Data Storage (II)

Android Storage Racap

- Files
  - App-specific storage
    - Internal
    - External
  - Shared storage
    - Media & Doc
- Preferences
- Databases

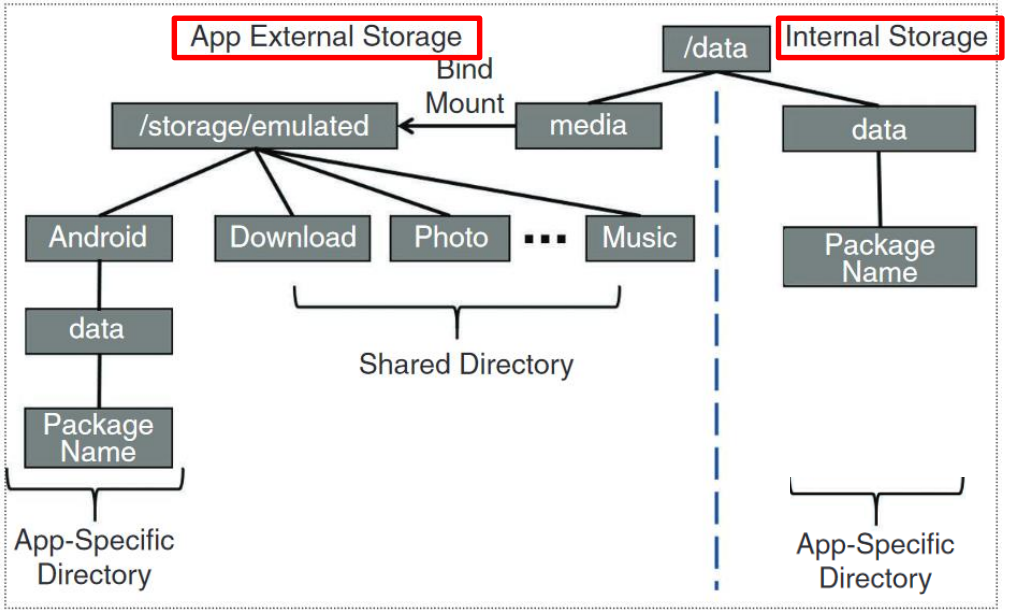


Image adapted from:
IEEE Security & Privacy
Demystifying Android's Scoped Storage Defense. Sept.-Oct. 2021, pp. 16-25, vol. 19
DOI Bookmark: 10.1109/MSEC.2021.3090564

Android Storage Racap

- Files
  - App-specific storage
    - Internal
    - External
  - Shared storage
    - Media & Doc
- Preferences
- Databases

