

## ĐỀ CƯƠNG BÀI GIẢNG

### BÀI 7. LƯU TRỮ DỮ LIỆU VỚI FILE

**Nội dung bài học trước khi lên lớp** (trang 1 đến trang 5):

- Lưu trữ dữ liệu với Shared Preferences.

**Nội dung bài học thực hiện lên lớp** (trang 6 đến hết):

- Lưu trữ dữ liệu trong thiết bị- Internal storage.
- Lưu trữ dữ liệu với thẻ nhớ - External storage.

**Nội dung bài học sau khi lên lớp:** Phiếu bài tập 7.1 đến 7.6

#### NỘI DUNG BÀI HỌC

<b>1. Giới thiệu.....</b>	<b>2</b>
<b>2. Lưu trữ dữ liệu với Shared Preferences . ....</b>	<b>2</b>
2.1. Lưu thông tin với SharedPreferences.....	2
2.2. Khôi phục thông tin trạng thái với SharedPreferences. ....	5
<b>3. Lưu trữ dữ liệu trong thiết bị- Internal storage.....</b>	<b>6</b>
3.1. Các bước ghi file .....	6
3.2. Đọc file từ Internal Storage.....	7
3.3. Lưu trữ dữ liệu sử dụng bộ nhớ cache .....	8
<b>4. Lưu trữ dữ liệu với thẻ nhớ - external storage.....</b>	<b>9</b>
4.1. Khai báo quyền đọc, ghi của ứng dụng trong file cấu hình AndroidManifest.xml .....	10
4.2. Đọc thẻ nhớ .....	10
4.3. Ghi dữ liệu vào thẻ nhớ.....	13
<b>5. Thao tác với file Json hoặc file có cấu trúc .....</b>	<b>13</b>

## 1. Giới thiệu

Để lưu trữ dữ liệu trong Android, nhà phát triển đã cung cấp nhiều lựa chọn lưu trữ dữ liệu phụ thuộc vào nhu cầu bảo mật và kích thước file cần lưu trữ. Một số phương pháp lưu trữ điển hình như:

- Lưu trữ trong bộ nhớ thiết bị - Internal file storage: Lưu trữ các file riêng tư của ứng dụng (app-private files) trong hệ thống file của thiết bị.
- Lưu trữ trong bộ nhớ ngoài - External file storage: Lưu trữ file trên hệ thống file chia sẻ ra được bên ngoài. Đây là phương pháp lưu trữ thường dùng cho các tình huống người dùng chia sẻ hình ảnh, video.
- Lưu trữ với Shared Preferences: Lưu trữ dữ liệu trên tập tin ứng dụng dưới dạng dữ liệu thô (các cặp key-value).
- Lưu trữ dữ liệu trong cơ sở dữ liệu – Database storage: Lưu trữ các dữ liệu có cấu trúc trong cơ sở dữ liệu.

Ngoại trừ một số loại file ở bộ nhớ ngoài (External Storage), thì tất cả các phương pháp lưu trữ dành cho dữ liệu riêng tư của ứng dụng - dữ liệu không thể truy cập tự nhiên vào các ứng dụng khác. Nếu muốn chia sẻ dữ liệu (file) với các ứng dụng khác sẽ có cách thực hiện riêng biệt sử dụng **FileProvider** API

Nếu ứng dụng muốn chia sẻ dữ liệu cho ứng dụng khác sử dụng, hệ điều hành Android cung cấp các lựa chọn qua thành phần Content Provider (nội dung đã viết chi tiết trong chương 5 giáo trình<sup>1</sup>). Content Provider có thể giúp người dùng thiết lập quyền kiểm soát đọc/ghi cho các ứng dụng khác bất kể cách thức lưu trữ dữ liệu đã có theo cách nào (thường là lưu trữ với cơ sở dữ liệu).

## 2. Lưu trữ dữ liệu với Shared Preferences .

### 2.1. Lưu thông tin với SharedPreferences

SharedPreferences là một dạng lưu trữ nhanh bằng file XML, để truy xuất dữ liệu yêu cầu thực hiện theo cặp khóa key – value đã được hệ điều hành hiểu và thực thi.

