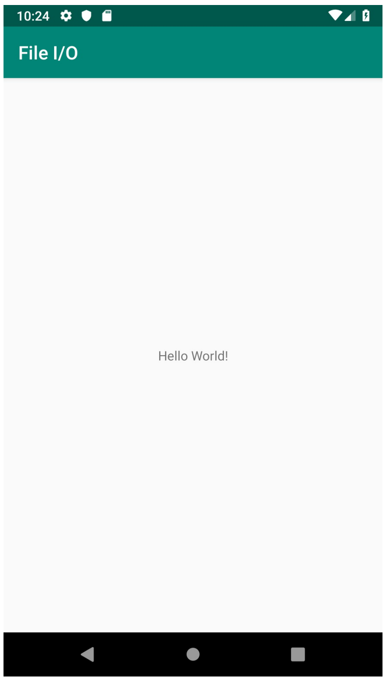
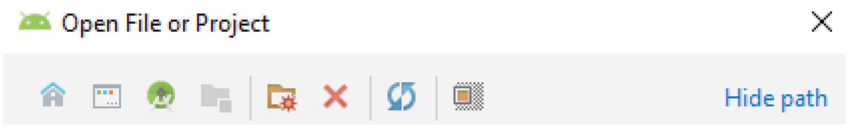
November 28th, a reminder to students that the labs are only taking place in Kevin Street i.e. rooms 1-015 or 3-006.

Last time we created an app that used AsyncTask to run a thread that faked a download task and updated a **progress bar** on the home screen. This week we will not really be creating an App i.e. the UI is the default “Hello World!” screen. Instead, we will be examining the different ways to write files to the Android File System.



Project Set Up

Unzip the MyEightApp.zip file to the folder where you normally keep your lab work. Then navigate to this folder in Android Studio using file -> open.



Hit ok to open the project in Android Studio. Should you have any difficulty with this, please **just ask your lab assistant**.

IMPORTANT: the method writeOutput can be called to do the actual writing of the data to files.

Internal Storage

Part 1 Fill in the code for the method writeToInternalPrivate

1. This method should first log the absolute path of your app's default internal private directory.
2. Save the text in `POEM` to a file in this directory.

When you are ready to **test your method** you can uncomment the call to writeToInternalPrivate in onCreate and run the app. Use the Device File Manager (or adb) to confirm that the file was successfully written to the Emulator or mobile device.

Part 2 Fill in the code for the method writeToCache

1. This method should first log the absolute path of your app's default internal cache directory.
2. Then it should make a sub-directory within this directory (Slides 36 and 38 show the syntax for creating a sub-directory)
3. If this is successful save the text in `CONTENT` to a file in this sub-directory.

When you are ready to **test your method** you can uncomment the call to writeToCache in onCreate and run the app. Use the Device File Manager (or adb) to confirm that the file was successfully written to the Emulator or mobile device.

External Storage

Part 3 Fill in the code for the method writeToPrivateExternal

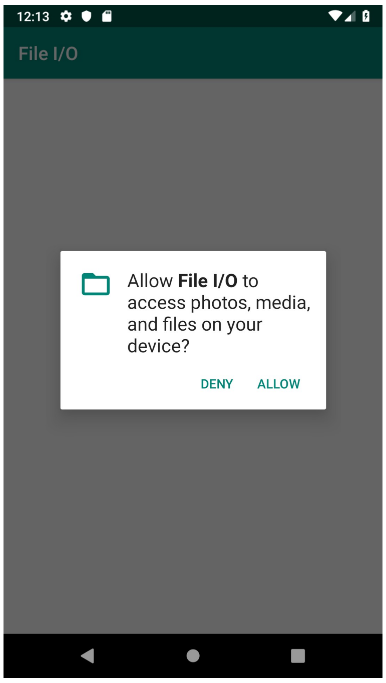
1. This method should first log the absolute path of your app's external private DOCUMENTS directory.
2. Then save the text in `LYRIC` to a file in this directory.
3. The last line in writeToPrivateExternal should call the method **readInput** to read the file back in and log the String contents to logcat.

When you are ready to **test your method** you can uncomment the call to writeToPrivateExternal in onCreate and run the app. P.T.O. ->->->

Part 4 Fill in the code for the method writeToPublicSD

1. As we are performing an operation (writing to public external storage) that has the potential to affect the user's stored data or the operation of other apps, it is first necessary to ask for write permissions to external storage in the **manifest file**.
2. The method itself should first check that external storage is available i.e. media is mounted.
3. If this is true it should log the absolute path of your device's external public DOWNLOADS directory.
4. Then save the text in `QUOTE` to a file in this directory.

When you are ready to **test your method** you can uncomment the call to getPermission in onCreate and run the app. The getPermission method provides a second layer of protection by showing a dialog to the user to confirm that they are cool about your app accessing their files. If the user consents by hitting ALLOW, then the code is set up to call writeToPublicSD as part of a system callback.


12:13

File I/O

Allow **File I/O** to access photos, media, and files on your device?