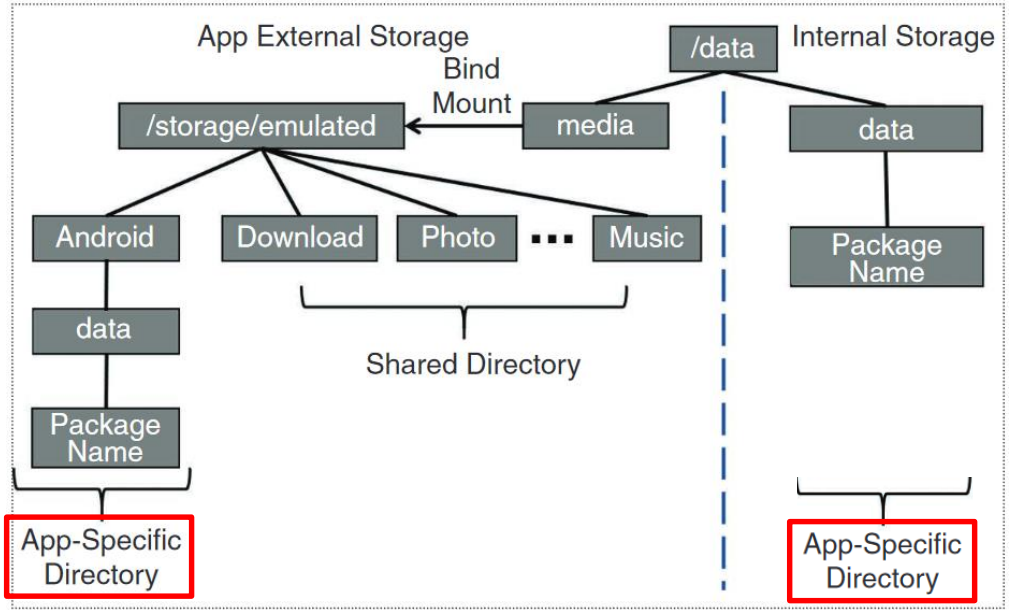


Image adapted from:
IEEE Security & Privacy
Demystifying Android's Scoped Storage Defense. Sept.-Oct. 2021, pp. 16-25, vol. 19
DOI Bookmark: 10.1109/MSEC.2021.3090564

Android Storage Racap

- Files
  - App-specific storage
    - Internal
    - External
  - Shared storage
    - Media & Doc
- Preferences
- Databases

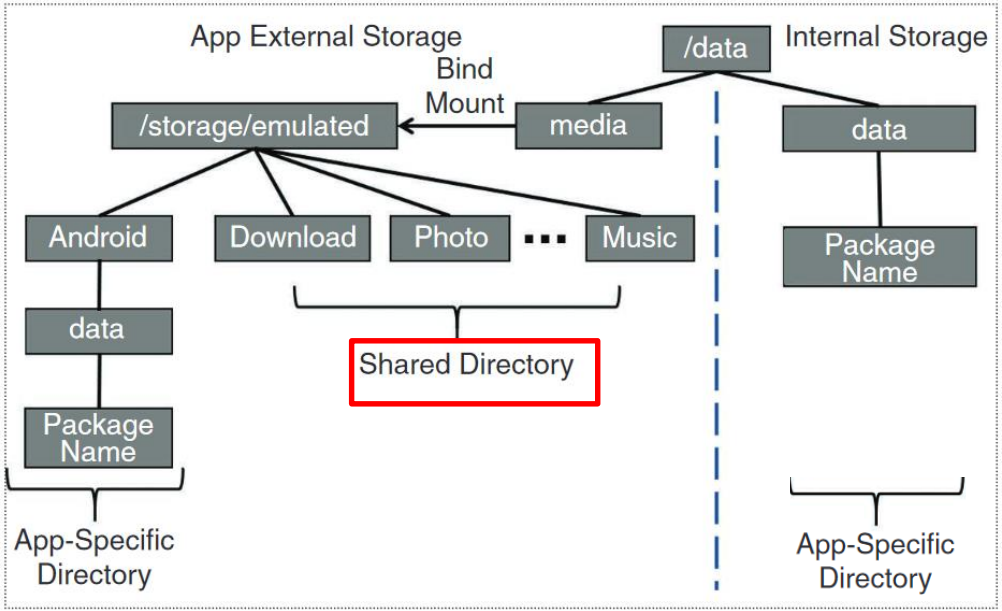


Image adapted from:
IEEE Security & Privacy
Demystifying Android's Scoped Storage Defense. Sept.-Oct. 2021, pp. 16-25, vol. 19
DOI Bookmark: 10.1109/MSEC.2021.3090564

Android Storage Racap

- Files
  - App-specific storage
    - Internal
    - External
  - Shared storage
    - Media & Doc
- Preferences
- Databases