---

<sup>1</sup> Giáo trình Phát triển Ứng dụng trên thiết bị di động; TS Nguyễn Bá Nghiễn; ĐHCN HN

SharedPreferences có thể lưu trữ được các kiểu dữ liệu cơ bản như: Boolean; Float; Double; Int; Long; String. Dữ liệu vẫn được bảo toàn ngay cả khi ứng dụng bị đóng. Cách lưu trữ thông tin này thường **dùng để lưu trữ trạng thái thao tác của người dùng trên phần mềm.**

SharedPreferences pref=getSharedPreferences("PREP",MODE\_PRIVATE)  
nhận lấy thể hiện



**Vị trí lưu file:** Data/data/<package ID>/shared\_prefs/<file.xml>

Ví dụ: Vị trí lưu file trong máy ảo cài đặt ứng dụng trong giáo trình:

Device File Explorer		
Emulator Pixel_XL_API_29 Android 8.1.0, API 27		
Name	Date	
> com.google.android.youtube	2022-05-27 16:31	
> com.google.ar.core	2022-05-27 16:30	
> com.ustwo.lwp	2022-05-27 16:30	
> jp.co.omronsoft.openwnn	2022-05-27 16:30	
> org.chromium.webview_shell	2022-05-27 16:30	
> vuduong.com	2022-05-27 16:37	
> vuthiduong.m.bai1_hello_2022	2022-05-27 18:01	
> vuthiduong.m.bai4_12_todolist	2022-05-27 23:36	
> vuthiduong.m.bai4_spinner_listview_v2	2022-05-27 22:55	
✓ vuthiduong.m.bai8_1_sharedpreference	2022-05-31 09:17	
cache	2022-05-31 09:17	
code_cache	2022-05-31 09:17	
shared_prefs	2022-05-31 09:17	
login.xml	2022-05-31 09:29	

**a. Một số chế độ lưu trữ (MODE) được sử dụng**

<b>MODE_PRIVATE</b>	Lưu trữ riêng tư không cho phép các ứng dụng khác có thể truy cập vào tập tin SharedPreferences trong ứng dụng.
<b>MODE_WORLD_READABLE</b>	Cho phép các ứng dụng khác chỉ có thể đọc dữ liệu từ tập tin SharedPreferences trong ứng dụng nhưng không được phép ghi dữ liệu.
<b>MODE_WORLD_WRITEABLE</b>	Cho phép các ứng dụng khác có thể ghi dữ liệu vào trong tập tin SharedPreferences trong ứng dụng.

**b. Các phương thức lưu trữ dữ liệu của Editor**

Các phương thức của Editor	Mô tả
<b>putBoolean(Key, value)</b>	Phương thức lưu giá trị Boolean
<b>putFloat(Key, value)</b>	Phương thức lưu giá trị Float
<b>putInt(Key, value)</b>	Phương thức lưu giá trị <b>Integer</b>
<b>putLong(Key, value)</b>	Phương thức lưu giá trị Long
<b>putString(Key, value)</b>	Phương thức lưu giá trị <b>String</b>
<b>putSet(Key, value)</b>	Phương thức lưu giá trị mảng giá trị String
<b>Commit()</b>	Xác thực trạng thái lưu vào XML

Ví dụ: Lưu thông tin trạng thái tên ứng dụng và giá trị ứng dụng đang hoạt động:

```
SharedPreferences sharedPref= context.getSharedPreferences("MyPref",  
MODE_PRIVATE);  
SharedPreferences.Editor editor = sharedPref.edit();  
editor.putString("name", "DHCNHN");  
editor.putBoolean("active", true);  
editor.commit();
```

### c. Các phương thức đưa dữ liệu vào *SharedPreferences*

Các phương thức của Editor	Mô tả
<b>apply()</b>	Cập nhật dữ liệu mà không trả về kết quả thực thi lệnh thành công hay thất bại
<b>commit()</b>	Cập nhật dữ liệu và trả về kết quả là <b>true</b> nếu thực thi lệnh thành công và trả về <b>false</b> nếu thất bại
<b>clear()</b>	Xóa toàn bộ dữ liệu trong Shared Preference
<b>remove(String key)</b>	Xóa dữ liệu trong Shared Preference dựa vào “key” tương ứng.

Ghi chú:

- Lưu giá trị thông qua **apply()** vs **commit()**
  - o **commit()** lưu trữ dữ liệu đồng bộ (synchronously)
  - o **apply()** là không đồng bộ (asynchronously).
- Sử dụng phương thức **apply()** sẽ thực thi lệnh nhanh hơn so với **commit()**.

