
    public void buttonUpCallback() {
        Log.v("UserImageButton", "buttonUpCallback ...");

        imagePlayer.changeScaleType();
        scaletype_msg();
    }
}
```

6. Initialize the ICNImagePlayer, will be used in the button functions of pause, resume, and scale type change to perform these actions.



```
//imagePlayer = new ICNImagePlayer(this, ImageView.ScaleType.CENTER_INSIDE);
imagePlayer = new ICNImagePlayer(this, ImageView.ScaleType.FIT_CENTER, 1);
try{
```

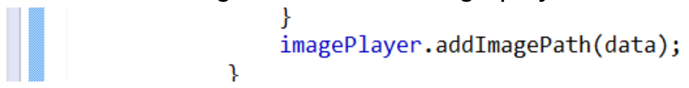
---Parameter definition---

2nd `parameter` : The image scale type

3rd `parameter` : 0 means use image resource inside the project;

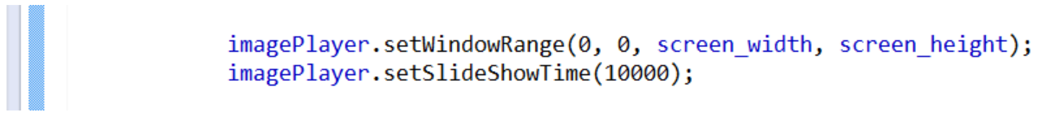
1 means use image resource by external path

7. Add the image resource to image player



```
                    }
                    imagePlayer.addImagePath(data);
                }
```

8. Set image's position, size, and slide show time



```
                imagePlayer.setWindowRange(0, 0, screen_width, screen_height);
                imagePlayer.setSlideShowTime(10000);
```

---Parameter definition---

.SetWindowRange():

1st parameter : Coordinate x of position

2nd parameter : Coordinate y of position

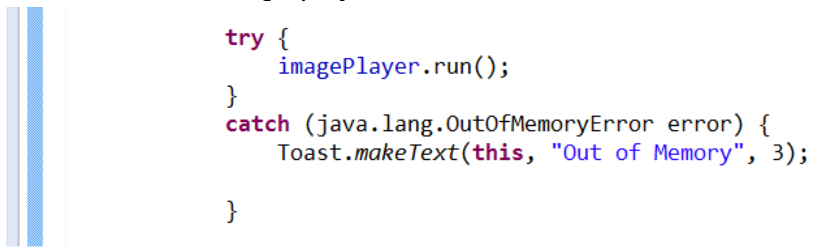
3rd parameter : Width of image

4th parameter : Height of image

.SetSlideShowTime():

parameter : The interval time between each images when slideshow is going on

9. Start to run image player



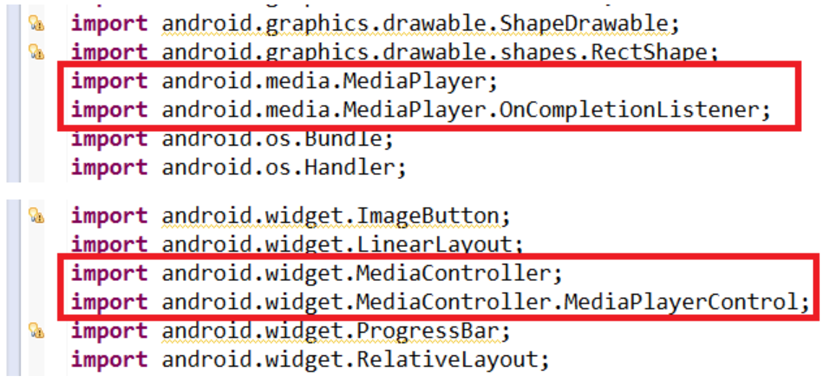
```
                try {
                    imagePlayer.run();
                }
                catch (java.lang.OutOfMemoryError error) {
                    Toast.makeText(this, "Out of Memory", 3);

                }
```

Sample6: Media Player

=== Part 1 ===

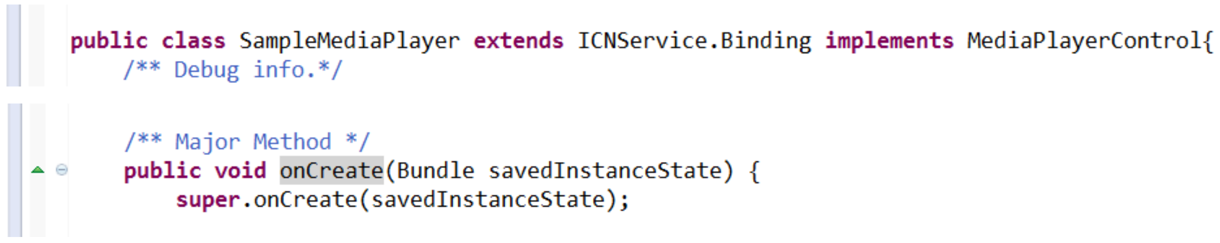
This section illustrates how the `SampleMediaPlayer` works, with a simple guide to demonstrate the development of MediaPlayer.

1. Declare a public class named `SampleDigitalSignage` that extends the `ICNService.Binding` class.



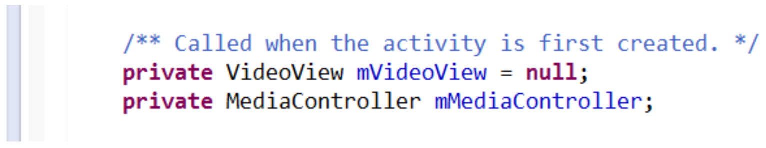
```
import android.graphics.drawable.ShapeDrawable;
import android.graphics.drawable.shapes.RectShape;
import android.media.MediaPlayer;
import android.media.MediaPlayer.OnCompletionListener;
import android.os.Bundle;
import android.os.Handler;

import android.widget.ImageButton;
import android.widget.LinearLayout;
import android.widget.MediaController;
import android.widget.MediaController.MediaPlayerControl;
import android.widget.ProgressBar;
import android.widget.RelativeLayout;
```

2. First of all, you need to create a class that extends an activity. Then, you implement the MediaPlayerControl. If you have remote control feature, you could create a `SampleMediaPlayer` class that extend `ICNService.Binding` instead.



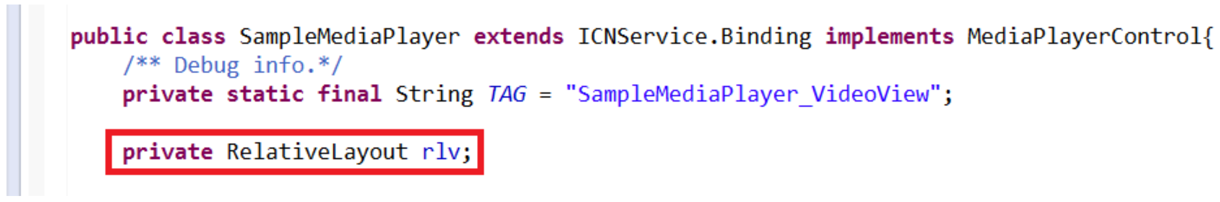
```
public class SampleMediaPlayer extends ICNService.Binding implements MediaPlayerControl{
    /** Debug info.*/

    /** Major Method */
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

3. Then, you can set up the required objects.



```
    /** Called when the activity is first created. */
    private VideoView mVideoView = null;
    private MediaController mMediaController;
```

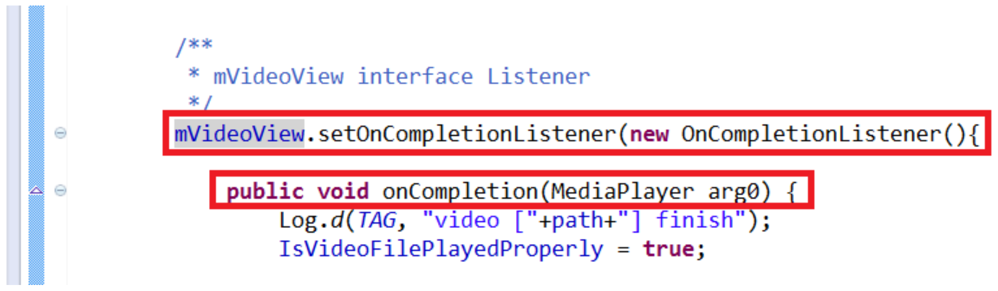
4. To specify the position of the video, use RelativeLayout. Let's create a relative layout. Declare a layout container and its stored location in the VideoView class.



```
public class SampleMediaPlayer extends ICNService.Binding implements MediaPlayerControl{
    /** Debug info.*/
    private static final String TAG = "SampleMediaPlayer_VideoView";

    private RelativeLayout rlv;
```

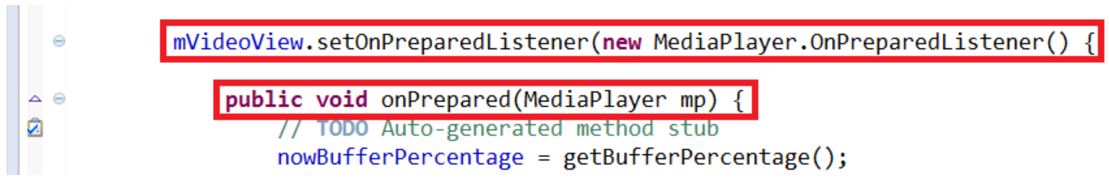
5. In the onCreate function, add the feature and listener staus to the VideoView object and MediaController object.

