JÖNKÖPING UNIVERSITY

School of Engineering

ANDROID FILE SYSTEM

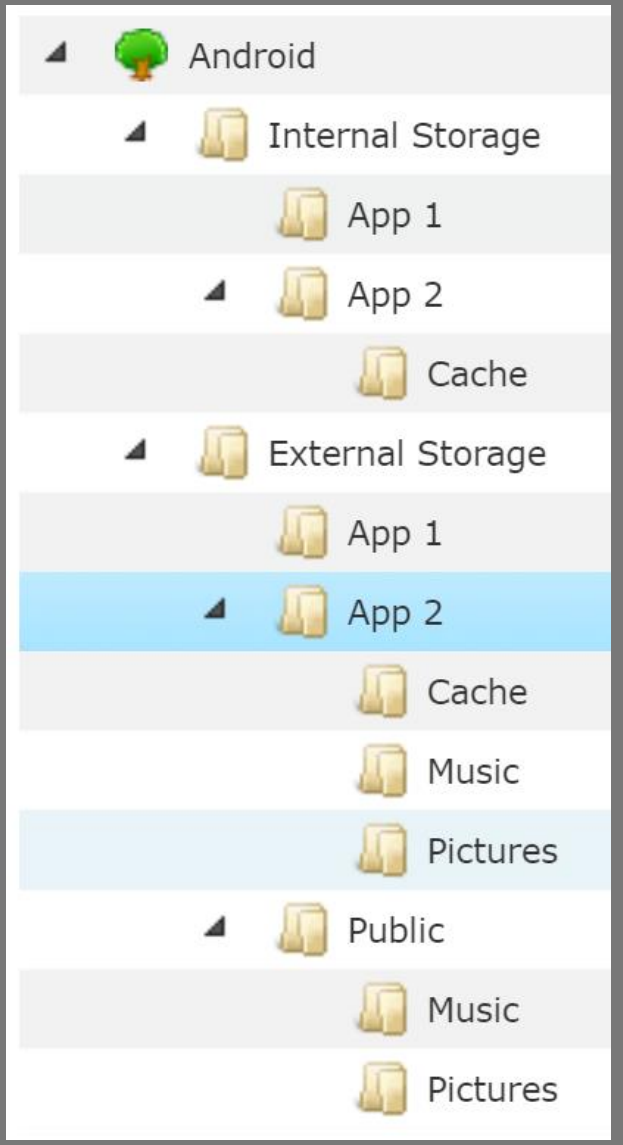
Peter Larsson-Green

Jönköping University

Spring 2021

THE FILE SYSTEM

- Consists of files and folders.



INTERNAL STORAGE

Where all apps are installed.

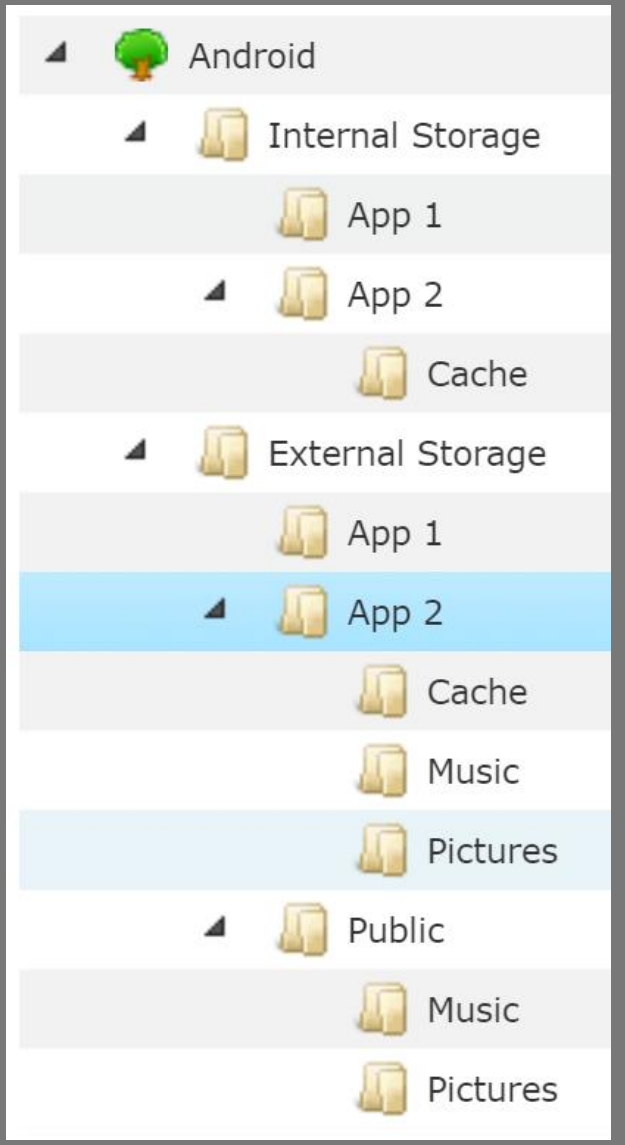
- Each app has its own folder on internal storage.
  - An app can't access other apps' app folder.

```
val folder: File = aContext.filesDir
```