### 2.2. Khôi phục thông tin trạng thái với *SharedPreferences*.

Khi đọc dữ liệu thì ta có thể dùng trực tiếp từ *SharedPreferences* mà không cần thông qua đối tượng Editor. Tất cả các thao tác đọc thông tin được đi qua bộ nhớ trong, điều đó có nghĩa là ngay lập tức và tránh được các thao tác nhập/xuất dữ liệu. Lý do Android cho phép thực hiện như vậy bởi vì tất cả các thao tác đi qua bộ nhớ trong nên nó đảm bảo rằng giá trị trả về sẽ là giá trị mới nhất.

Thao tác đọc thông tin: **getXXX(key, default value)**.

<b>getBoolean(Key, value)</b>	Phương thức lấy giá trị Boolean
<b>getFloat(Key, value)</b>	Phương thức lấy giá trị Float
<b>getInt(Key, value)</b>	Phương thức lấy giá trị <b>Integer</b>
<b>getLong(Key, value)</b>	Phương thức lấy giá trị Long
<b>getString(Key, value)</b>	Phương thức lấy giá trị <b>String</b>
<b>getSet(Key, value)</b>	Phương thức lấy giá trị mảng giá trị String

Ví dụ: lấy thông tin đã được lưu trong mục.

```
SharedPreferences sharedPref = context.getSharedPreferences("MyPref",  
Context.MODE_PRIVATE);  
String name = sharedPref.getString("name", "");  
boolean active = sharedPref.getBoolean("active", false);
```

Việc lưu và đọc trạng thái rất quan trọng trong các phần mềm. Các thao tác thường lưu như tên đăng nhập, dòng dữ liệu người dùng vừa thao tác để khi đóng phần mềm, các thông tin quan trọng đã được lưu lại à khi phần mềm được khôi phục, dữ liệu trước khi đóng ứng dụng lại được hiển thị trở lại màn hình giao diện. Điều này giúp cho người dùng có trải nghiệm tốt hơn.

Thông thường, các thao tác lưu trữ hay được sử dụng trong **onPause()** và được khôi phục trong sự kiện **onResume()** của Activity. Các sự kiện này đc được trình bày rõ trong bài 3 của bài giảng.

### 3. Lưu trữ dữ liệu trong thiết bị- Internal storage

Android Internal Storage lưu trữ dữ liệu riêng tư của ứng dụng trên thiết bị. Mặc định các file được lưu trữ riêng tư và các ứng dụng khác không có quyền truy cập đến các loại dữ liệu. Tuy nhiên khi ứng dụng bị xóa, các file kèm ứng dụng sẽ bị xóa.

Các file lưu trữ trong bộ nhớ chỉ là các file đơn giản, không thể làm việc với file có đường dẫn

#### 3.1. Các bước ghi file

Bước 1: Gọi `openFileOutput()` với tên file và tham số chế độ hoạt động. Trả về một `FileOutputStream`

Bước 2: Ghi tới file sử dụng `write()`

Bước 2: Đóng stream sử dụng `close()`

👉 **Minh họa các bước qua câu lệnh:**

**Bước 1:**

```
String fileName = "myfile.txt";  
// Mở một luồng ghi file.  
FileOutputStream out = openFileOutput(fileName, MODE_PRIVATE);  
OutputStreamWriter writer= new OutputStreamWriter(out);
```

**Bước 2:** `writer.write(.....);`

**Bước 3:** `writer.close();`

👉 **Ghi chú:**

- `MODE_PRIVATE`: Đây là chế độ mặc định, file ghi ra chỉ được sử dụng bởi ứng dụng tạo ra nó, hoặc chia sẻ với cùng định danh người dùng.
- `MODE_APPEND`: Chế độ nối thêm dữ liệu vào file nếu nó đã tồn tại.
- `MODE_WORLD_READABLE/WRITEABLE`: Các chế độ này không an toàn, giống như một lỗ hổng bảo mật trong Android,. Hiện tại đã có các kỹ thuật khác thay thế : `ContentProvider`; `BroadcastReceiver`; `Service`

### 3.2. Đọc file từ Internal Storage

**Bước 1:** Gọi `openFileInput()`- truyền tên file muốn đọc. Trả về `FileInputStream`

**Bước 2:** Đọc sử dụng `read()`.

**Bước 3:** Sau đó đóng stream sử dụng `close()`.

👉 **Minh họa các bước qua câu lệnh:**

**Bước 1:** Gọi và tạo luồng.

```
FileInputStream in= openFileInput("myfile.txt");
BufferedReader reader=new BufferedReader( new
InputStreamReader(in));
```

**Bước 2:** Đọc luồng

```
String data="";
StringBuilder builder=new StringBuilder();
while((data=reader.readLine())!=null) {
    builder.append(data);
    builder.append("\n");
}
```

