* Các dữ liệu có thể được lưu trữ theo những cách khác nhau tùy theo nhu cầu bảo mật, kích thước, thời gian, …
* Một số tiêu chí khi lưu trữ dữ liệu:
  + Khả năng truy cập dữ liệu
    - Từ
      * Chính ứng dụng
      * Ứng dụng khác
    - Khi ứng dụng
      * Đã hoặc chưa gỡ cài đặt
      * Kết nối mạng hay chưa
    - Dung lượng lưu trữ
* Content Provider: Giới hạn quyền truy cập dữ liệu áp đặt bởi lập trình viên.
* Các giải pháp lưu trữ dữ liệu
  + Shared Preferences:
    - Lưu trạng thái đăng nhập người dùng
    - Lưu cài đặt như ngôn ngữ
    - Chứa dữ liệu private nguyên thủy dưới dạng key-value: boolean, float, …
    - Có thể dụng để tạo thiết lập người dùng
    - Được lưu trữ trong tệp XML: /data/data/<application’s package name>/shared\_prefs
    - Sẽ bị xóa khi gỡ ứng dụng
    - Có thể liên kết với ứng dụng hoặc một Activity cụ thể
    - Có thể được liên kết với ứng dụng hoặc một Activity cụ thể
    - Có 2 cách lấy:
      * getSharedPreferences(String name, int mode): Nôm na là khi cần lấy sharePreferece từ activity khác hoặc có nhiều sharePreference
      * getPreferences(): Sử dụng khi activity đó chỉ có 1 sharePreference
      * Một số phương thức:
        + contains(String key) - Kiểm tra xem preferences có chứa một preference cụ thể không
        + getAll() - Lấy toàn bộ values từ preferences
        + getFloat(String key, float defValue) - Lấy một giá trị kiểu float từ preferences.
        + getInt(String key, int defValue) - Lấy một giá trị kiểu int từ preferences.
        + getStringSet(String key, Set<String> defValues) - Lấy một tập giá trị kiểu String từ preferences.
      * Các bước ghi giá trị
        + Gọi hàm edit() để nhận SharedPreferences.Editor
        + Thêm giá trị bằng các hàm putBoolean() và putString()
        + Ghi nhận giá trị mới với hàm commit()
  + Android File System:
    - Lưu trữ dữ liệu tạm thời
    - Lữu trữ dữ liệu JSON hoặc CSV
    - Sử dụng Linux kernel tuân thủ VFS (Virtual FileSystem) – một lớp trừu tượng phía trên của các tệp tin hệ thống cụ thể
      * Giúp dễ dàng thêm kiểu tệp tin hệ thống vào Android
      * Đa số lữu trữ định theo EXT4
    - Có cấu trúc tương tự với các bản phân phối Linux chuẩn
    - Cấu trúc thư mục thông thường:
      * etc - Một liên kết tượng trưng tới /system/etc
      * proc - mount point cho tệp tin hệ thống procfs, cho phép truy cập tới cấu trúc dữ liệu kernel. Ứng dụng như ps, lsof, và vmstat sử dụng /proc như nguồn dữ liệu
      * root - Thư mục gốc cho tài khoản root
      * sdcard - Một liên kết tượng trưng tới một thư mục trong /storage/\*
      * sys - mount point cho tệp tin hệ thống sysfs pseudo, là sự ánh xạ của cấu trúc đối tượng thiết bị của kernel.
      * system - mount point cho /dev/block/mtdblock0. Các thư mục trong đây thường được nhìn thấy trong thư mục root của các bản phân phối tiêu chuẩn Linux bao gồm các thư mục bin, etc, lib, usr, và xbin.
      * vendor - một liên kết tượng trưng tới /system/vendor chứa các đoạn mã thuộc về nhà sản xuất.
  + Bộ nhớ trong:
    - Lưu trữ CSDL SQLite(thông tin người dùng, danh sách sản phẩm, lịch sử truy cập, vv)
    - Lưu trữ tệp ảnh tạo ra bởi người dùng trong ứng dụng
    - Thường private cho ứng dụng
    - Bị xóa khi gỡ cài đặt UD
    - Để đọc một tệp tin từ bộ nhớ trong:
      * Gọi openFileInput() và truyền vào tên của tệp tin cần đọc. Hàm sẽ trả lại một FileInputStream.
      * Đọc bytes từ tệp tin với hàm read()
      * Đóng luồng với hàm close()
    - Đọc và ghi một tệp private trong bộ nhớ trong:
      * Gọi hàm openFileOutput() với tên của tệp tin và chế độ họat động. Hàm trả về một FileOutputStream.
