Google admob ads android tutorial

[{ type: thumb-down, id: missingTheInformationINeed, label:Missing the information need },{ type: thumb-down, id: tooComplicatedTooManySteps, label:Too many complex/too many steps },{ type: thumb-down, id: outOfDate, label:Out of date },{ type: thumb-down, id: samplesCodeIssue, label:Samples/Code issue },{ type: thumb-down, id: otherDown, [{ type: thumb-up, id: easyToUnderstand, label:Easy to understand },{ type: thumb-up, id: solvedMyProblem, label:Solved my problem },{ type: thumb-up, id: otherUp, label:Other }] Banner ads occupy a seat at the top of the application layout or at the top of the device screen. They remain on the screen while users interact with the app and can automatically update after a while. If you're new to mobile ads, they are a great place to start. Case study. This guide describes how to integrate banner ads from AdMob into the Android app. In addition to code snippets and instructions, it also provides information about the size of resource bands and links to additional resources. Prerequisites Import the Google Ads Mobile Ads SDK yourself or as part of Firebase. Add AdView to the layout The first step toward displaying the banner is to put AdView in the activity or snippet layout where you want to display it. The easiest way to do this is to add it to the corresponding XML layout file. Here's an example showing the activity of AdView: #main_activity.xml... &com.google.android.gms.ads.AdView xmlns:ads= android:id=@+id/adView android:layout_width=wrap_content android:layout_height=wrap_content android:layout_centerhorizontal=true android:layout_alignparentbottom=true ads:adsize=BANNER ads:adunitid=ca-app-pub-3940256099942544/6300978111> </com.google.android.gms.ads.AdView> I don't know Note the following attributes: ads:adSize – Set the ad size you want to use. If you don't want to use a standard size defined by a constant, you can set a custom size. For more information, see the banner size section below. Ads:adUnitId – Set a unique identifier for the app's ad unit that will show your ads. If you show banner ads in different activities, each of them will require an ad unit. You can also programmatically create AdView: AdView adView = new AdView (this); adView.setAdSize (AdSize.BANNER); adView.setAdUnitId(ca-app-pub-3940256099942544/6300978111); TODO: Add adView to the view hierarchy. val adView = AdView (this) adView.adSize = AdSize.BANNER adView.adUnitId =ca-app-pub-3940256099942544/6300978111 // TODO: Add adView to your view hierarchy. Warning: Make sure you've set your ad size and ad unit ID in the same way (that is, set up XML or both programmatically). When and test your apps, make sure you're using test ads instead of direct production ads. If you don't do this, your account may be suspended. The easiest way to load pilot ads is to use our special test ad unit ID for Android banners: ca-app-pub-3940256099942544/6300978111 It has been specifically configured to return test ads for each application, and you can use it in your apps during coding, testing and debugging. Before you publish an app, make sure you change your ad device ID. For more information about how Mobile Ads SDK pilot ads work, see Test Ads. Upload your ad note: Make all calls to the Mobile Ads SDK's main thread. Once AdView is in place, the next step will be to load the ad. This is done by using the loadAd() method in the AdView class. It takes an AdRequest parameter that contains runtime information (such as targeting information) for a single ad request. Here's an example that shows how to load an ad onCreate() method activity: MainActivity (excerpt) package... Import... import com.google.android.gms.ads.AdRequest; import com.google.android.gms.ads.AdView; public class MainActivity extends AppCompatActivity { private AdView mAdView; protected undo onCreate (Bundle savedInstanceState) { super.onCreate (savedInstanceState); setContentView (R.layout.activity_main); MobileAds.initialize(this is, new OnInitializationCompleteListener() { @Override public undo onInitializationComplete(InitializationStatus initializationStatus) { }); mAdView =findViewById(R.id.adView); AdRequest adRequest = new AdRequest.Builder().build(); mAdView.loadAd(adRequest); } } package ... Import... import com.google.android.gms.ads.AdRequest import com.google.android.gms.ads.AdView class MainActivity: AppCompatActivity() { lateinit var mAdView: AdView ignore funCreate (savedInstanceState: Bundle?) { super.onCreate(savedInstanceState) setContentView(R.layout.activity_main) MobileAds.initialize(this) {} mAdView=findViewById(R.id.adView) hr adRequest = AdRequest.Builder().build() mAdView.loadAd(adRequest) } } } Note: If your ad can't load, you don't need to explicitly ask for another one if you've configured your ad device to update; The Google Mobile Ads SDK respects all the refresh rates that you specified in the AdMob user interface. If you haven't turned on the update, you'll need to request a new one. That's it! The app is ready to show banner ads. Ad events To further customize the ad's behavior, you can hook up to these ad lifecycle events: loading, opening, closing, and