5.1. `VideoView.setOnCompletionListener`



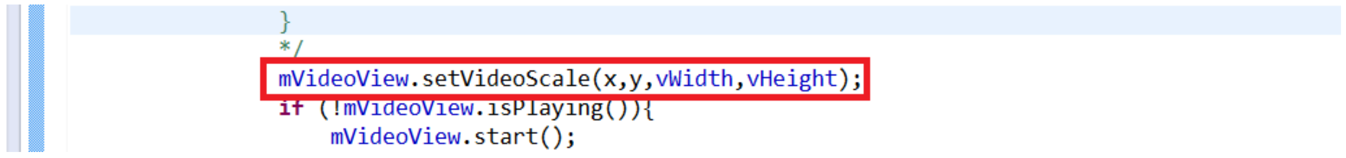
```
/**
 * mVideoView interface Listener
 */
mVideoView.setOnCompletionListener(new OnCompletionListener(){

    public void onCompletion(MediaPlayer arg0) {
        Log.d(TAG, "video ["+path+"] finish");
        IsVideoFilePlayedProperly = true;
```

5.2. `VideoView.setOnPreparedListener`



```
mVideoView.setOnPreparedListener(new MediaPlayer.OnPreparedListener() {

    public void onPrepared(MediaPlayer mp) {
        // TODO Auto-generated method stub
        nowBufferPercentage = getBufferPercentage();
```

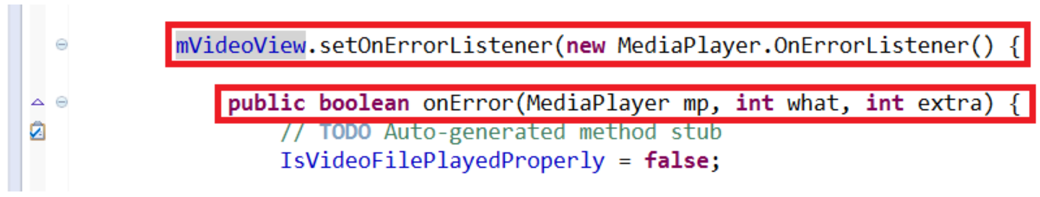
5.2.1. Use `setVideoScale` function within the `onPrepared` function to set the position and size of the media player.



```
        }
        */
        mVideoView.setVideoScale(x,y,vWidth,vHeight);
        if (!mVideoView.isPlaying()){
            mVideoView.start();
```

The above explains how you can specify the position of and resize your video. Part of the methods are created and owned by ICNexus. If you need other functions, you may also try to edit on your own.

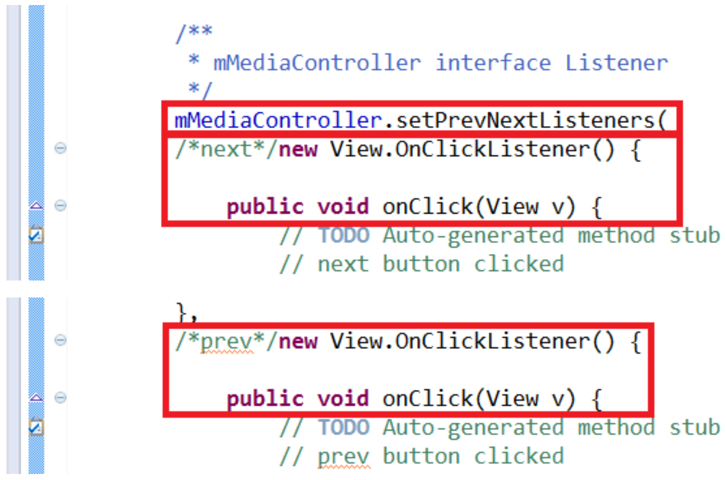
5.3. `VideoView.setOnErrorListener`



```
mVideoView.setOnErrorListener(new MediaPlayer.OnErrorListener() {

    public boolean onError(MediaPlayer mp, int what, int extra) {
        // TODO Auto-generated method stub
        IsVideoFilePlayedProperly = false;
```

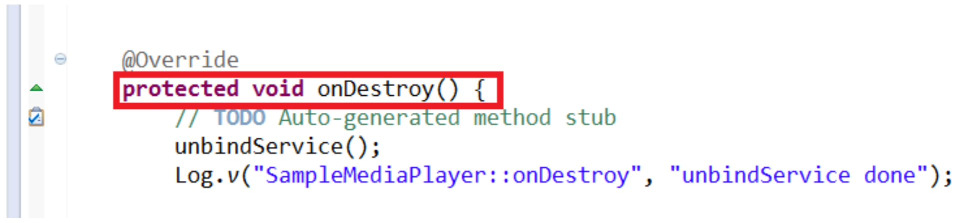
`ErrorListener` is an optional item. You do not have to customize or respond to an error message.

6. Users must customize their own button method of cache replacement for streaming media. Below is an example:



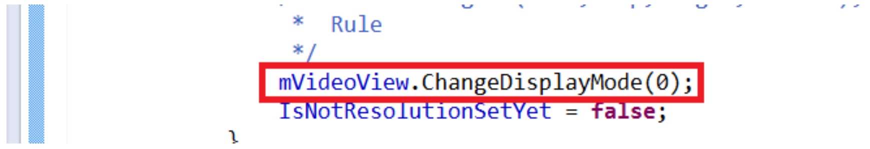
```
/**
 * mMediaController interface Listener
 */
mMediaController.setPrevNextListeners(
/*next*/new View.OnClickListener() {

    public void onClick(View v) {
        // TODO Auto-generated method stub
        // next button clicked

},
/*prev*/new View.OnClickListener() {

    public void onClick(View v) {
        // TODO Auto-generated method stub
        // prev button clicked
```

7. It is recommended to call the onDestry() method to stop the process so that the service is effectively terminated.



```
@Override
protected void onDestroy() {
    // TODO Auto-generated method stub
    unbindService();
    Log.v("SampleMediaPlayer::onDestroy", "unbindService done");
```

8. There are four video display modes: the origin / natural size, full screen (fixed length to width ratio), custom size screen (fixed length to width ratio) and free size (you may set the length and the width any form you like). The video display modes range from 0 to 3. It is recommended that you choose 2 (custom size screen).

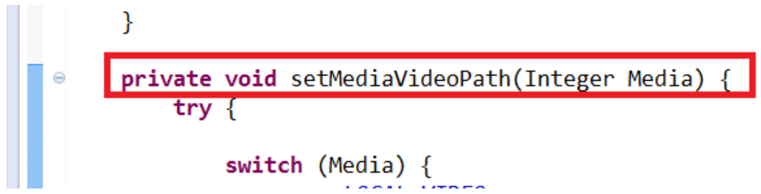


```
 *   Rule
 */
mVideoView.ChangeDisplayMode(0);
IsNotResolutionSetYet = false;
}
```

=== Part 2 ===

This section tells you how to control playback of your media files, such as setting a looping order, play/pause and next/previous.
Here we will use some examples which you can apply to a variety of tasks, such as modifying your playback sequence, adding more media files during the running time and exchanging media files.
Below are the examples. Please note that we did not provide the full source code. If you would like to view the full code, please refer to your JAR files.

1. Confirm the path for the file. This process appears once every time you start the program. You may also call it *initialization check*.



```
}

private void setMediaVideoPath(Integer Media) {
    try {

        switch (Media) {
```

2. We will not go into the detailed discussion of VideoView and MediaController. Here you will learn basic principles of the playback controls, such as play, pause, changing timelines and stop.

Name	Description
`public void start()`	Starts or resumes playback.
`public void pause()`	Pauses playback.
`public int getDuration()`	Gets the duration of the current media file.
`public void seekTo(int pos)`	Seeks to specified time position. The offset is in milliseconds from the start to seek to. Must be smaller than the total duration.
`public int getCurrentPosition()`	Gets the current playback position.
`public int getBufferPercentage()`	The percentage of the current Video that has been buffered.

To exit the player completely, you must close the player window or call the stopCallback method via remote control. If you would like to stop playing the current media file only, please use Pause or other restart methods.

3. The service implementation examples include seven methods, control functions and corresponding functions as shown below.

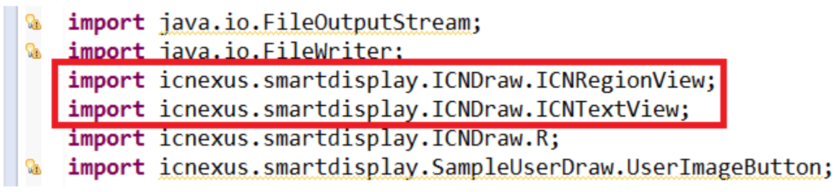
Name	Description
`playCallback ()`	start()
`stopCallback ()`	pause()
`pauseCallback ()`	pause()
`nextCallback ()`	setPreNextListeners
`prevCallback ()`	setPreNextListeners
`fastForwardCallback ()`	copy from MediaController
`fastBackCallback ()`	copy from MediaController

If you would like to add new commands, please see the Service section.

Sample7: Usage GUI

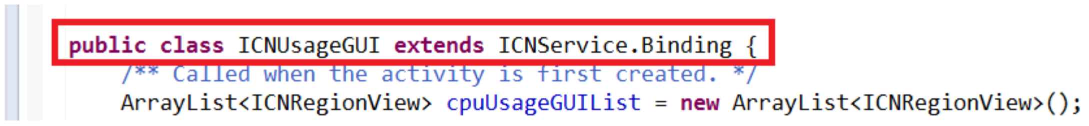
This section illustrates how the `SampleUsageGUI` works, with a simple guide to demonstrate the development of `UsageGUI`.