- Each app has its own cache folder on internal storage.
  - The OS can delete these files if low on memory.

```
val folder: File = aContext.cacheDir
```

- (apps can be installed on external storage from API level 8)
  - https://developer.android.com/guide/topics/data/install-location.html
  - Typically used by large games.
  - If external storage not available, app can't run.



EXTERNAL STORAGE

Where apps can share files.

- Is not always available.
  - For example, if it's on a removable SD card.
- Each app has its own folder on external storage.

```
val folder: File = aContext.getExternalFilesDir(null)
```

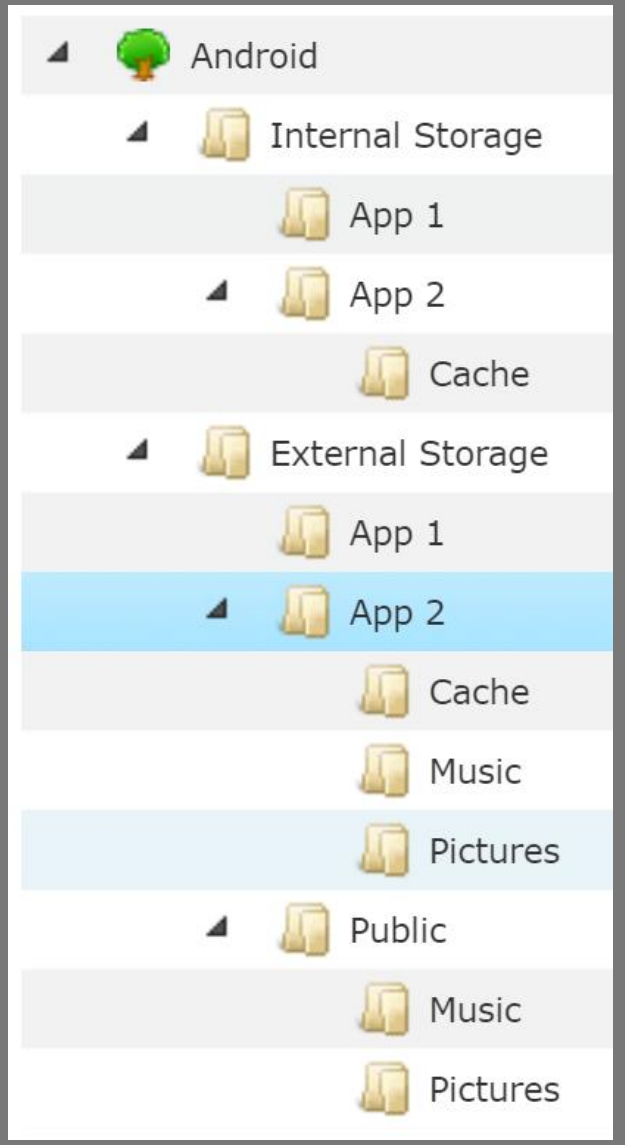
```
val folder: File = aContext.getExternalFilesDir(
  Environment.DIRECTORY_MUSIC
)
```

- Each app has its own cache folder on external storage.

```
val folder: File = aContext.externalCacheDir
```

- Each type of media has its own folder on external storage.

