Native ads are loaded via the Adloader class, which has its own Builder class to customize it during creation. By adding listeners to the AdLoader while building it, an app specifies which types of native ads it is ready to receive. The AdLoader then requests just those types.

### **Build an AdLoader**

val adLoader = AdLoader.Builder(this, "ca-app-pub-3940256099942544/2247696110")

.forNativeAd { ad : NativeAd ->

// Show the ad.

}

.withAdListener(object : AdListener() {

override fun onAdFailedToLoad(adError: LoadAdError) {

// Handle the failure by logging, altering the UI, and so on.

}

})

.withNativeAdOptions(NativeAdOptions.Builder()

// Methods in the NativeAdOptions.Builder class can be

// used here to specify individual options settings.

.build())

.build()

#### **repare for the NativeAd format**

The first method above is responsible for preparing the AdLoader for the NativeAd format:

For Nativead:

Calling this method configures the AdLoader to request native ads. When an ad has loaded successfully, the listener object's onNativeAdLoaded() method is called.

When the AdLoader makes an ad request, Google selects and returns the ad that maximizes publisher yield.

### **Use AdListener with an AdLoader**

During creation of the AdLoader above, the withAdlistner function sets an Adlistner This is an optional step. The method takes an AdListener as its lone parameter, which receives callbacks from the AdLoader when ad lifecycle events take place:

.withAdListener(object : AdListener() {

// AdListener callbacks can be overridden here.

})

### **Loading ads**

Once you've finished building an AdLoader, it's time to use it to load ads. There are two methods available for this: loadAd() and loadAds().

adLoader.loadAd(AdRequest.Builder().build())

The loadAds() method sends a request for multiple ads (up to 5):

adLoader.loadAds(AdRequest.Builder().build(), 3)

Both of these methods take a Adrequest object as their first parameter. This is the same AdRequest class used by banners and interstitials, and you can use methods of the AdRequest class to add targeting information, just as you would with other ad formats.

loadAds() takes an additional parameter: the number of ads the SDK should attempt to load for the request. This number is capped at a maximum of five, and it's not guaranteed that the SDK will return the exact number of ads requested. If multiple ads are returned by a call to loadAds(), they will be different from each other.

After a call to loadAd(), a single callback will be made to the listener methods defined above to deliver the native ad object or report an error.

After a call to loadAds(), multiple such callbacks will be made (at least one, and no more than the number of ads requested). Apps requesting multiple ads should call AdLoader.isLoading() in their callback implementations to determine whether the loading process has finished.

Here's an example showing how to check isLoading() in the onNativeAdLoaded() callback:

lateinit var adLoader: AdLoader

...

adLoader = AdLoader.Builder(this, "ca-app-pub-3940256099942544/2247696110")

.forNativeAd {

...

// some code that displays the ad.

...

**if (adLoader.isLoading) {**

**// The AdLoader is still loading ads.**

**// Expect more adLoaded or onAdFailedToLoad callbacks.**

**} else {**

**// The AdLoader has finished loading ads.**

**}**

}.build()

adLoader.loadAds(AdRequest.Builder().build(), 3)

### **Cleaning up**

Be sure to use the destroy() method on loaded native ads. This frees up utilized resources and prevents memory leaks.

**Key Point:** **destroy()** must be called on **all ads**, even if they weren't used or referenced.

Ensure that all NativeAd references are destroyed in your activity's onDestroy() method.

In your onNativeAdLoaded callback, make sure to destroy any existing native ads that will be dereferenced.

Another key check is if the activity is destroyed and if so, call destroy() on the returned ad and return immediately:

lateinit var adLoader: AdLoader

...

adLoader = AdLoader.Builder(this, "ca-app-pub-3940256099942544/2247696110")

.forNativeAd { nativeAd ->

**// If this callback occurs after the activity is destroyed, you**

**// must call destroy and return or you may get a memory leak.**

**// Note `isDestroyed` is a method on Activity.**

**if (isDestroyed) {**

**nativeAd.destroy()**

**return@forNativeAd**

**}**

...

}.build()