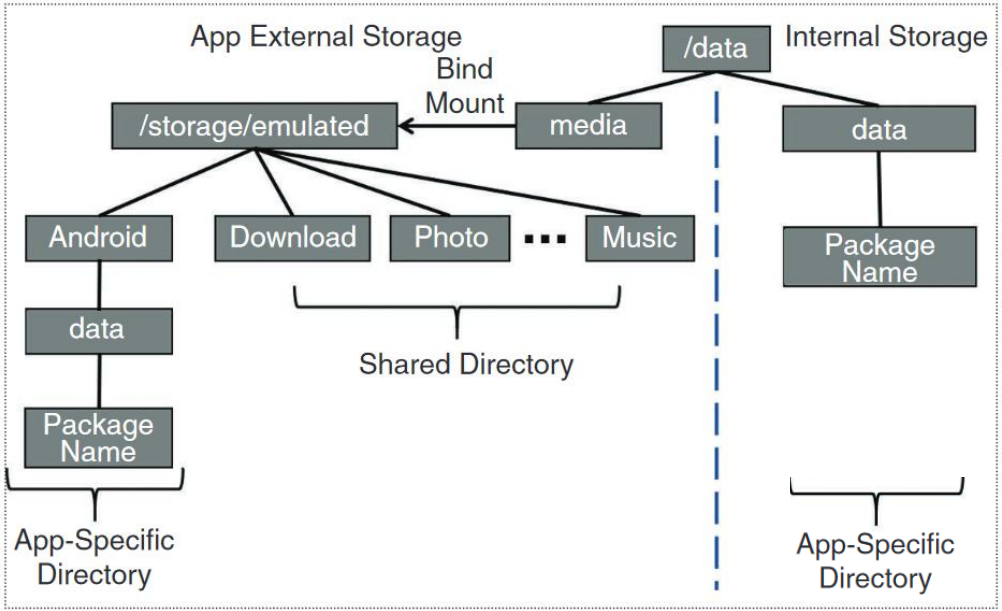


Image adapted from:
IEEE Security & Privacy
Demystifying Android's Scoped Storage Defense. Sept.-Oct. 2021, pp. 16-25, vol. 19
DOI Bookmark: 10.1109/MSEC.2021.3090564

Android Storage Racap

- Files
  - App-specific storage
    - Internal
    - External
  - Shared storage
    - Media & Doc
- Preferences
- Databases

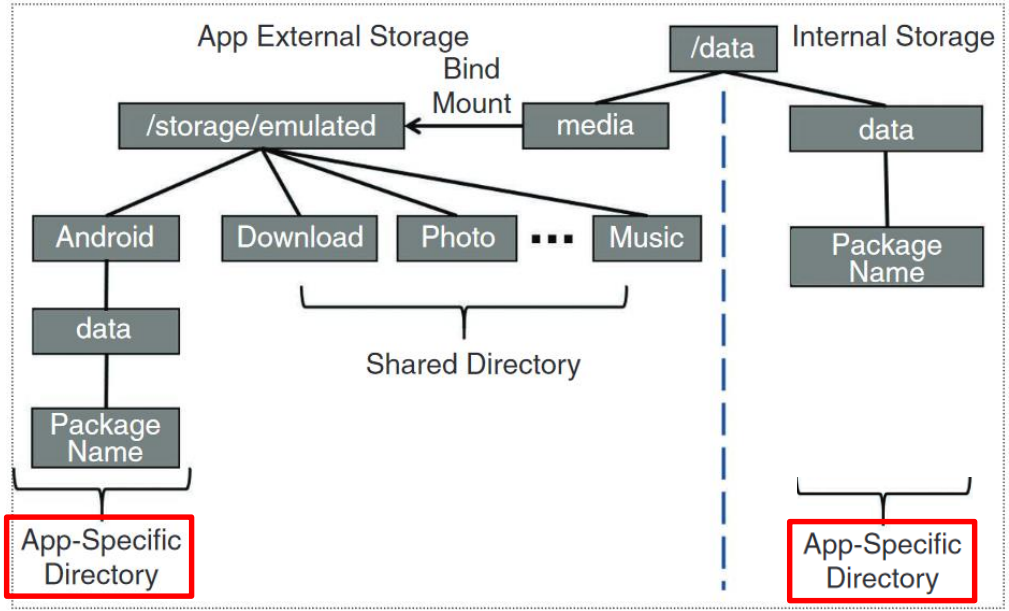


Image adapted from:
IEEE Security & Privacy
Demystifying Android's Scoped Storage Defense. Sept.-Oct. 2021, pp. 16-25, vol. 19
DOI Bookmark: 10.1109/MSEC.2021.3090564

App-specific files Storage

Both Internal and External storage include a dedicated location for

- storing persistent files
- storing cache data

Files stored in these directories are meant for use only by your app (App-specific)