1. Import the icnexus library `ICNDraw.ICNRegionView` and `ICNDraw.ICNTextView`.



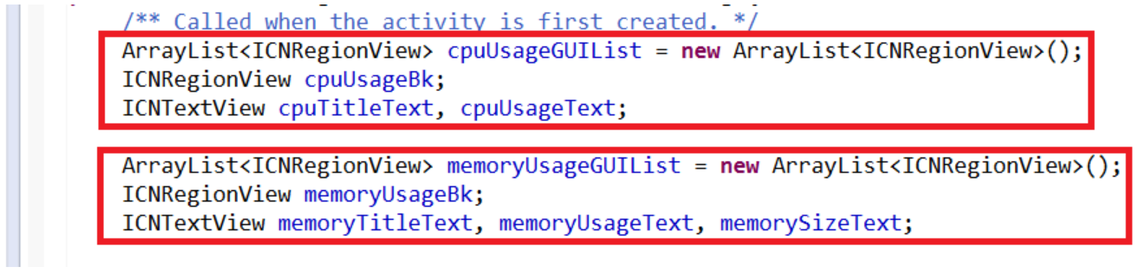
```
import java.io.FileOutputStream;
import java.io.FileWriter;
import icnexus.smartdisplay.ICNDraw.ICNRegionView;
import icnexus.smartdisplay.ICNDraw.ICNTextView;
import icnexus.smartdisplay.ICNDraw.R;
import icnexus.smartdisplay.SampleUserDraw.UserImageButton;
```

2. Declare a public class named something likes UsageGUI that extends the `ICNService.Binding` class.



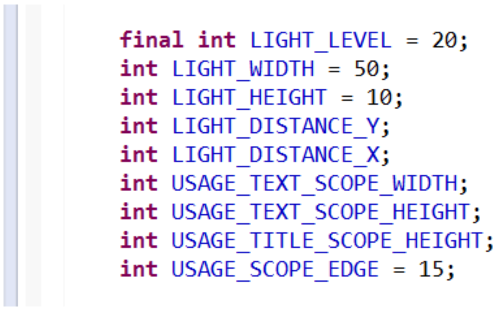
```
public class ICNUsageGUI extends ICNService.Binding {
    /** Called when the activity is first created. */
    ArrayList<ICNRegionView> cpuUsageGUIList = new ArrayList<ICNRegionView>();
```

3. Declare ICNRegionView object to construct usage ui of CPU and memory. Declare ICNTextView object to show the usage text.



```
    /** Called when the activity is first created. */
    ArrayList<ICNRegionView> cpuUsageGUIList = new ArrayList<ICNRegionView>();
    ICNRegionView cpuUsageBk;
    ICNTextView cpuTitleText, cpuUsageText;

    ArrayList<ICNRegionView> memoryUsageGUIList = new ArrayList<ICNRegionView>();
    ICNRegionView memoryUsageBk;
    ICNTextView memoryTitleText, memoryUsageText, memorySizeText;
```

4. Modify usage GUI parameters.



```
    final int LIGHT_LEVEL = 20;
    int LIGHT_WIDTH = 50;
    int LIGHT_HEIGHT = 10;
    int LIGHT_DISTANCE_Y;
    int LIGHT_DISTANCE_X;
    int USAGE_TEXT_SCOPE_WIDTH;
    int USAGE_TEXT_SCOPE_HEIGHT;
    int USAGE_TITLE_SCOPE_HEIGHT;
    int USAGE_SCOPE_EDGE = 15;
```

--- Variable definition---

`final int LIGHT_LEVEL` : Level numbers of usage

`int LIGHT_WIDTH` : Width of a usage bar

`int LIGHT_HEIGHT` : Height of a usage bar

`int USAGE_SCOPE_EDGE` : The distance from background outline to usage bar

5. Set exit button.

```
info_update(false);
IBInfo ibi;
if((screen_width / 8) >= (screen_height / 5))
    ibi = new IBInfo(screen_width * 9 / 10, 0, screen_width / 10, screen_height / 10, "", R.drawabl
else
    ibi = new IBInfo(screen_width * 9 / 10, 0, screen_width / 10, screen_width / 16, "", R.drawable
UserImageButton ib = new UserImageButton(this, 0);
ib.setButtonUpAuto(true);
if((screen_width * 3) >= (screen_height * 5))
    ib.setFontSize(screen_height * 30 / 480);
else
    ib.setFontSize(screen_width * 30 / 800);
ib.createButton(ibi);
ib.setText(getText(R.string.exit).toString());
rlv_main.addView(ib);
setContentView(rlv_main);
```

6. Update memory usage and CPU usage for serviceCallBack.

```
public void info_update(boolean justInfoRefresh)
{
    if (justInfoRefresh){
        readMEMUsage();
        readCPUUsage();
        updateUsedMemorySize(Mem_Used, ICN_MEM_UNIT.ICN_MEM_KB);
    }else{
        updateMemoryUsage((int)readMEMUsage());
        updateCPUUsage((int)readCPUUsage());
        updateUsedMemorySize(Mem_Used, ICN_MEM_UNIT.ICN_MEM_KB);
    }
}
```

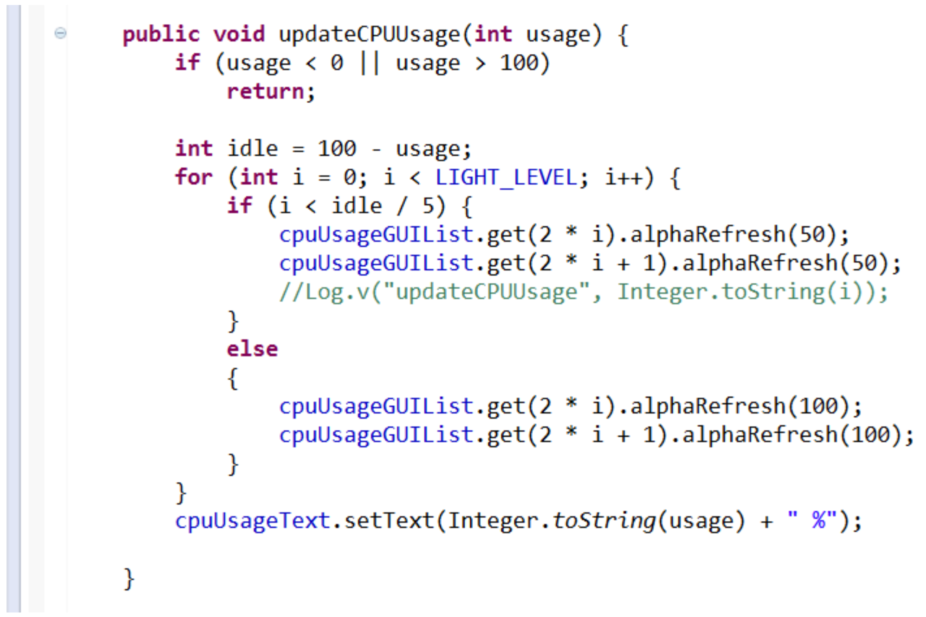
7. Initialize the CPU usage GUI.

```
public void createCPUUsageGUI(RelativeLayout rlv, int x, int y) {
```

8. Initialize the memory usage GUI.

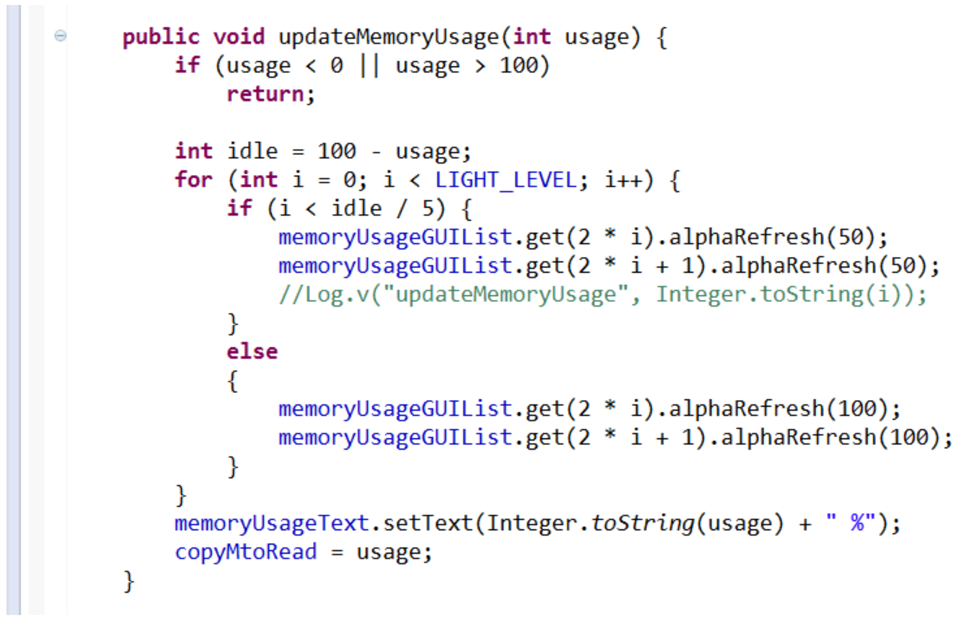
```
public void createMemoryUsageGUI(RelativeLayout rlv, int x, int y) {
```

9. Update the CPU usage GUI with input value.



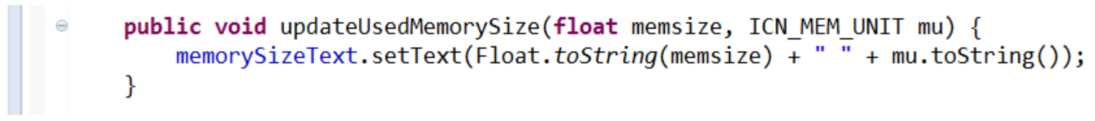
```
public void updateCPUUsage(int usage) {
    if (usage < 0 || usage > 100)
        return;

    int idle = 100 - usage;
    for (int i = 0; i < LIGHT_LEVEL; i++) {
        if (i < idle / 5) {
            cpuUsageGUIList.get(2 * i).alphaRefresh(50);
            cpuUsageGUIList.get(2 * i + 1).alphaRefresh(50);
            //Log.v("updateCPUUsage", Integer.toString(i));
        }
        else
        {
            cpuUsageGUIList.get(2 * i).alphaRefresh(100);
            cpuUsageGUIList.get(2 * i + 1).alphaRefresh(100);
        }
    }
    cpuUsageText.setText(Integer.toString(usage) + " %");

}
```

