Steps to install SDK

1. Firstly check whether you have a system with JDK environment or not. If not, then download the JDK from <https://cds.sun.com/is-bin/INTERSHOP.enfinity/WFS/CDS-CDS_Developer-Site/en_US/-/USD/ViewProductDetail-Start?ProductRef=jdk-6u24-oth-JPR@CDS-CDS_Developer>.

(Make sure you download the file for the OS you are working on)

1. After installing the JDK, we need to install Eclipse IDE from the following site: <http://www.eclipse.org/downloads/>.

(It is recommended that you install the Eclipse Classic 3.6.2-171MB)

1. After installing the JDK on your system, we need to install the SDK starter package from the following site: <http://developer.android.com/sdk/index.html>

Be sure to download both the files given for Windows (if working on windows).

1. After the system is SDK enabled, you need to download a plugin called the ADT (Android Development Tools) by running the Eclipse Software. The steps for the downloading and configuring the plugin are given below:

* Start Eclipse, then select **Help** > **Install New Software...**
* Click **Add**, in the top-right corner.
* In the Add Repository dialog that appears, enter "ADT Plugin" for the Name and the following URL for the Location:

<https://dl-ssl.google.com/android/eclipse/>

* Note: If you have trouble acquiring the plugin, try using "http" in the Location URL, instead of "https" (https is preferred for security reasons). Click **OK**.
* In the Available Software dialog, select the checkbox next to Developer Tools and click **Next**.
* In the next window, you'll see a list of the tools to be downloaded. Click **Next**.
* Read and accept the license agreements, then click **Finish**.
* When the installation completes, restart Eclipse.

**Configuring the ADT**

Once you've successfully downloaded ADT as described above, the next step is to modify your ADT preferences in Eclipse to point to the Android SDK directory:

* Select Window > Preferences... to open the Preferences panel (Mac OS X: Eclipse > Preferences).
* Select Android from the left panel.
* For the *SDK Location* in the main panel, click Browse... and locate your downloaded SDK directory.
* Click Apply, and then OK.
* If you are behind a firewall (such as a corporate firewall), make sure that you have properly configured your proxy settings in Eclipse. In Eclipse, you can configure proxy information from the main Eclipse menu in **Window** (on Mac OS X, **Eclipse**) > **Preferences** > **General** > **Network Connections**.

1. The last step in setting up your SDK is using the Android SDK and AVD Manager (a tool included in the SDK starter package) to download essential SDK components into your development environment.

* If you used the Windows installer, when you complete the installation wizard, it will launch the Android SDK and AVD Manager with a default set of platforms and other components selected for you to install. Simply click **Install** to accept the recommended set of components and install them.
* You can launch the Android SDK and AVD Manager in one of the following ways:
* From within Eclipse, select **Window > Android SDK and AVD Manager**.
* On Windows, double-click the SDK Manager.exe file at the root of the Android SDK directory.
* On Mac or Linux, open a terminal and navigate to the tools/ directory in the Android SDK, then execute:
* Android
* To download components, use the graphical UI of the Android SDK and AVD Manager.
* Optionally, you might want to add the location of the SDK's *tools/ and platform-tools* to your PATH environment variable, to provide easy access to the tools.

To Install A Platform

To run the Hello World application, you need to install at least one Android platform in your SDK environment. If you have not already performed this step, you need to do it now.

To install a platform in Eclipse:

1. In the Android SDK and AVD Manager, choose **Available Packages** in the left panel.
2. Click the repository site checkbox to display the components available for installation.
3. Select at least one platform to install, and click **Install Selected**. If you aren't sure which platform to install, use the latest version.

Create a AVD

In this tutorial, you will run your application in the Android Emulator. Before you can launch the emulator, you must create an Android Virtual Device (AVD). An AVD defines the system image and device settings used by the emulator.

To create an AVD:

1. In Eclipse, choose **Window > Android SDK and AVD Manager**.
2. Select **Virtual Devices** in the left panel.
3. Click **New**.

The **Create New AVD** dialog appears.

1. Type the name of the AVD, such as "my\_avd".
2. Choose a target. The target is the platform (that is, the version of the Android SDK, such as 2.1) you want to run on the emulator.

