Importing the Bindings Project

NOTE: This is for the deprecated Admob bindings because I haven't seen any glitches from them. The RoboPod for Admob is easier to setup but I've had problems with it so I'm using the tried and true bindings.

Go to https://github.com/BlueRiverInteractive/robovm-ios-bindings and to find the bindings.

First I downloaded the entire repo as a zip, not necessary if you can get a hold of just the bindings that you want. To import into Eclipse, go to File > Import > Gradle/Gradle Project and browse to the folder that contains the ENTIRE bindings folder you downloaded from the repo. Don't go INTO the bindings folder because the project will lack some things you need if you do this. E.g. go to the folder where the top-level of the admob bindings folder is stored. Don't go into any of the admob folders looking for the project. I originally thought that the admob folder contained the folder for the admob bindings project, but the whole thing IS the project. For me, I browsed to the folder containing all the bindings I downloaded from the repo (I extracted them into this folder of course). Now click Build Model and wait for your projects to show up. When the project(s) show up in the dialog, select which ones you want and click Finish. This properly imports the bindings project into Eclipse so you can use it with your LibGDX project. I know for some people this is all obvious but I kept getting stuck on the one-line instruction "Import the project of the binding you want into Eclipse".

Next you want to add it to the build path of your LibGDX project. To do this, right click on your -ios project folder and go to Build Path > Configure Build Path. Then click the Projects tab and click the Add button. Select the newly imported bindings project from the list and hit OK. Now you can access the bindings code from your iOS project. NOTE: I don't know how to add this to the build path using Gradle, so each time you refresh the Gradle dependencies you will have to re-add the bindings to the build path. Not efficient but that's the only way I know how to make it work.

The rest of the instructions on the repo are fairly straight-forward even to beginners. Note that when copying the <libs> from the bindings robovm.xml to your project's robovm.xml, you will need to change the path to the lib in order for Eclipse to find it. I only had to move 1 lib for admob so I just copied it over into my ios project and referenced it by file name only. Again probably not the best way to do it but it works.

Adding the Ad Code

Now your project should be setup with the Admob bindings and you can start working on code. Get IOSLauncher.java, AdController.java, and AdHelper.java from my repo at https://github.com/MrPat/Admob-iOS. Make sure to put AdController.java and AdHelper.java into your core project (drag and drop does this on Eclipse). Then copy my IOSLauncher code and paste it into yours, or however you want to handle that. I tried to comment things that aren't obvious. Replace the placeholder Ad IDs with your own Admob Ids, and change the package names of the files to match your package name.

You'll really want to be running this on the latest stable versions of LibGDX and RoboVM. At least that's what I'm doing.

Now, you just need to provide a constructor in your main Gdx class that takes an AdController as an argument. You'll also need to provide a blank constructor after making an explicit one. These should look something like this:

```
[illegible]
        [illegible]
        [illegible]
[illegible]

[illegible]
        [illegible]
[illegible]
```

Showing Ads

Now you're ready to show ads. Just call AdHelper.showBanner() from your main class when you want to show the banner. Note that I don't have any code to hide the banner because I never hide it. That should be available in another tutorial though. For the interstitial, call AdHelper.loadInterstitial() when you are ready to start loading. You can check AdHelper.isInterstitialLoaded() and AdHelper.isInterstitialFailed() to see the status of the loading. Once you're ready to show it, just call AdHelper.showInterstitial().