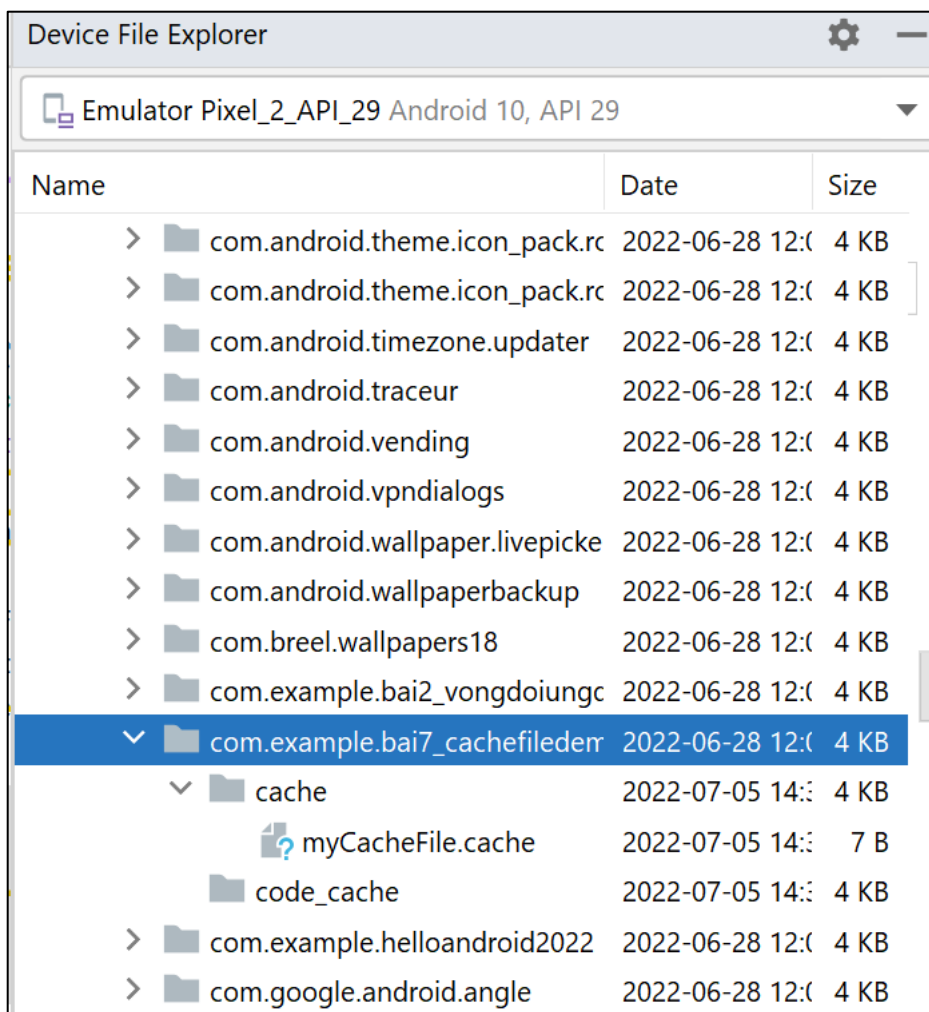
**Bước 3:** Đóng luồng

```
in.close();
```

### 3.3. Lưu trữ dữ liệu sử dụng bộ nhớ cache

Để tăng tốc độ xử lý của ứng dụng khi thường xuyên truy cập internet thì phương pháp lưu trữ dữ liệu trên cache là lựa chọn tối ưu.

- Khi lưu thông qua bộ nhớ cache thì thông tin sẽ được lưu trữ tại đường dẫn **data/data/<gói thư mục ứng dụng>/cache/<tên file>**. Minh hoạ hình sau: **Giả sử gói lưu thông tin ứng dụng com.example.bai7.cachefiledemo:**



Name	Date	Size
> com.android.theme.icon_pack.rc	2022-06-28 12:00	4 KB
> com.android.theme.icon_pack.rc	2022-06-28 12:00	4 KB
> com.android.timezone.updater	2022-06-28 12:00	4 KB
> com.android.traceur	2022-06-28 12:00	4 KB
> com.android.vending	2022-06-28 12:00	4 KB
> com.android.vpndialogs	2022-06-28 12:00	4 KB
> com.android.wallpaper.livepicke	2022-06-28 12:00	4 KB
> com.android.wallpaperbackup	2022-06-28 12:00	4 KB
> com.breel.wallpapers18	2022-06-28 12:00	4 KB
> com.example.bai2_vongdoiungc	2022-06-28 12:00	4 KB
▼ com.example.bai7_cachefiledemo	2022-06-28 12:00	4 KB
▼ cache	2022-07-05 14:00	4 KB
myCacheFile.cache	2022-07-05 14:00	7 B
code_cache	2022-07-05 14:00	4 KB
> com.example.helloandroid2022	2022-06-28 12:00	4 KB
> com.google.android.angle	2022-06-28 12:00	4 KB