You can ignore the rest of the fields for now.

1. Click **Create AVD**.

Create an Android Project

After you've created an AVD, the next step is to start a new Android project in Eclipse.

1. From Eclipse, select **File > New > Project**.

If the ADT Plugin for Eclipse has been successfully installed, the resulting dialog should have a folder labeled "Android" which should contain "Android Project". (After you create one or more Android projects, an entry for "Android XML File" will also be available.)

1. Select "Android Project" and click **Next**.
2. Fill in the project details with the following values:

* *Project name:* HelloAndroid
* *Application name:* Hello, Android
* *Package name:* com.example.helloandroid (or your own private namespace)
* *Create Activity:* HelloAndroid,
* Click **Finish**.
* Your Android project is now ready. It should be visible in the Package Explorer on the left. Open the HelloAndroid.java file, located inside HelloAndroid > src > com.example.helloandroid). It should look like this:
* package com.example.helloandroid;  
    
  import android.app.Activity;  
  import android.os.Bundle;  
    
  public class HelloAndroid extends Activity {  
      /\*\* Called when the activity is first created. \*/  
      @Override  
      public void onCreate(Bundle savedInstanceState) {  
          super.onCreate(savedInstanceState);  
          setContentView(R.layout.main);  
      }  
  }
* Notice that the class is based on the [Activity](http://developer.android.com/reference/android/app/Activity.html) class. An Activity is a single application entity that is used to perform actions. An application may have many separate activities, but the user interacts with them one at a time. The [onCreate()](http://developer.android.com/reference/android/app/Activity.html" \l "onCreate(android.os.Bundle)) method will be called by the Android system when your Activity starts — it is where you should perform all initialization and UI setup. An activity is not required to have a user interface, but usually will.
* Now let's modify some code!

Construct The UI

Take a look at the revised code below and then make the same changes to your HelloAndroid class. The bold items are lines that have been added.

package com.example.helloandroid;  
  
import android.app.Activity;  
import android.os.Bundle;  
**import android.widget.TextView;**  
  
public class HelloAndroid extends Activity {  
   /\*\* Called when the activity is first created. \*/  
   @Override  
   public void onCreate(Bundle savedInstanceState) {  
       super.onCreate(savedInstanceState);  
       **TextView tv = new TextView(this);  
       tv.setText("Hello, Android");  
       setContentView(tv);**  
   }  
}

**Tip:** An easy way to add import packages to your project is to press **Ctrl-Shift-O** (**Cmd-Shift-O**, on Mac). This is an Eclipse shortcut that identifies missing packages based on your code and adds them for you.

An Android user interface is composed of hierarchies of objects called Views. A [View](http://developer.android.com/reference/android/view/View.html) is a drawable object used as an element in your UI layout, such as a button, image, or (in this case) a text label. Each of these objects is a subclass of the View class and the subclass that handles text is [TextView](http://developer.android.com/reference/android/widget/TextView.html).

In this change, you create a TextView with the class constructor, which accepts an Android [Context](http://developer.android.com/reference/android/content/Context.html) instance as its parameter. A Context is a handle to the system; it provides services like resolving resources, obtaining access to databases and preferences, and so on. The Activity class inherits from Context, and because your HelloAndroid class is a subclass of Activity, it is also a Context. So, you can pass this as your Context reference to the TextView.

Next, you define the text content with [setText()](http://developer.android.com/reference/android/widget/TextView.html" \l "setText(java.lang.CharSequence)).

Finally, you pass the TextView to [setContentView()](http://developer.android.com/reference/android/app/Activity.html" \l "setContentView(android.view.View)) in order to display it as the content for the Activity UI. If your Activity doesn't call this method, then no UI is present and the system will display a blank screen.

There it is — "Hello, World" in Android! The next step, of course, is to see it running.

Run the Program

The Eclipse plugin makes it easy to run your applications:

1. Select **Run > Run**.
2. Select "Android Application".

The Eclipse plugin automatically creates a new run configuration for your project and then launches the Android Emulator. Depending on your environment, the Android emulator might take several minutes to boot fully, so please be patient. When the emulator is booted, the Eclipse plugin installs your application and launches the default Activity.