10. Update the memory usage GUI with input value.



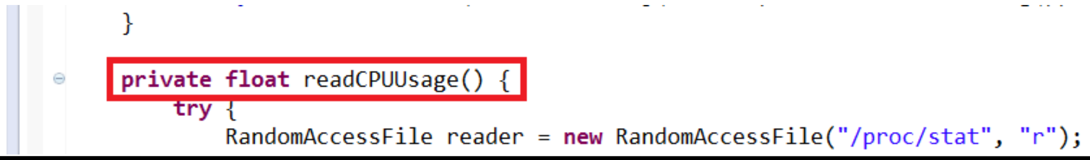
```
public void updateMemoryUsage(int usage) {
    if (usage < 0 || usage > 100)
        return;

    int idle = 100 - usage;
    for (int i = 0; i < LIGHT_LEVEL; i++) {
        if (i < idle / 5) {
            memoryUsageGUIList.get(2 * i).alphaRefresh(50);
            memoryUsageGUIList.get(2 * i + 1).alphaRefresh(50);
            //Log.v("updateMemoryUsage", Integer.toString(i));
        }
        else
        {
            memoryUsageGUIList.get(2 * i).alphaRefresh(100);
            memoryUsageGUIList.get(2 * i + 1).alphaRefresh(100);
        }
    }
    memoryUsageText.setText(Integer.toString(usage) + " %");
    copyMtoRead = usage;
}
```

11. Update the label of memory usage value.



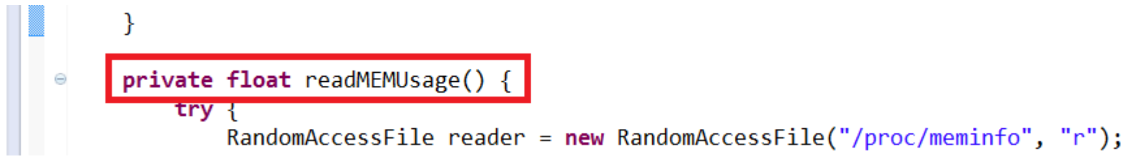
```
public void updateUsedMemorySize(float memsize, ICN_MEM_UNIT mu) {
    memorySizeText.setText(Float.toString(memsize) + " " + mu.toString());
}
```

12. Read the local CPU usage value.



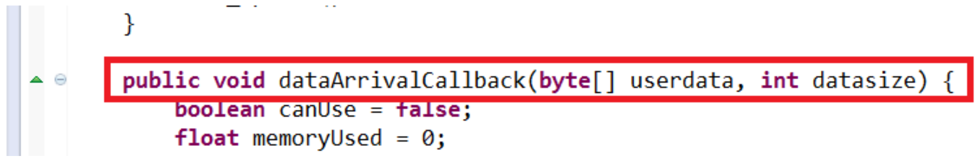
```
}

private float readCPUUsage() {
    try {
        RandomAccessFile reader = new RandomAccessFile("/proc/stat", "r");
```

13. Read the local memory usage value.



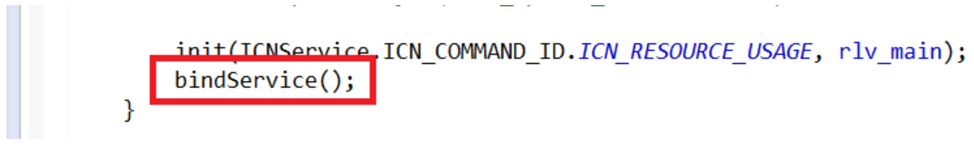
```
    }

    private float readMEMUsage() {
        try {
            RandomAccessFile reader = new RandomAccessFile("/proc/meminfo", "r");
```

14. The service call back function for the usage update of remote device.



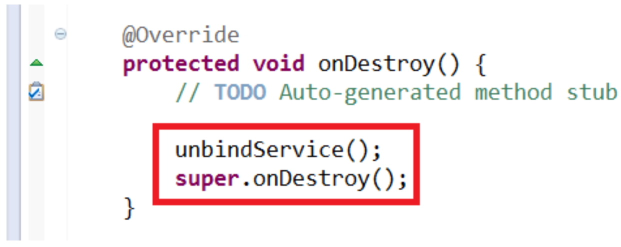
```
    }

    public void dataArrivalCallback(byte[] userdata, int datasize) {
        boolean canUse = false;
        float memoryUsed = 0;
```

15. In the onCreate method, call the `bindService()` function.



```
        init(ICNService.ICN_COMMAND_ID.ICN_RESOURCE_USAGE, rlv_main);
        bindService();
    }
```

16. In the onDestroy function, add `unbindService()` and `super.onDestroy()`.

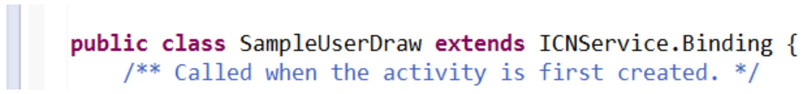


```
    @Override
    protected void onDestroy() {
        // TODO Auto-generated method stub

        unbindService();
        super.onDestroy();
    }
```

Sample8: User Draw

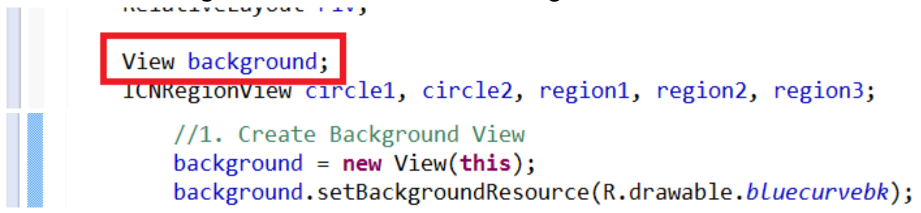
This section illustrates how the `SampleUserDraw` works.

1. Declare a public class named something likes UserDraw that extends the `ICNService.Binding` class



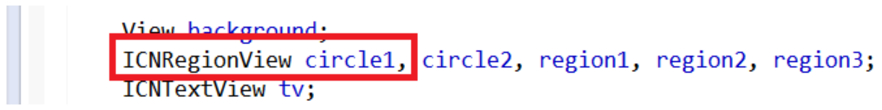
```
public class SampleUserDraw extends ICNService.Binding {
    /** Called when the activity is first created. */
```

2. Set an image resource to be the background

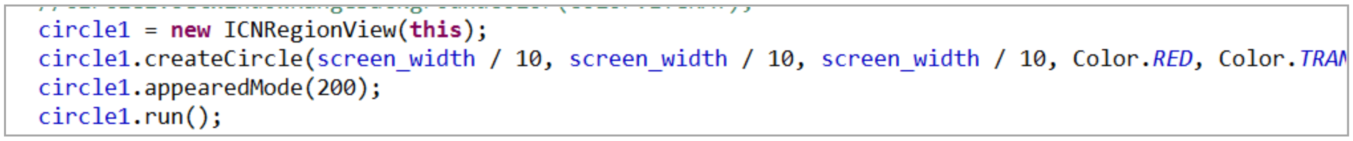


```
    RelativeLayout rl;

    View background;
    ICNRegionView circle1, circle2, region1, region2, region3;

        //1. Create Background View
        background = new View(this);
        background.setBackgroundResource(R.drawable.bluecurvebk);
```

3. Create a circle pre-defined by ICNEXUS



```
    View background;
    ICNRegionView circle1, circle2, region1, region2, region3;
    ICNTextView tv;
```



```
  circle1 = new ICNRegionView(this);
  circle1.createCircle(screen_width / 10, screen_width / 10, screen_width / 10, Color.RED, Color.TRAN
  circle1.appearedMode(200);
  circle1.run();
```

---Parameter definition---

`.createCircle():`

1st `parameter` : Coordinate x of position

2nd `parameter` : Coordinate y of position

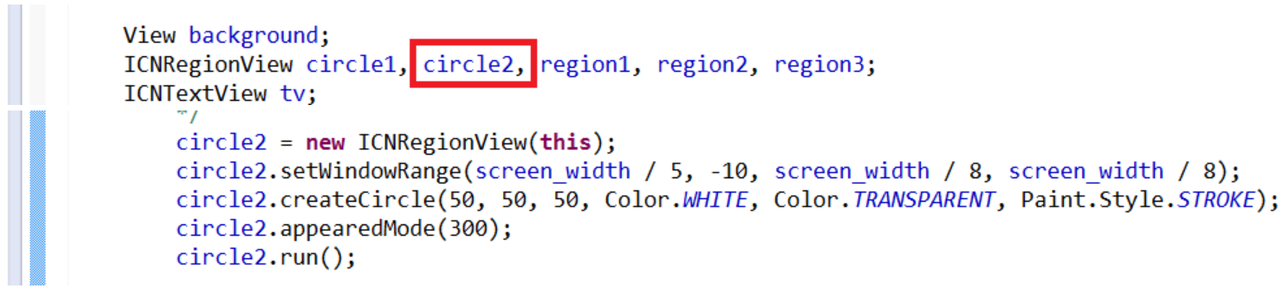
3rd `parameter` : The radius of circle

4th `parameter` : The foreground color

5th `parameter` : The background color

6th `parameter` : The graphics paint style

4. Create a circle pre-defined by ICNEXUS and limit it in a specific range



```
    View background;
    ICNRegionView circle1, circle2, region1, region2, region3;
    ICNTextView tv;
        */
        circle2 = new ICNRegionView(this);
        circle2.setWindowRange(screen_width / 5, -10, screen_width / 8, screen_width / 8);
        circle2.createCircle(50, 50, 50, Color.WHITE, Color.TRANSPARENT, Paint.Style.STROKE);
        circle2.appearedMode(300);
        circle2.run();
```

---Parameter definition---

.setWindowRange():

1st parameter : Coordinate x of window range
2nd parameter : Coordinate y of window range
3rd parameter : The width of window range
4th parameter : The height of window range

5. Create a rectangle pre-defined by ICNEXUS

```
        //4. Create Rectangle View
        region1 = new ICNRegionView(this);
        Rect rect1 = new Rect(40, 40, 130, 130);
        region1.createRect(rect1, Color.YELLOW, Color.TRANSPARENT, Paint.Style.FILL_AND_STROKE);
        region1.disappearedMode(80);
        region1.run();
```