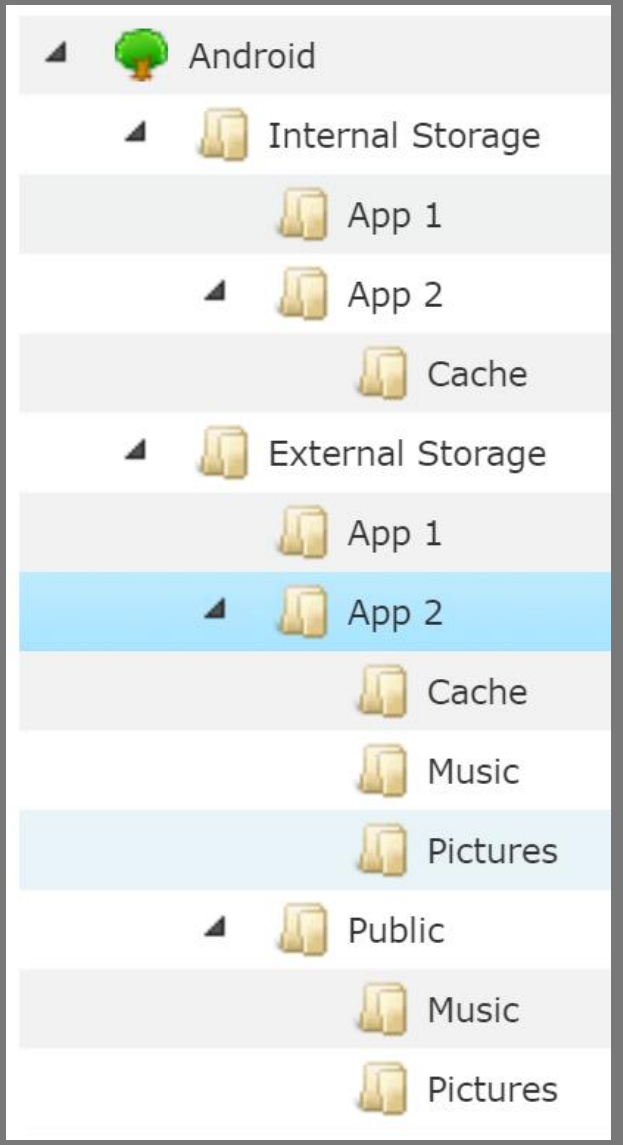
```
val folder: File = Environment.getExternalStoragePublicDirectory(
  Environment.DIRECTORY_DCIM
)
```



PERMISSIONS

- Internal Storage:
  - Each app has permission to access its own folder.
  - An app can't access another app's folder.
- External Storage:

API level	READ_EXTERNAL_STORAGE	WRITE_EXTERNAL_STORAGE
1	Didn't exist	Didn't exist
4		Required for all external storage
16	Introduced	
19	Enforced. Not required for own app folder	Not required for own app folder
29	Scoped storage introduced = Can't read/write to other app folders	
30	Scoped storage enforced	




```
val folder: File = Environment.getExternalStoragePublicDirectory(
  Environment.DIRECTORY_DCIM
)
```

WRITE TO A FILE

```
val folder = aContext.filesDir
val file = File(folder, "my-file.txt")
file.writeText("The content.")
```

READ FROM A FILE

```
val folder = aContext.filesDir
val file = File(folder, "my-file.txt")
val content = file.readText()
```

STORING USER FILES

- Use the Media Store.
  - For media files (images, videos, audio, ...)
  - Introduced in API level 1
- Use the Storage Access Framework.
  - For other type of files
  - Introduced in API level 19

MEDIA STORE

Provides a Content Provider with access to the files.

- Contract: https://developer.android.com/reference/kotlin/android/provider/MediaStore
- For Android <= 9:
  - `READ_EXTERNAL_STORAGE` for reading any file.
  - `WRITE_EXTERNAL_STORAGE` for writing any file.
- For Android >= 10:
  - `READ_EXTERNAL_STORAGE` for reading files added by other apps.

STORAGE ACCESS FRAMEWORK

Your app needs no general read/write permission,
the user selects file for us.

1. Ask the user to pick the place through the storage access framework.
2. Obtain a Content Provider URI with permission.

SECONDARY EXTERNAL STORAGE

API level 19 started support for secondary external storage.

- E.g., the device's main storage memory contains both internal and external storage, and then supports SD card too.
  - https://developer.android.com/about/versions/android-4.4#ExternalStorage

```
val folders: Array<File> = aContext.externalFilesDirs
```

```
val folders: Array<File> = aContext.externalCacheDirs
```

SHARED PREFERENCES

Key/value pairs of String/primitive data types stored in files.

- Inside an activity (Activity specific):

```
SharedPreferences preferences = getPreferences(MODE_PRIVATE);
```

- Inside an activity (Activity independent):

```
SharedPreferences preferences = getSharedPreferences(
  "the-name",
  MODE_PRIVATE
);
```

Older versions of Android supported different modes.

SHARED PREFERENCES

Write:

```
SharedPreferences.Editor editor = preferences.edit();
editor.putInt("luckyNumber", 7);
editor.putString("name", "Hello");
editor.apply(); // Save changes asynchronously or...
editor.commit(); // ...save changes synchronously.
```

Read:

```
int luckyNumber = preferences.getInt("luckyNumber", -1);
String name = preferences.getString("name", "???");
```