**Mã lệnh đọc ghi file cache:** lấy đường dẫn của thư mục Cache trong ứng dụng- `getCacheDir()`. Các thao tác file thực hiện tương tự như quy định của java.

**Ghi file:**

```
public void createCache() {
    try {
        File pathCacheDir = getCacheDir();
        String strCacheFileName = "myCacheFile.cache";
        String strFileContents = editdata.getText()+"";
        File newCacheFile = new
            File(pathCacheDir, strCacheFileName);
        newCacheFile.createNewFile();
        FileOutputStream foCache = new FileOutputStream(
            newCacheFile.getAbsolutePath());
        foCache.write(strFileContents.getBytes());
        foCache.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

**Đọc file:**

```
public void readCache() {
    try {
        File pathCacheDir = getCacheDir();
        String strCacheFileName = "myCacheFile.cache";
        File newCacheFile = new
            File(pathCacheDir, strCacheFileName);
        Scanner sc=new Scanner(newCacheFile);
        String data="";
        while(sc.hasNext())
        {
            data+=sc.nextLine()+"\n";
        }
        editdata.setText(data);
        sc.close();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    }
}
```

#### 4. Lưu trữ dữ liệu với thẻ nhớ - external storage

Android External Storage: là nơi lưu trữ dữ liệu ngoài của Android, các file dữ liệu lưu trữ tại đây không được hệ thống áp dụng bảo mật. Ưu điểm nổi bật của cách lưu trữ này là có thể lưu trữ dung lượng lớn.

Thông thường có 2 loại lưu trữ ngoài (external storage):

- Lưu trữ ngoài cố định: Thường được hiểu là ổ cứng của điện thoại.
- Lưu trữ ngoài lưu động (Removable Storage): SD Card

#### 4.1. Khai báo quyền đọc, ghi của ứng dụng trong file cấu hình

##### AndroidManifest.xml

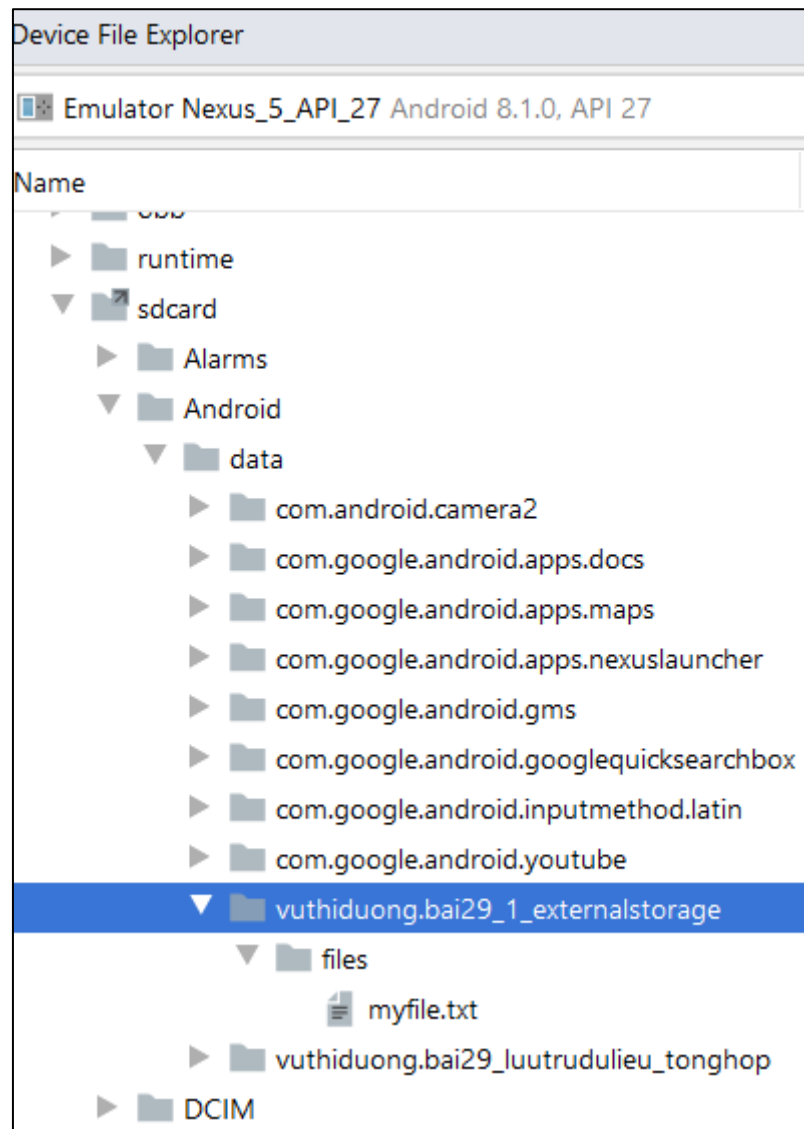
```
<uses-permission  
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>  
<uses-permission  
android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