---Parameter definition---

new Rect():

1st parameter : Coordinate x of rectangle
2nd parameter : Coordinate y of rectangle
3rd parameter : The screen width – rectangle width – x of rectangle
4th parameter : The screen height – rectangle height – y of rectangle

.createRect():

1st parameter : Rect
2nd parameter : The foreground color
3rd parameter : The background color
4th parameter : The graphics paint style

6. Create a text area pre-defined by ICNEXUS

```
//5. Create Text View
tv = new ICNTextView(this);
tv.setWindowRange(screen_width / 16, screen_height / 3, screen_width * 2 / 5, screen_height / 3);
```

---Parameter definition---

.setWindowRange():

1st parameter : Coordinate x of position
2nd parameter : Coordinate y of position
3rd parameter : The width of text area
4th parameter : The height of text area

6.1. Set text shape

```
Path p = new Path();
p.moveTo(0, screen_height * 150 / 480);
p.cubicTo(0, screen_height * 150 / 480, screen_width * 100 / 800, screen_height * 75 / 480, screen_
tv.setPath(p);
tv.setText("Graphical Demo");
tv.setTextSize(screen_width * 26 / 800);
```

---Parameter definition---

.moveTo():

1st parameter : Coordinate x of beginning contour

2nd parameter : Coordinate y of beginning contour

.cubicTo():

1st parameter : The x-coordinate of the first control point on a cubic curve

2nd parameter : The y-coordinate of the first control point on a cubic curve

3rd parameter : The x-coordinate of the second control point on a cubic curve

4th parameter : The y-coordinate of the second control point on a cubic curve

5th parameter : The x-coordinate of the end point on a cubic curve

6th parameter : The y-coordinate of the end point on a cubic curve

6.2. Set text color and format

```
        tv.setTextColor(Color.BLUE);
        tv.setBackgroundColor(Color.TRANSPARENT);
        Typeface tf = Typeface.createFromAsset(getAssets(), "fonts/bullpen3.ttf");
        tv.setTypeface(tf);
        tv.appearedMode(100);
        tv.run();
```

---Parameter definition---

.setTextColor():

parameter : Text color

.setBackgroundColor():

parameter : Text background color

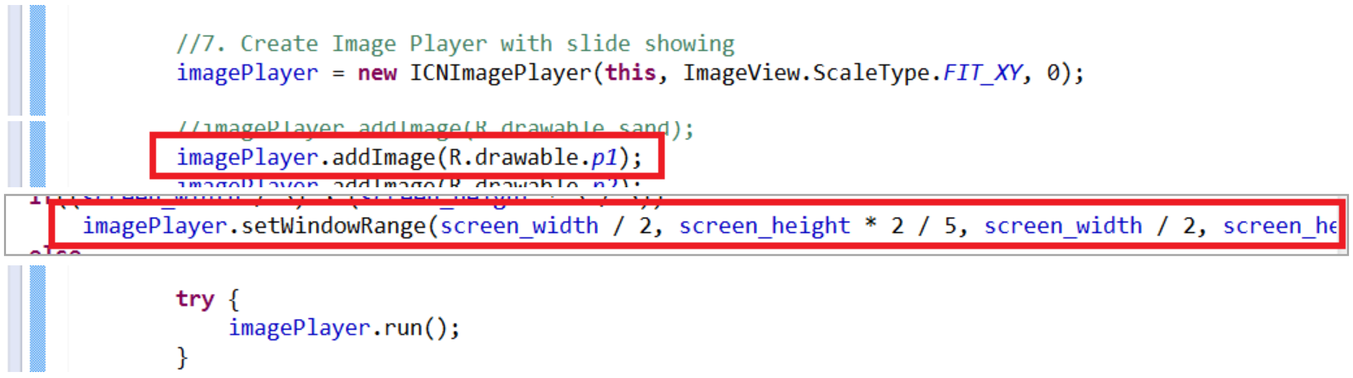
.setTypeface():

parameter : Typeface

7. Create a rectangle area pre-defined by ICNEXUS

```
//6. Create Round Rectangle View
region2 = new ICNRegionView(this);
Rect roundRect = new Rect(screen_width / 10, screen_height * 5 / 6, screen_width / 2, screen_height
region2.createRoundRect(roundRect, Color.BLUE, Color.TRANSPARENT, Paint.Style.FILL_AND_STROKE);
region2.appearedMode(800);
region2.run();
```

8. Create image player pre-defined by ICNEXUS, reference to SampleImagePlayer



```
        //7. Create Image Player with slide showing
        imagePlayer = new ICNImagePlayer(this, ImageView.ScaleType.FIT_XY, 0);

        //imagePlayer.addImage(R.drawable.sand);
        imagePlayer.addImage(R.drawable.p1);
```

```
  imagePlayer.setWindowRange(screen_width / 2, screen_height * 2 / 5, screen_width / 2, screen_he
```

```
        try {
            imagePlayer.run();
        }
```

Chapter 5: Test Application

This chapter will explain the test program action and process of every function, base on our test APP '**Semi Auto Test**'. You can match the source code – **`ICNSemiAutoTest.java`** – with this chapter.

5-1. onCreate

Called when the activity is starting

1.1 setContentView

Set the activity content from a layout resource

1.2 PermissionsGet

Get super user permission

1.3 initAudioPlayer

Initialize the audio player and set audio source:
/system/media/audio/ringtones/DreamTheme.ogg

1.4 ScreenColorSet

Set background color to #FFFFFF

1.5 ComPortPriority

Adjust priority of the assigned Com port

1.6 DeviceIDFilePriority

Adjust priority of the file /proc/cmdline that contains information of the board

1.7 BoardDialogHandler

Go to Step2

5-2. Runnable boardDialogHandler

Call getBoardTypeDialog

2.1 AlertDialog getBoardTypeDialog

Show the board type list to user for selection

2.1.1 setSingleChoiceItems

Get board type

2.1.2 setPositiveButton

2.1.2.1 GetDeviceID

Get device id from /proc/cpuinfo or /proc/cmdline depends on board type

2.1.2.2 testItemDialogHandler

Go to Step3

2.1.3 setNegativeButton

Exit

5-3. testItemDialogHandler

Call getTestItemDialog

3.1 AlertDialog getTestItemDialog

Show the test item list to user for selection

3.1.1 Load_test_pref

Load the preference of selected test item saved at the previous time

3.1.2 setMultiChoiceItems

Update the test item checking status while the click action performed by user

3.1.3 setPositiveButton

3.1.3.1 save_test_pref

Save the preference of the selected test for the next time

3.1.3.2 TouchCalibrationConfirmDialog

Go to Step4

3.1.4 setNegativeButton

Perform the toggle or un-toggle action of the total test item list

5-4. Runnable TouchCalibrationConfirmDialog

If this test item was not checked, go to Step 5

4.1 AlertDialog getTouchCalibrationDialog

Show a dialog questioned the user whether or not tests the Touch Calibration

4.1.1 setPositiveButton

4.1.1.1 Calibration_Launch

Lunch the touch screen calibration application touchscreen.calibration/.TouchCalibration

4.1.1.2 pwmDialogHandler

Go to Step5

4.1.2 setNegativeButton

Exit

5-5. Runnable pwmDialogHandler

If this test item was not checked, go to Step 6

5.1 AlertDialog getPWMDialog

Show a dialog questioned the user whether or not tests the power management

5.1.1 setPositiveButton

5.1.1.1 Runnable PWMTest

Call PWMTest thread to test the power management and then go to Step 6

5.1.2 setNegativeButton

Exit

5-6. Runnable BrightSpotDialogHandler

If this test item was not checked, go to Step 7

6.1 AlertDialog getBrightSpotDialo

Show a dialog questioned the user whether or not does the screen bright spot test

6.1.1 setPositiveButton

6.1.1.1 Runnable screenColorDialogHandler

Call screenColorDialogHandler thread to do the screen bright spot test and then go to Step 7

6.1.2 **setNegativeButton**

Exit

5-7. Runnable speakerDialogHandler

If this test item was not checked, go to Step 8

7.1 **AlertDialog getSpeakerDialog**

Show a dialog questioned the user whether or not does the speaker test

7.1.1 **setPositiveButton**

7.1.1.1 **mPlayer.start**

Call mPlayer function to do the speaker test by playing music and then go to Step 8

7.1.2 **setNegativeButton**

Exit

5-8. Runnable recorderDialogHandler

If this test item was not checked, go to Step 9

8.1 **AlertDialog getRecorderDialog**

Show a dialog questioned the user whether or not does the audio recording test

8.1.1 **setPositiveButton**

8.1.1.1 **mPlayer.stop**

Stop mPlayer

8.1.1.2 **Recorder_Launch**

Call recorder application com.android.soundrecorder/.SoundRecorder to do the recording test and then go to Step 9

8.1.2 **setNegativeButton**

Exit

5-9. Runnable uart0DialogHandler

If this test item was not checked, go to Step 10

9.1 **AlertDialog getUart0Dialog**

Show a dialog questioned the user whether or not does the Uart0 test

9.1.1 **setPositiveButton**

9.1.1.1 **Runnable uart0Handler**

Call uart0Handler thread to do the Uart0 test and then go to Step 10

9.1.2 **setNegativeButton**

Exit

5-10. Runnable uart1DialogHandler

If this test item was not checked, go to Step 11

10.1 AlertDialog getUart1Dialog

Show a dialog questioned the user whether or not does the Uart1 test

10.1.1 setPositiveButton

10.1.1.1 Runnable uart1Handler

Call uart1Handler thread to do the Uart1 test and then go to Step 11

10.1.2 setNegativeButton

Exit

5-11. Runnable uart2DialogHandler

If this test item was not checked, go to Step 12

11.1 AlertDialog getUart2Dialog

Show a dialog questioned the user whether or not does the Uart2 test

11.1.1 setPositiveButton

11.1.1.1 Runnable uart2Handler

Call uart2Handler thread to do the Uart2 test and then go to Step 12

11.1.2 setNegativeButton

Exit

5-12. Runnable uart4rs485DialogHandler

If this test item was not checked, go to Step 13

12.1 AlertDialog getUart4rs485Dialog

Show a dialog questioned the user whether or not does the Uart4 (RS485) test

12.1.1 setPositiveButton

12.1.1.1 Runnable uart4rs485Handler

Call uart4rs485Handler thread to do the Uart4 (RS485) test and then go to Step 13