- Otherwise use Shared storage (Photo, Video, Docs etc)

App-specific files Storage

When storing sensitive data (data that shouldn't be accessible from any other app), use

- internal storage
- Preferences
- database

Internal storage => data being hidden from users

When the user uninstalls your app, the files saved in app-specific storage are removed

External Storage for App-specific files

- If internal storage doesn't provide enough space to store app-specific files => use external storage

- The system provides directories within external storage where an app can organize files that provide value to the user only within your app

- The files in these directories aren't guaranteed to be accessible, such as when a removable SD card is taken out of the device. If your app's functionality depends on these files => internal storage

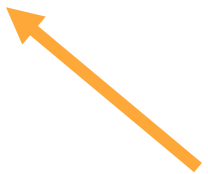
	Type of content	Access method	Permissions needed	Can other apps access?	Files removed on app uninstall?
App-specific files	Files meant for your app's use only	From internal storage, getFilesDir() or getCacheDir() From external storage, getExternalFilesDir() or getExternalCacheDir()	Never needed for internal storage Not needed for external storage when your app is used on devices that run Android 4.4 (API level 19) or higher	No	Yes
Media	Shareable media files (images, audio files, videos)	MediaStore API	READ_EXTERNAL_STORAGE when accessing other apps' files on Android 11 (API level 30) or higher READ_EXTERNAL_STORAGE or WRITE_EXTERNAL_STORAGE when accessing other apps' files on Android 10 (API level 29) Permissions are required for **all** files on Android 9 (API level 28) or lower	Yes, though the other app needs the READ_EXTERNAL_STORAGE permission	No
Documents and other files	Other types of shareable content, including downloaded files	Storage Access Framework	None	Yes, through the system file picker	No
App preferences	Key-value pairs	Jetpack Preferences library	None	No	Yes

Always check availability of storage

- Because the **external storage may be unavailable**
  - such as when the user has **mounted the storage to a PC** or has **removed the SD card** that provides the external storage
- you should **always verify that the volume is available** before accessing it

```
/* Checks if external storage is available for read and write */

public boolean isExternalStorageWritable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state)) {
        return true;
    }
    return false;
}
```



if the returned state is equal to MEDIA_MOUNTED, then you can read and write your files

Always check availability of storage (2)

```
/* Checks if external storage is available to at least read */
public boolean isExternalStorageReadable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state) ||
        Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
        return true;
    }
    return false;
}
```

Always check availability of storage (2)

```
/* Checks if external storage is available to at least read */
public boolean isExternalStorageReadable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state) ||
        Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
        return true;
    }
    return false;
}
```

Returns the current state of the primary external storage media…

otherwise use (e.g., for SD card etc):
`String getExternalStorageState (File path)`

Accessing External storage directories

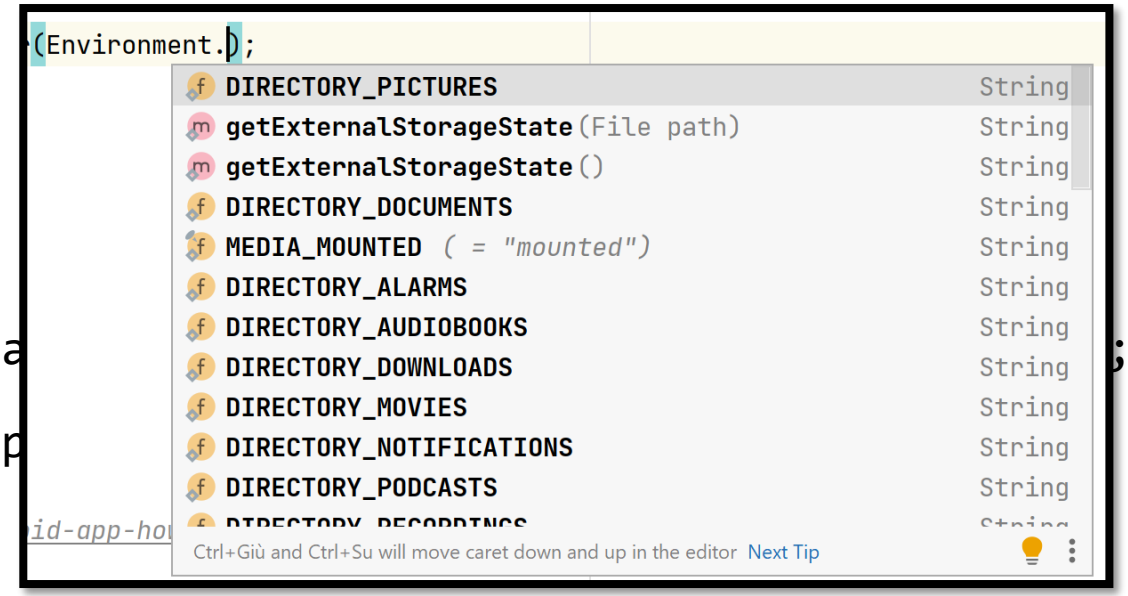
1. Get a path using **getExternalFilesDir()**
2. Create file

