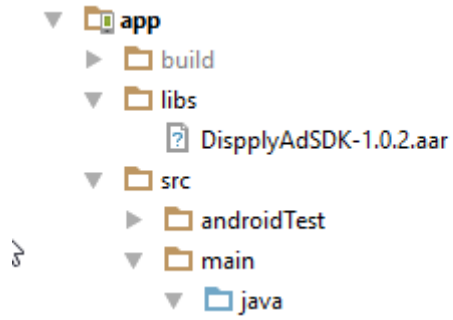
1.Add following under manifest tag to your AndroidManifest.xml:

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

2. Put the DispplyAdSDK-1.1.0.aar in “libs” folder in your Android Studio



3. Add it to dependencies in build.grandle file with “flatDir” repository. Also you need to add google play services.

```
allprojects {
    repositories {
        jcenter()
        flatDir {
            dirs 'libs'
        }
    }
}
dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:23.3.0'
    compile 'com.android.support:support-v4:23.3.0'
    compile 'com.google.android.gms:play-services-ads:8.4.0'
    compile 'com.google.android.gms:play-services-base:8.4.0'
    compile(name: 'DispplyAdSDK-1.1.0', ext: 'aar')
```

4. Then, create a function that requests a native ad:

```
private void showNativeAd() {
    adFrame = (FrameLayout) findViewById(R.id.adContent);
    nativeAd = new NativeAd(this, "YOUR_PLACEMENT_ID"); //Native AD constructor
    nativeAd.setContent("title,icon,main,description"); // Set content to load
    nativeAd.setAdListener(new AdListener() { // Add Listeners
        @Override
        public void onAdLoaded(Ad ad) { // Called when AD is Loaded

        }
        @Override
        public void onError(Ad nativeAd, String error) { // Called when load is
fail

        }
```

```
        @Override
        public void onAdClicked() { // Called when user click on AD

        }
    });
    nativeAd.loadAd(); // Call to load AD
}
```

5. The next step is to extract the ad metadata and use its properties to build your customized native UI. You can either create your custom view in a layout .xml, or you can add elements in code.

The custom layout .xml. For example:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" >
    <ImageView
        android:id="@+id/ivIcon"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true" />
    <TextView
        android:id="@+id/tvTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_toRightOf="@+id/ivIcon"
        android:paddingLeft="10dp"
        android:textSize="18sp"
        android:textStyle="bold"
        android:typeface="monospace" />
    <ImageView
        android:id="@+id/ivImage"
        android:layout_width="480dp"
        android:layout_height="168dp"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/ivIcon" />
    <TextView
        android:id="@+id/tvDescription"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/ivImage" />
</RelativeLayout>
```

Now you can use this layout .xml as a frame. For example:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
```

```
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!" />
    <FrameLayout
        android:id="@+id/adContent"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true" >
    </FrameLayout>
</RelativeLayout>
```

6. Modify the onAdLoaded function above to retrieve the ad properties. For example:

```
private NativeAd nativeAd;
private View AdView;
private FrameLayout adFrame; //FrameLayout with all views that you need

---------------------------------------------------------------------------


@Override
public void onAdLoaded(Ad ad) { // Called when AD is Loaded
    Toast.makeText(MainActivity.this, "Native ad loaded",
Toast.LENGTH_SHORT).show();
    AdView = nativeAd.getNativeAdView(ad, R.layout.native_ad_layout); //
Registering view for AD
    adFrame.addView(AdView); //Adding view to frame
    // Create native UI using the ad metadata.
    TextView tvTitle = (TextView) AdView.findViewById(R.id.tvTitle);
    TextView tvDescription = (TextView) AdView.findViewById(R.id.tvDescription);
    ImageView ivIcon = (ImageView) AdView.findViewById(R.id.ivIcon);
    ImageView ivImage = (ImageView) AdView.findViewById(R.id.ivImage);
    // Setting the Text.
    tvTitle.setText(ad.getTitle());
    tvDescription.setText(ad.getDescription());
    // Downloading and setting the ad icon.
    NativeAd.downloadAndDisplayImage(ivIcon, ad.getIcon_url());
    // Download and setting the cover image.
    NativeAd.downloadAndDisplayImage(ivImage, ad.getImage_url());
}
```

The SDK will log the impression and handle the click automatically.