12.1.2 setNegativeButton

Exit

5-13. Runnable uart5DialogHandler

If this test item was not checked, go to Step 14

13.1 AlertDialog getUart5Dialog

Show a dialog questioned the user whether or not does the Uart5 test

13.1.1 setPositiveButton

13.1.1.1 Runnable uart5Handler

Call uart5Handler thread to do the Uart5 test and then go to Step 14

13.1.2 setNegativeButton

Exit

5-14. Runnable i2cDialogHandler

If this test item was not checked, go to Step 15

14.1 Runnable i2c_C600Handler

If the board type got from Step2 was C600, call i2c_C600Handler thread to do the i2c test and then go to Step 15

14.2 AlertDialog getI2cTestDialog

Show a dialog questioned the user whether or not does the i2c test

14.2.1 setPositiveButton

14.2.1.1 Runnable i2c_8574_Handler

If the board type got from Step2 was C500, call i2c_8574_Handler thread to do the i2c test and then go to Step 15

14.2.1.2 Runnable i2cHandler

Call i2cHandler thread to do the i2c test and then go to Step 15

14.2.2 setNegativeButton

Exit

5-15. Runnable gpioDialogHandler

If this test item was not checked, go to Step 16

15.1 Runnable gpioAutoTestHandler

If the board type got from Step2 was C600, call gpioAutoTestHandler thread to do the GPIO test and then go to Step 16

15.2 Runnable gpioHandler

Call gpioHandler thread to do the GPIO test and then go to Step 16

5-16. Runnable ethernetDialogHandler

If this test item was not checked, go to Step 17

16.1 Runnable ethernetEnableHandler

Call the ethernetEnableHandler thread to do the Ethernet enable action in the beginning

16.2 Runnable ethernetDHCPHandler

Call the ethernetDHCPHandler thread after the Ethernet enabled to do the Ethernet DHCP action

16.3 Runnable ethernetPingHandler

Call ethernetPingHandler thread finally to do Ethernet ping action for Ethernet test and then go to Step 17

5-17. Runnable wifiDialogHandler

If this test item was not checked, go to Step 18

17.1 Runnable wifiProcessHandler

Call the wifiProcessHandler thread to do the Wi-Fi prepared action (the default wireless ap is “ICNEXUS-1”) in the beginning

17.2 Runnable wifiPingHandler

Call wifiPingHandler thread finally to do the Wi-Fi ping action for Wi-Fi test and then go to Step 18

5-18. Runnable macDialogHandler

If this test item was not checked, go to Step 19

18.1 Runnable macProcessHandler

Call macProcessHandler thread to do the MAC test and then go to Step 19

5-19. Runnable threeGDialogHandler

If this test item was not checked or the board type got from Step2 was not C600, go to Step 20

19.1 Runnable threeGProcessHandler

Call threeGProcessHandler thread to do the 3G modem tests (the default modem is Huawei and need to make sure the command "pppd call huawei" is work) and then go to Step 20

5-20. Runnable storageTestBeginHandler

If this test item was not checked, go to Step 21

20.1 Runnable storageTestBeginHandler

Call storageTestBeginHandler thread to do the external storage read/write tests (include SD card and all USB flash drive) and then go to Step 21

5-21. Runnable reportDialogHandler

21.1 AlertDialog getReportDialog

Show a dialog that list the results of the automatically test items and go to Step 22 after user click the "OK" button

5-22. Runnable finishDialogHandler

22.1 AlertDialog getFinishDialog

Show a test finish dialog

22.1.1 setPositiveButton

Save the test report and then exit

22.1.2 setNegativeButton

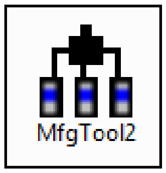
Exit and without sawing test report

Chapter 6: Download Firmware Image

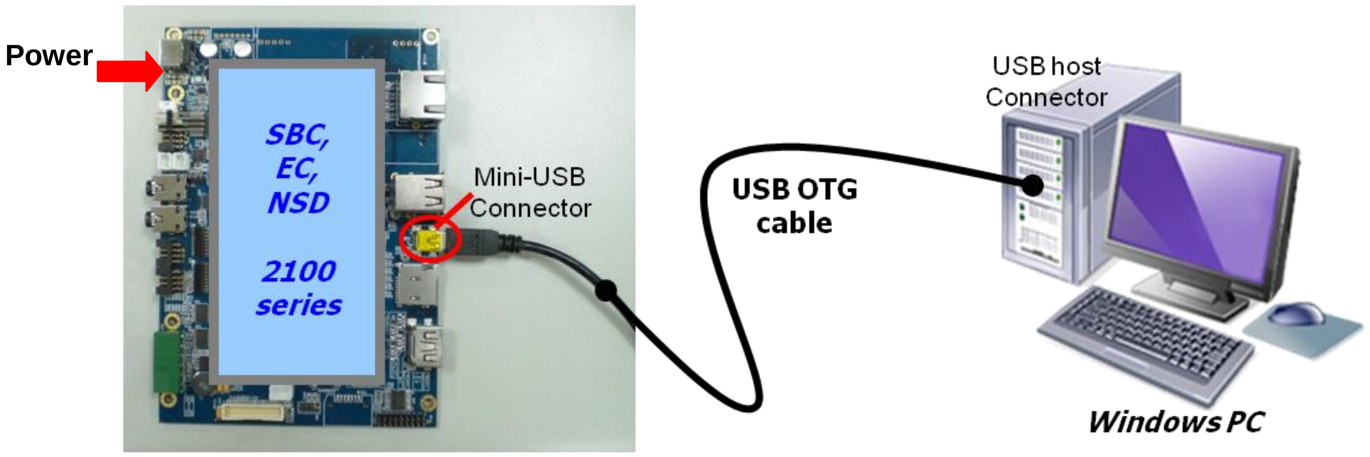
6.1 Software Utility

If you want to update the operating system, before getting the new image you need to have the uploading tool first. The tool we use is called **MfgTool2**.

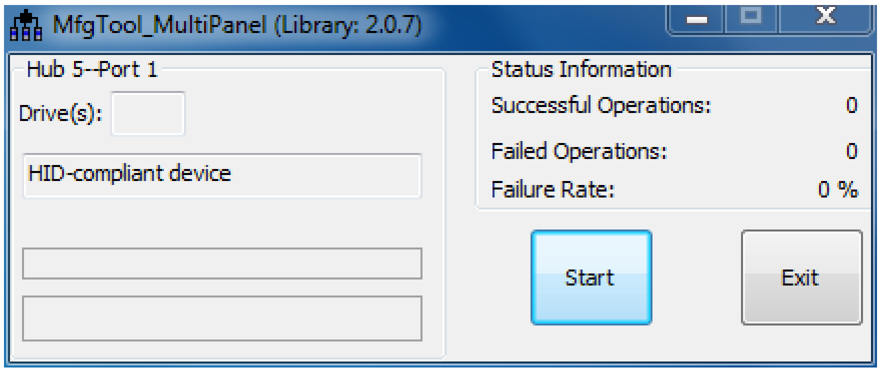
1. Decompress the utility package as a folder in <Mfgtools>, and you will find **MfgTool2.exe** in <Mfgtools>\.



2. Connect the device to your PC via a USB OTG cable. Turn on the device.



3. Run the exe file and you will see a dialog as below. Do not press Start yet.



Hub 5--Port 1

Drive(s):

HID-compliant device

Status Information

Successful Operations: 0

Failed Operations: 0

Failure Rate: 0 %

6.2 Download New Image

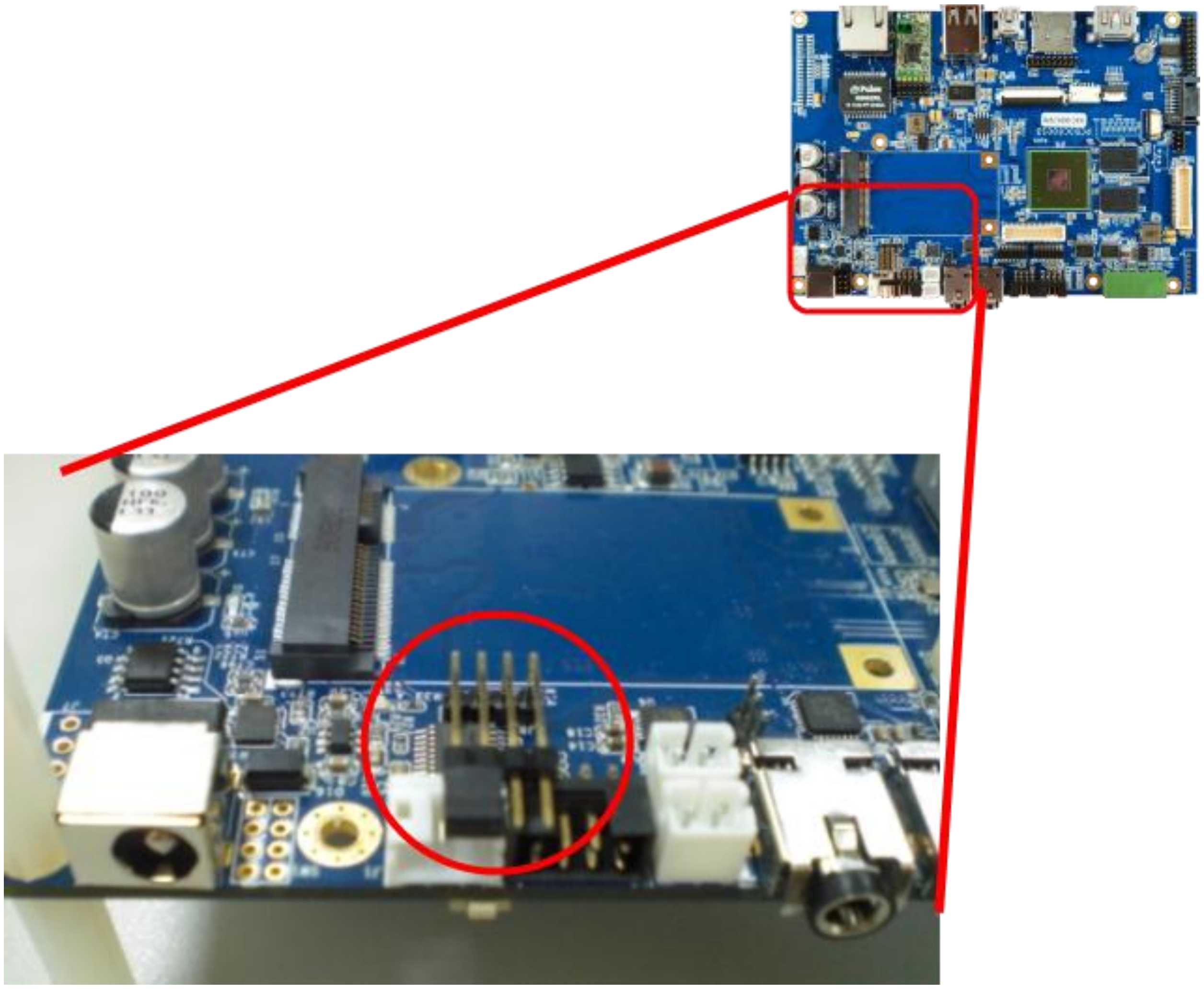
Applicable products:

OS – **Android** / Ubuntu

CPU – **Quad- / Dual- / Solo-core**

DDR size – **1GB, 2GB RAM (Quad & Dual) / 512 MB RAM (Solo)**

1. First, remove the jumpers as shown below.



2. Modify the file <Mfgtools>**cfg.ini**.

- You will need to download twice for 2 different parts: the bootloader and the Android system.
- Download **Android bootloader** for **1GB RAM (Quad & Dual) / 512 MB RAM (Solo)**:

```
[LIST]
name = c600 u-boot
```

- Download **Android bootloader** for **2GB RAM (Quad & Dual)**:

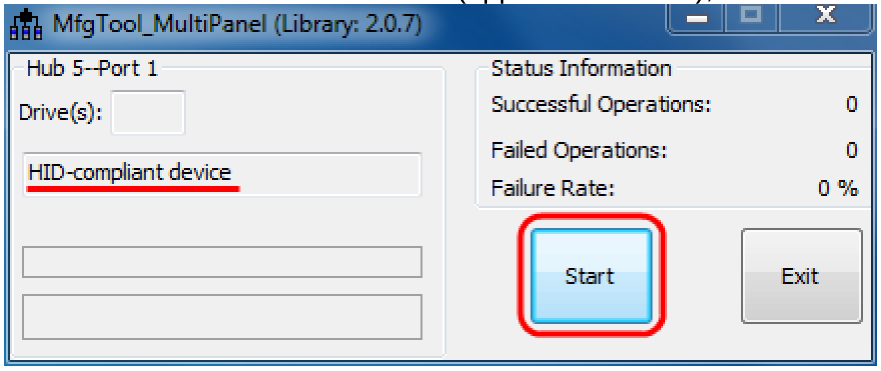
```
[LIST]
name = c6002G u-boot
```

Notes:
- If you want to boot from microSD card, put the microSD card in the slot. However, if you want to boot from eMMC, **do not** plug in the microSD card.
- Once you modify the **<Mfgtools>\cfg.ini** file, you **must** close MfgTool.exe and restart the program before using.

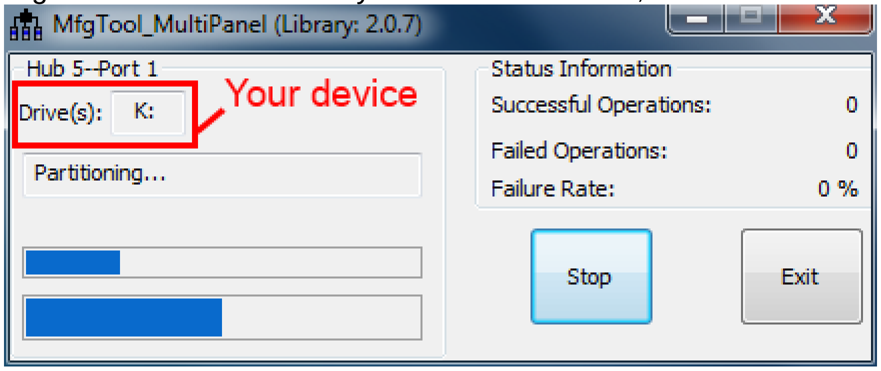
3. Put image files in the following path:

- **Android image** for **1GB RAM (Quad & Dual)**:
<Mfgtools>\Profiles\MX6**Q** Linux Update\OS Firmware\files\sabrelite\android
- **Android image** for **2GB RAM (Quad & Dual)**:
<Mfgtools>\Profiles\MX6**Q** Linux Update\OS Firmware\files\sabrelite\android_2g
- **Android image** for **512 MB RAM (Solo)**:
<Mfgtools>\Profiles\MX6**DL** Linux Update\OS Firmware\files\sabrelite\android

4. Start MfgTool2.exe, and then connect the OTG cable to the PC and power on device. When the device is connected and detected (appears as below), click **Start**.



5. Downloading will now start. If it asks you to format the disk, select ***Cancel***.



MfgTool_MultiPanel (Library: 2.0.7)

Hub 5--Port 1

Drive(s): K:

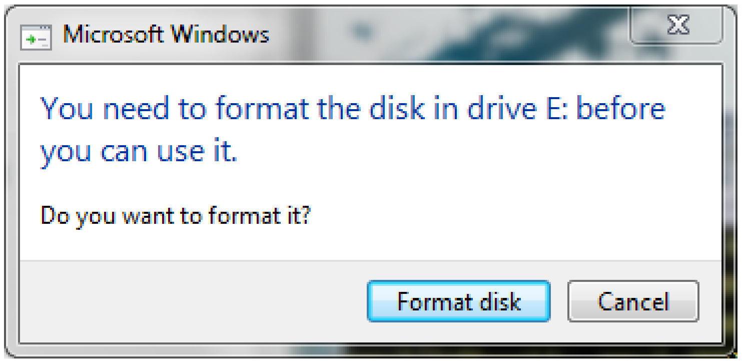
Partitioning…

Status Information

Successful Operations: 0

Failed Operations: 0

Failure Rate: 0 %

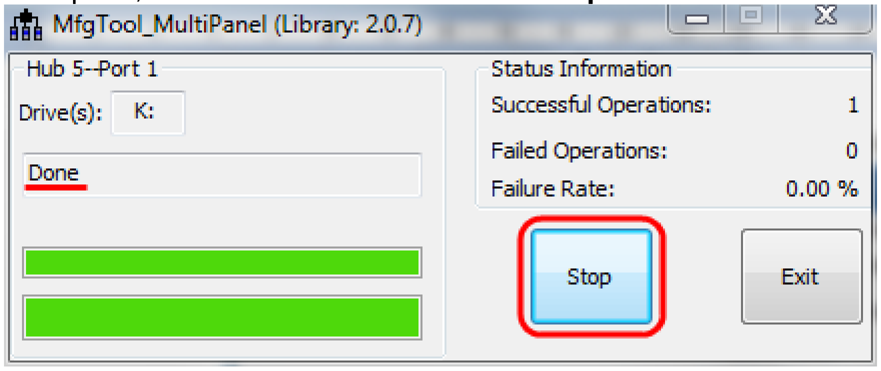


Microsoft Windows

You need to format the disk in drive E: before you can use it.

Do you want to format it?

6. When it is complete, the tool will show *Done*. Click **Stop**.



MfgTool_MultiPanel (Library: 2.0.7)

Hub 5--Port 1

Drive(s): K:

Done

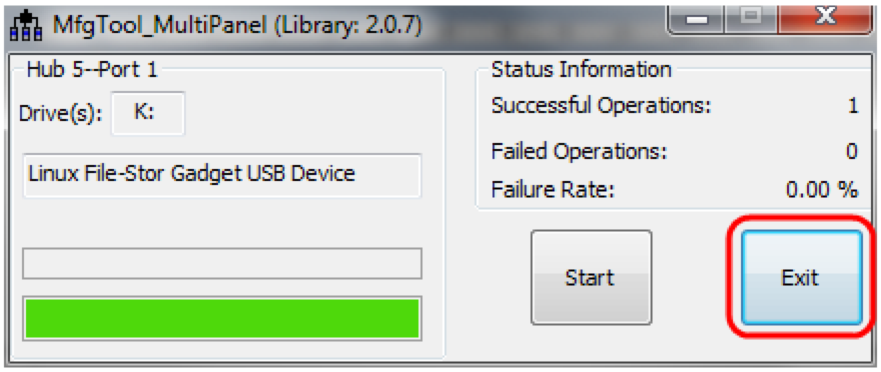
Status Information

Successful Operations: 1

Failed Operations: 0

Failure Rate: 0.00 %

7. Click **Exit** to close the program. **Turn off the device** power and then **remove the OTG cable**.



MfgTool_MultiPanel (Library: 2.0.7)

Hub 5--Port 1

Drive(s): K:

Linux File-Stor Gadget USB Device

Status Information

Successful Operations: 1

Failed Operations: 0

Failure Rate: 0.00 %

8. Open the file <Mfgtools>**\cfg.ini** again and modify as shown below:

- Download the **Android OS** for **1GB RAM (Quad & Dual) / 512 MB RAM (Solo)**:

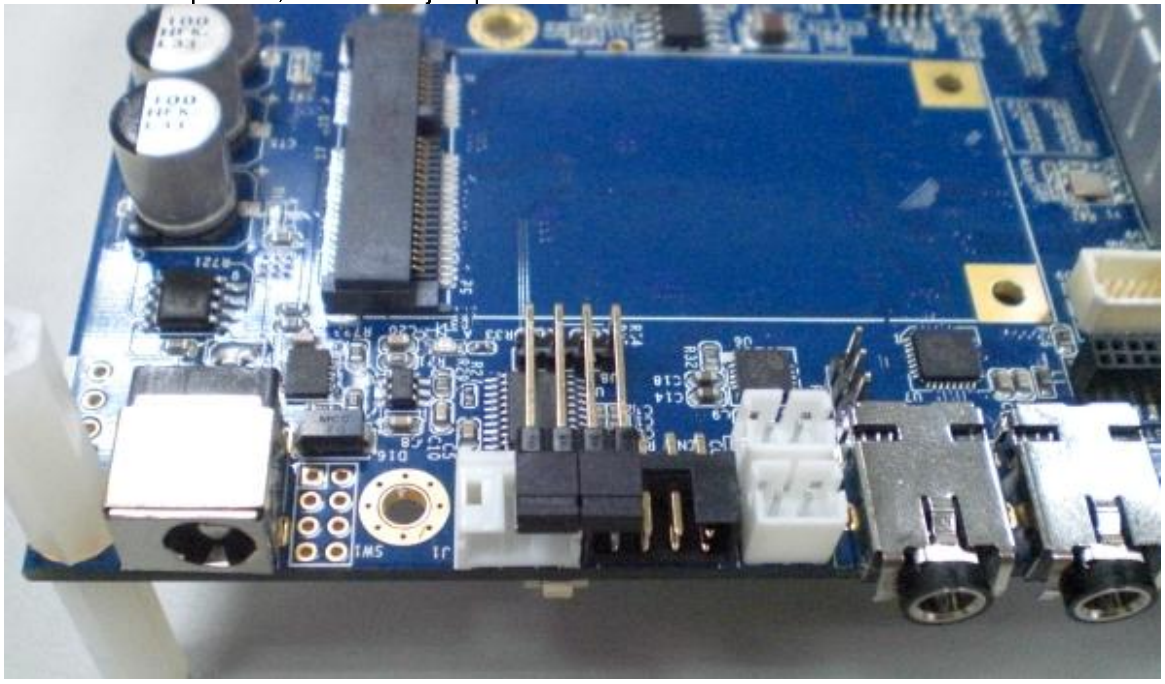
```
[LIST]
name = System Only JellyBean
```

- Download the **Android OS** for **2GB RAM (Quad & Dual)**:

```
[LIST]
name = System2G Only JellyBean
```

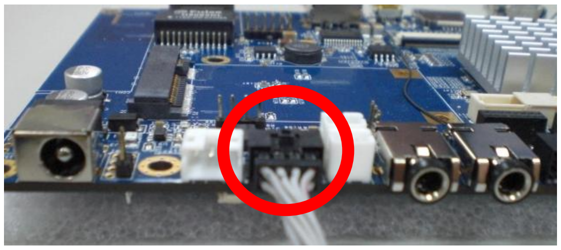
9. Repeat steps 4 to 7.

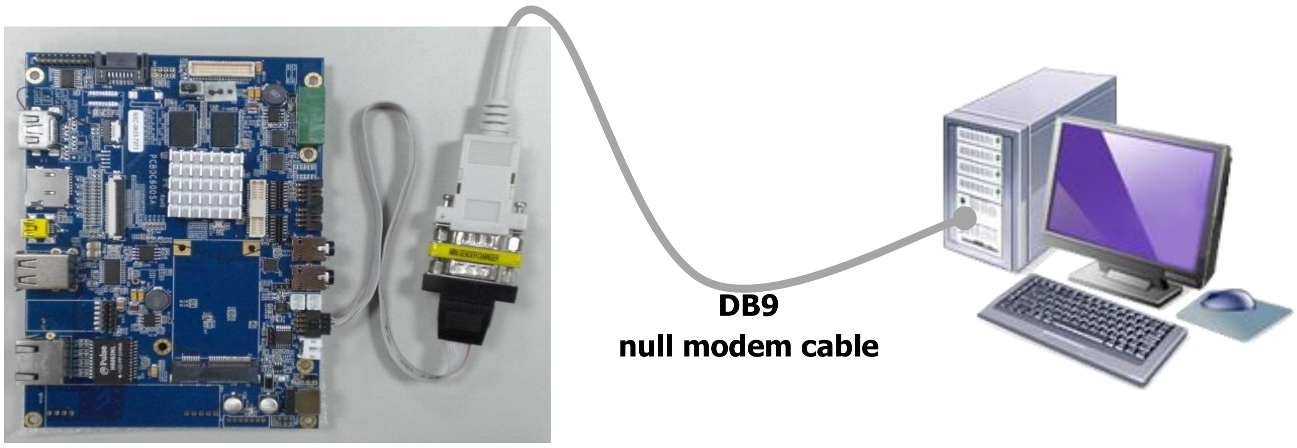
10. After completion, return the jumpers as shown below.



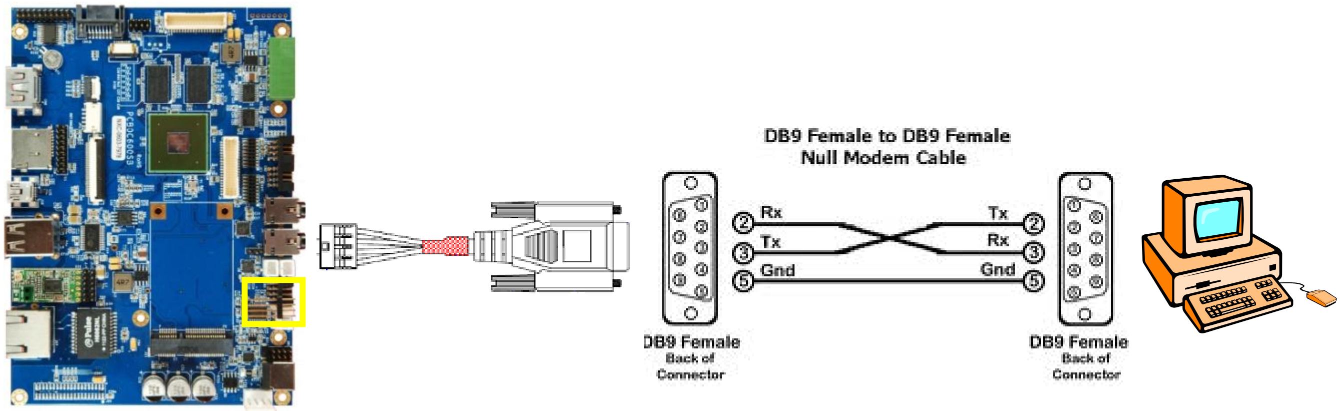
6.3 Set Your Output

If you want to change your output display, please follow the below steps.
First, connect to debug port and run the hyper terminal on PC.





Console/Debug Port Connection Diagram



UART1 is dedicated as the debug port. UART1 default settings are ***Baud Rate 115200, 8 data bits, no parity, 1 stop bit and no flow control***.

A DB9 **null modem cable** (or adapter) is required when you want to connect UART1 to a PC with terminal emulation software such as TeraTerm.
Changing the output display setting is through the U-boot environment.

Then power up the board and you will see a message from the debug port as shown below. Quickly hit **Enter** within **3** seconds.

```
          Hit any key to stop autoboot:  3
// hit Enter
          > printenv
```

Find the variable **“panel”** and modify its value for changing the output:

```
          > setenv panel 'string value'
```

The **string value** depends on what output you want (PS. There is different between Android and Ubuntu):

- **HDMI (1920 x 1080)**

```
'video=mxcfb0:dev=hdmi,1920x1080M@60,if=RGB24,bpp=32 video=mxcfb1:off cea'
```

- **7” LCD-PT (800 x 480, with RTP)**

```
'video=mxcfb0:dev=lcd,LCD-WVGA,if=RGB24,bpp=32 video=mxcfb1:off'
```

- **7” LCD-AWT (800 x 480, with RTP)**

```
'video=mxcfb0:dev=lcd,AWT-WVGA,if=RGB24,bpp=32 video=mxcfb1:off'
```

★ For other LCD / LVDS panels we support, please contact to our sales in order to get right string value.

Finally, don’t forget to save the environment. Then reset the power or type in **boot** to start.

```
          > saveenv
Saving Environment to SPI Flash...
Erasing SPI flash...Writing to SPI flash................................SUCCESS

Done
          > boot
```

Note:

- If your OS is *Android* and the display is a *resistive touch panel* (RTP), you should calibrate the touch when used for the first time. Refer to the next section **Calibrate the touch** for more details.

Chapter 7: FAQ

7.1 Calibrate the touch

If you want or need to calibrate the touch for any reason such as:

- Touch is not accurate
- You have changed output
- It is your first boot after you updated firmware and set output.

Use the commands to calibrate after devices boot.

When device power on, please notice the messages from console show **‘adb_open’**, quickly input **stop** and below command:

```
        adb_bind_config
        adb_open
        # stop
//now the system would stop and display stays at penguin icon
        # ts_calibrator
//now there would be a white ‘┼’ at left, it is calibration point
```

When the calibration point ‘┼’ shows, please quickly and accurately touch the cross on panel. Be careful that this calibrator is with timeout function. If it is timeout, just input **ts_calibrator** again.

After touching several crosses, there will appear a green and red small square. Touch the left green one a while. It is accurate confirmation function; all blue crosses should be within the green square. If not, the calibrator will let you calibrate again automatically.

Finally, reset power or keyin **start** to continue system.

```
        # start
```