Example

```
File path = getExternalFilesDir(Environment.DIRECTORY_PICTURES);
File file = new File(path, "DemoPicture.jpg");
```

Accessing External storage directories

1. Get a path using **getExternalFilesDir()**
2. Create file



Example

```
File path = getExterna
File file = new File(p
```

See: https://developer.android.com/reference/android/os/Environment#fields

Select a physical storage location

A device that allocates a partition of its internal memory as external storage can also provide an SD card slot.

This means that the device has multiple physical volumes that could contain external storage, so you need to select which one to use for your app-specific storage

```
File[] externalStorageVolumes =
            ContextCompat.getExternalFilesDirs(getApplicationContext(), Environment.DIRECTORY_PICTURES);
```

// probably a partition of the device internal memory as external storage

```
File pathPrimaryExternalStorage = externalStorageVolumes[0];
```

// probably this is the SD card

```
File pathSecondaryExternalStorage = externalStorageVolumes[1];
```

the first element in the returned array (i.e., [0]) is considered the primary external storage volume

How much storage left?

- If there is not enough space, throws IOException
- If you know the size of the file, check against space
  - getFreeSpace()
  - getTotalSpace().
- If you do not know how much space is needed
  - try/catch IOException

How much storage left?

- If there is not enough space, throws IOException
- If you know the size of the file, check against space
  - getFreeSpace()
  - getTotalSpace().
- If you do not know how much space is needed
  - try/catch IOException

Delete files no longer needed

- External storage

```
myFile.delete();
```

- Internal storage

```
myContext.deleteFile(fileName);
```

Do not delete the user's files!

When the user uninstalls your app, your app's private storage directory and all its contents are deleted

Do not use private storage for content that belongs to the user!

For example

- Photos captured or edited with your app
- Music the user has purchased with your app

Simple Camera App

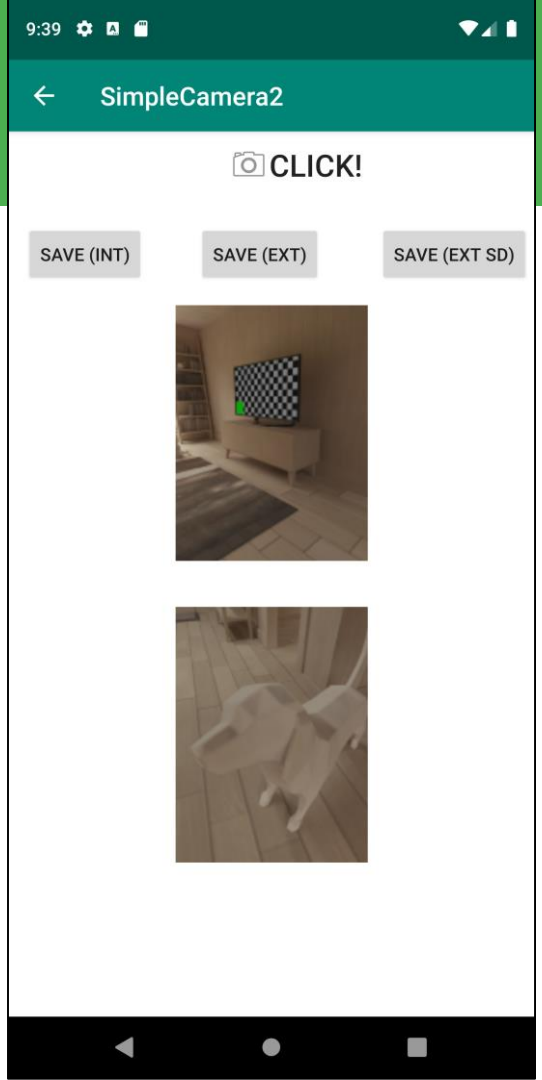
Add the possibility to save the **last picture** taken in the **External App-specific directories**:

in particular in both the:

- **External storage**
- **External SD removable storage**

Add the following ***UI elements*** to the ***Photo Activity***

- A **button** for **saving the Image** in the external storage
- A **button** for **saving the Image** in the external SD removable storage



Simple Camera App

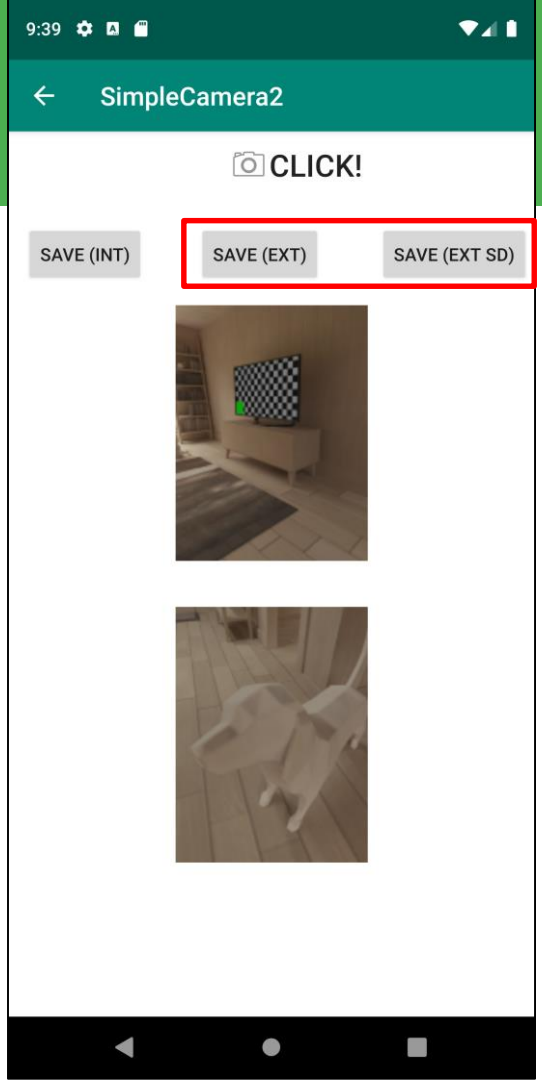
Add the possibility to save the **last picture** taken in the **External App-specific directories**:

in particular in both the:

- **External storage**
- **External SD removable storage**

Add the following ***UI elements*** to the ***Photo Activity***

- A **button** for **saving the Image** in the external storage
- A **button** for **saving the Image** in the external SD removable storage