#### 4.2. Đọc thẻ nhớ

Với Android API Level < 29. Các ứng dụng có thể lưu trữ dữ liệu của nó trên bộ lưu trữ ngoài (External Storage). Để lấy đường dẫn tới file lưu trữ theo mẫu câu lệnh: **Environment.getExternalStorageDirectory().getAbsolutePath()**.

Với cách lưu trữ như đề cập ở trên có thể gây ra xung đột giữa các ứng dụng khác nhau, vì tất cả các ứng dụng này có thể lưu trữ dữ liệu của nó vào cùng một thư mục (hoặc các thư mục con), hơn nữa nếu bạn gỡ bỏ (uninstall) một ứng dụng, dữ liệu do ứng dụng tạo ra vẫn tồn tại trên bộ lưu trữ ngoài.

Với Android API level 29+: mỗi ứng dụng sẽ lưu trữ dữ liệu của nó tạo ra tại một thư mục khác nhau trên bộ lưu trữ ngoài. Khi người dùng gỡ bỏ ứng dụng tất cả các dữ liệu do nó tạo ra cũng sẽ bị xóa bỏ. Để lấy đường dẫn tới file lưu trữ theo mẫu câu lệnh: **this.getExternalFilesDir(null).getAbsolutePath()**.



#### a. Đọc từ thẻ nhớ - dung Scanner

```
public void readData() {  
    //API level 29+  
    String sdcard= this.getExternalFilesDir(null).  
        getAbsolutePath()+"/myFile.txt";  
    try {  
        Scanner scan=new Scanner(new File(sdcard));  
        String data="";  
        while(scan.hasNext()) {  
            data+=scan.nextLine()+"\n";  
        }  
        scan.close();  
        editdata.setText(data+"");  
    } catch (FileNotFoundException e) {  
        e.printStackTrace();  
    }  
}
```

Để lấy được đường dẫn sử dụng các phương thức tĩnh của class Environment

- getDataDirectory(): trả về dữ liệu – file
- getDownloadCacheDirectory(): trả về thư mục lưu cache
- getExternalStorageState(): trả về state of main memory storage media
- getRootDirectory(): trả về thư mục gốc
- getExternalStorageDirectory()getAbsolutePath(): trả về đường dẫn bộ nhớ ngoài

***b. Đọc thẻ nhớ dùng file***

```
public void readData()    {
    //API level 29+
    String sdcard= this.getExternalFilesDir(null).
        getAbsolutePath()+"/myFile.txt";
    File file = new      File(sdcard);
    BufferedReader br =
        new BufferedReader(new      FileReader(file));
    String line;
    //Đọc dữ liệu từ file, nội dung sau khi đọc sẽ chứa
    trong biến content
    StringBuilder content = new StringBuilder();
    while ((line = br.readLine()) != null) {
        content.append(line);
        content.append('\n');
        br.close();
        editdata.setText(content + "");
    }
}
```

### 4.3. Ghi dữ liệu vào thẻ nhớ

```
public void writeData() {  
    //API level 29+  
    String sdcard= this.getExternalFilesDir(null).  
        getAbsolutePath()+"/myFile.txt";  
    try {  
        OutputStreamWriter writer=      new  
            OutputStreamWriter(  
                new FileOutputStream(sdcard));  
        writer.write("Mã 123");//dữ liệu cần ghi  
        writer.close();  
    } catch (FileNotFoundException e){  
        e.printStackTrace();  
    }  
} //end of writeData
```

### 5. Thao tác với file Json hoặc file có cấu trúc

Khuyến khích sinh viên tìm hiểu và trình bày tham luận