      * Ghi tệp sử dụng hàm write()
      * Đóng tệp sử dụng hàm close()
    - Chế độ openFileOutput
      * Mặc định là MODE\_PRIVATE
      * MODE\_APPEND: nếu tệp tin đã tồn tại, tiếp tục ghi dữ liệu vào cuối tệp tin thay vì xóa đi ghi lại
    - Tệp tin cache:
      * Sử dụng pt getCacheDir() mở file đại diện trong thư mục lưu trữ tạm thời
      * Các tệp cache sẽ tự động bị xóa khi hệ thống cần dung lượng trên đĩa
        + HT xóa tệp cũ bằng lastModified()
        + Một cách có kiểm soát hơn là sử dụng setCacheBehaviorGroup(File, boolean) và setCacheBehaviorTombstone(File, boolean)
      * getCacheQuotaBytes(java.util.UUID) quy định dung lượng cache bởi:
        + Tần suất người dùng tương tác với ứng dụng
        + Dung lượng đĩa system-wide disk sử dụng
      * Một số phương thức khác:
        + getFilesDir() - Lấy đường dẫn tuyệt đối tới thư mục filesystem, nơi mà các tệp internal được lưu trữ.
        + getDir() - Tạo (hoặc mở nếu đã tồn tại) thư mục bên trong bộ nhớ trong
        + deleteFile() - Xóa tệp tin trong bộ nhớ trong
        + fileList() - Trả về mảng tệp tin hiện tại được lưu trữ bởi ứng dụng.
  + Bộ nhớ ngoài:
    - Lưu trữ hình ảnh và video tải từ máy chủ hoặc từ người dùng khác
    - Lưu trữ tệp tài liệu nặng
    - Có thể là các thiết bị lưu trữ có thể tháo bỏ như SD card hoặc bộ nhớ không thể tháo bỏ(bộ nhớ trong)
    - Dữ liệu trong bộ nhớ ngoài là world-readable, có thể bị sửa đối khi bật USB mass storage để chuyển tệp trên máy tính
    - Cần yêu cầu READ\_EXTERNAL\_STORAGE hoặc WRITE\_EXTERNAL\_STORAGE trong file manifest để đọc và ghi
    - Nên kiểm tra trạng thái của thiết bị trước khi sử dụng do bộ nhớ ngoài có thể tháo bỏ bằng phương thức getExternalStorageState()
    - Tệp tin công khai:
      * Luôn sẵn sàng sử dụng bởi các ứng dụng khác và người dùng kể cả khi gỡ cài đặt ứng dụng
      * Gọi getExternalStoragePublicDirectory(String type) để nhận thư mục lưu trữ các tệp tin công khai
        + Tham số type thuộc vào các giá trị sau: DIRECTORY\_MUSIC, DIRECTORY\_PODCASTS, DIRECTORY\_RINGTONES, DIRECTORY\_ALARMS, DIRECTORY\_NOTIFICATIONS, DIRECTORY\_PICTURES, DIRECTORY\_MOVIES, DIRECTORY\_DOWNLOADS, DIRECTORY\_DCIM, hoặc DIRECTORY\_DOCUMENTS
        + Đường dẫn trả về bởi phương thức này có thể không tồn tại, nên sử dụng path.mkdirs() để đảm bảo điều này
    - Tệp tin private:
      * Không cung cấp giá trị cho người dùng ngoài ứng dụng
      * Các tệp private sẽ bị xóa khi gỡ ứng dụng
      * Gọi phương thức getExternalFilesDir(String type) để nhận thư mục lưu trữ các tệp tin private.
  + CSDL SQLite:
    - Lưu trữ thông tin sản phẩm
    - Lưu trữ lịch sử truy cập hoặc lịch sử tìm kiếm để đưa ra gợi ý
    - Phù hợp với dữ liệu lặp hoặc có cấu trúc
    - Giá trị khóa nên tăng dần giúp tìm kiếm nhanh hơn
    - Lưu trữ trên một vùng đìa private liên kết với ứng dụng
    - Lớp Contract
      * Chỉ định khung lược đồ dữ liệu, tự tài liệu hóa
      * Cho phép sử dụng chung hằng trong toàn bộ lớp cùng package
  + Network:
    - Lưu trữ dữ liệu từ máy chủ từ xa như Firebase
    - Lưu trữ dữ liệu realtime từ một máy chủ từ xa (WebSocket)
    - Truy cập dữ liệu qua mạng
    - Thao tác qua các gói java.net.\*, android.net.\*