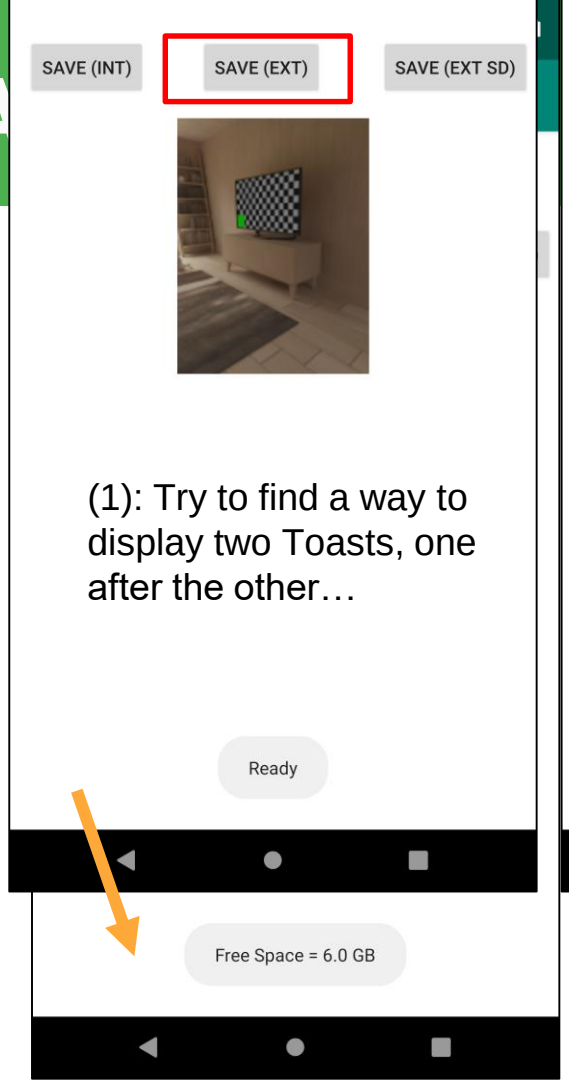
Simple Camera A

Then each time the user clicks on the two new save buttons:

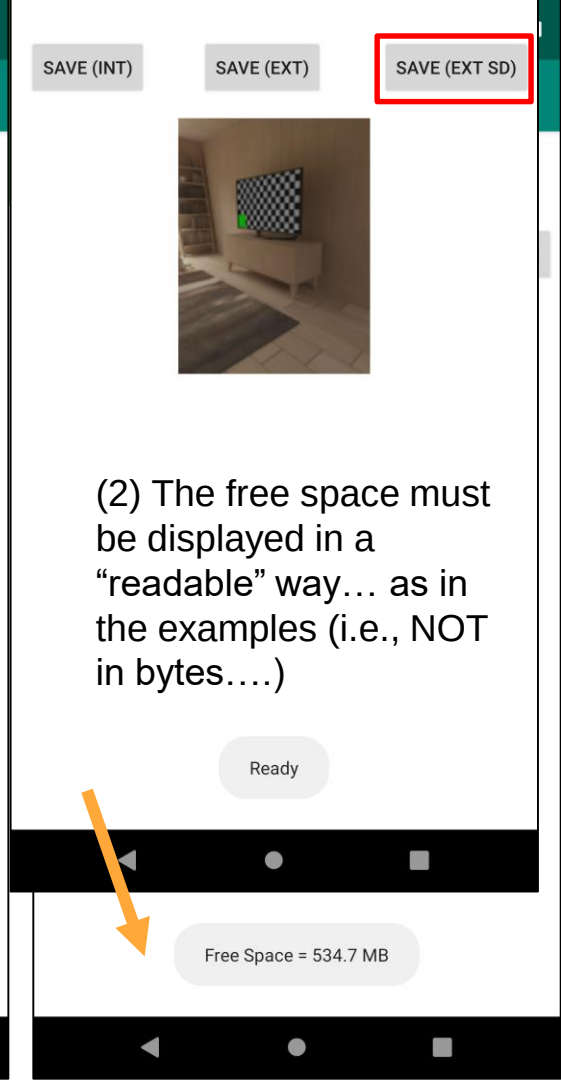
- a Toast appears to notify if the corresponding **external storage is available or not**
  External or SD card

- After 2-3 sec another Toast appears that shows the **free space available** on the selected ***external/SD storage***

Clearly, proceed with saving the image only if the storage is "Ready"



(1): Try to find a way to display two Toasts, one after the other…



(2) The free space must be displayed in a "readable" way… as in the examples (i.e., NOT in bytes….)

Simple Camera App

Finally, modify the main activity to show the images saved on the three possible locations for the App-specific files:

- ***Internal storage***
- ***External storage***
- ***External SD removable storage***

Navigate the device file system to find the two images saved on the

- ***External storage***
- ***External SD removable storage***