so on. To use AdListener with AdView, call the setAdListener() method: mAdView.setAdListener(new AdListener() { @Override public void { // Code to run when the ad finishes uploading. } @Override public void onAdFailedToLoad(LoadAdError adError) { // Code to be when the ad request fails. } @Override undo onAdOpened() { // Code to be executed when the ad opens an overlay that // includes the screen. } @Override undo onAdClicked() { // Code to be executed when the user clicks the ad. } @Override public void onAdLeftApplication() { // Code to be executed when the user has left the application. } @Override public void onAdClosed() { // Code, which will be executed when the user returns soon // to the application when the ad is tapping} }); mAdView.adListener = Object: AdListener() { Override fun onAdLoaded() { // The code must be executed when the ad finishes loading. } ignore the fun onAdFailedToLoad(adError: LoadAdError) { // Code to be executed when the ad request fails. } override the fun onAdOpened() { // Code must be executed when the ad opens an overlay that // includes the screen. } ignore the fun onAdClicked() { // Code must be executed, when a user clicks on an ad. } override the fun onAdLeftApplication() { // Code to be executed when the user has left the application. } override the fun onAdClosed() { // Code that will run when the user returns // to the app after tapping the ad. } } Each AdListener overwritable method matches the ad's lifecycle event. The OnAdLoaded() method is ignored when the ad completes upload. For example, if you want to delay adding AdView to an activity or snippet until you're sure your ad will load, you can do so here. The onAdFailedToLoad() OnAdFailedToLoad() method is the only one that contains the parameter. The LoadAdError error parameter describes an error. For more information, see the ad load error matching documentation. onAdOpened() This method is called when a user touches an ad onAdLeftApplication() This method is called after onAdOpened() when another application (e.g. Google Play) is opened at the user's click, and the current app is supported. onAdClosed() When a user returns to the app after viewing the ad's destination URL, this method is called. The program can use it to resume a suspended activity or to resume any other work that is required to prepare it for interaction. For the introduction of ad listener methods in the Android API demo, see the AdMob AdListener example. Banner sizes Note: The size of the container you placed your ad in must be at least equal to the banner. If your container has a padding, it effectively reduces the size of your container. If the container can't fit your banner ad, the banner won't appear, and you'll receive this warning in the journals: W/Ads: Not enough space to show your ad. Requires 320x50 dp, but only 288x495 dp. The table below lists the standard banners Size dp (WxH) Description Accessibility AdSize Constant 320x50 Banner Phones and Tablets BANNER 320x100 Large Banner Phones and Tablets LARGE_BANNER 300x250 IAB Average Average Phones and tablets MEDIUM_RECTANGLE 468x60 IAB full-size banner tablets FULL_BANNER 728x90 IAB leaderboard tablets LEADERBOARD Supplied width x Adaptive height Adaptive Phones and Tablets N/A Screen width x 32|50|90 Smart banner Phones and Tablets SMART_BANNER Learn more about adaptive banners to replace smart banners. To set the custom banner size, set the AdSize you want, as shown here: AdSize adSize = new AdSize (300, 50); val adSize = AdSize(300, 50) For video ads to be successful in banner ad views, hardware acceleration must be enabled. Hardware acceleration is enabled by default, but some programs may choose to turn it off. If this applies to an app, we recommend that you enable hardware acceleration for business classes that use ads. Enabling hardware acceleration If a program doesn't work properly when hardware acceleration is turned on worldwide, you can also manage it for individual activities. To turn hardware acceleration on or off, you can use the Android:hardwareAccelerated attribute <application> <activity>and AndroidManifest.xml items. The following example enables hardware acceleration for an entire program, but disables it for one activity: <application android:hardwareaccelerated=true> <!-- For activities that use ads, hardwareAcceleration should be true. --> <activity android:hardwareaccelerated=true></activity> <!-- For activities that don't use ads, hardwareAcceleration can be false. --> <activity android:hardwareaccelerated=false></activity> </application> for more information about hardware acceleration control options, see the HW Acceleration Guide. Note that individual ad views can't be enabled for hardware acceleration if the activity is turned off, so hardware acceleration must be enabled for the activity itself. Additional Resources Examples GitHub Minimum Implementation banner advertising example: Java | Kotlin Advanced Features demo: Java | Kotlin Banner RecyclerView flagship program: Java Mobile Ads Garage video tutorials Banner Implementation Banner best practice success stories</activity> </application>