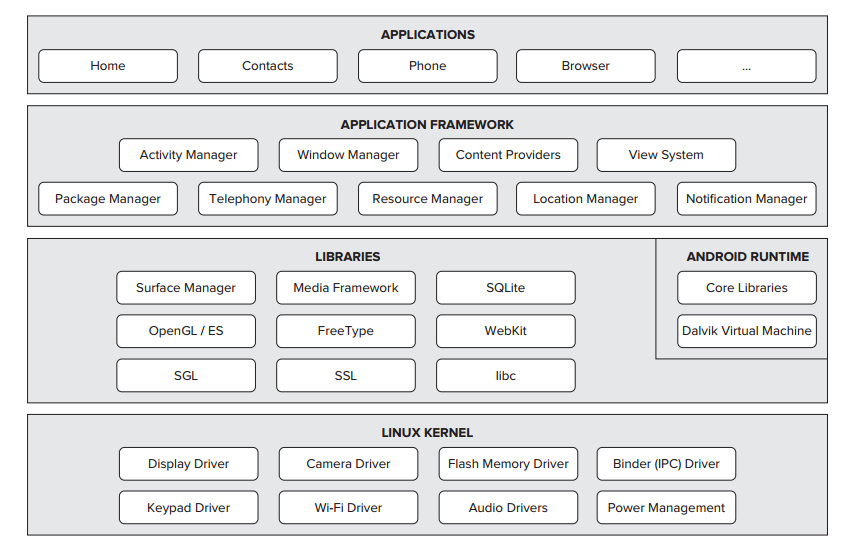
**Part 1**

1. **Tuần 1**

**Kiến trúc Android**

Hệ điều hành Android là 1 ngăn xếp các thành phần phần mềm, được chia thành 5 phần và 4 lớp chính như trong hình bên dưới.



**Linux Kernel**

Lớp nhân Linux là lớp phần mềm cao nhất trong nền tảng Android.

Lớp này cung cấp các dịch vụ cốt lõi mà bất kỳ thiết bị máy tính Android nào cũng sẽ dựa vào.

Nhân Linux của Android, giống như bất kỳ nhân Linux nào, cung cấp các dịch vụ hệ điều hành chung.

Nó xử lý các chi tiết mạng và tệp I / O cấp thấp. Và nó cũng cho phép bạn tải xuống các plugin Android giao tiếp với một loạt các thành phần phần cứng cấp thấp thường được kết hợp với các thiết bị điện toán, những thứ như bộ nhớ và radio và máy ảnh.

## Các thư viện

Các thư viện này thường được viết bằng C C ++ và vì lý do đó, chúng thường được gọi là các thư viện nguyên gốc. Và các thư viện riêng này xử lý rất nhiều cốt lõi, hoạt động nhạy cảm với hiệu suất trên thiết bị của bạn, những thứ như nhanh chóng làm các trang web và cập nhật màn hình.

## Các thư viện Android

Đây là các thư viện dựa trên Java phục vụ cho việc phát triển Android. Ví dụ của các thư viện này bao gồm các thư viện ứng dụng dùng để xây dựng giao diện người dùng, vẽ đồ họa hay truy cập cơ sở dữ liệu

* android.app - Cung cấp quyền truy cập vào ứng dụng và là nền tảng của tất cả ứng dụng Android.
* android.content - Cung cấp quyền truy cập nội dung (content), truyền tải thông điệp giữa các ứng dụng hay các thành phần của ứng dụng.
* android.database - Đựoc sử dụng để truy cập dữ liệu của content provider và cơ sở dữ liệu SQLite
* android.opengl - giao diện các phuơng thức Java để sử dụng OpenGL ES
* android.os - Cung cấp các ứng dụng với quyền truy cập vào các dịch vụ của hệ điều hành bao gồm thông điệp, các dịch vụ hệ thống và các giao tiếp nội tại
* android.text - Đựoc sử dụng để hiển thị và điều chỉnh chữ trên màn hình thiết bị
* android.view - Các thành phần cơ bản trong việc xây dựng giao diện người dùng của ứng dụng.
* android.widget - Tập các thành phần giao diện người dùng đã được xây dựng sẵn như các nút, các nhãn (label), list view,....
* android.webkit - Tập các lớp cho phép xây dựng khả năng duyệt web.

## Android Runtime

Đây là phần thứ 3 của kiến trúc và nằm ở lớp thứ 2 từ dưới lên. Phần này cung cấp 1 bộ phận quan trọng là Dalvik Vỉtual Machine - là 1 loại Java Virtual Machine được thiết kế đặc biệt để tối ưu cho Android.

Dalvik VM sử dụng các đặc trưng của nhân Linux như quản lý bộ nhớ và đa luồng, những thứ mà đã có sẵn trong Java. Dalvik VM giúp mọ ứng dụng Android chạy trong tiến trình riêng của nó, với các thể hiện (instance) riêng của Dalvik virtual Machine.

Android Runtime cũng cung cấp 1 tập các thư viện chính giúp các nhà phát triển ứng dụng Android có thể viết ứng dụng Android bằng Java

## Application Framework

Lớp Android Framework cung cấp các dịch vụ cấp độ cao hơn cho các ứng dụng dưới dạng các lớp Java. Các nhà phát triển ứng dụng được phép sử dụng các dịch vụ này trong ứng dụng của họ.

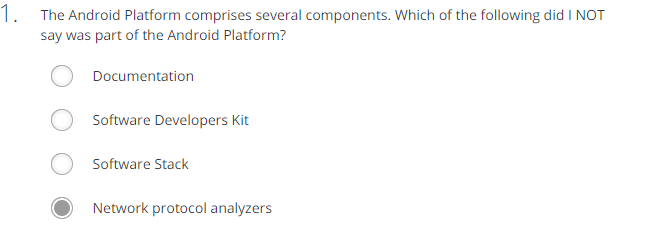
Một thành phần khung ứng dụng là Trình quản lý gói. Trình quản lý gói về cơ bản là một cơ sở dữ liệu theo dõi tất cả các ứng dụng hiện được cài đặt trên thiết bị của bạn.

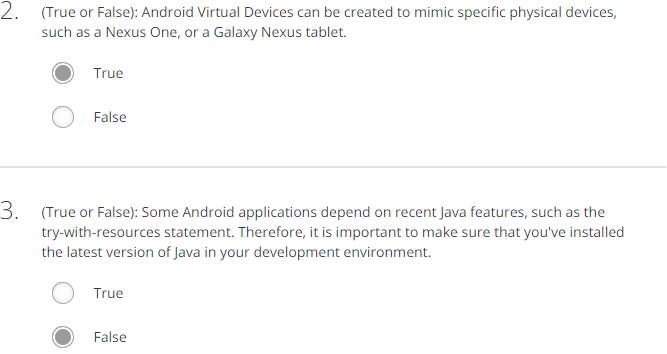
Android Framework bao gồm các dịch vụ chính sau:

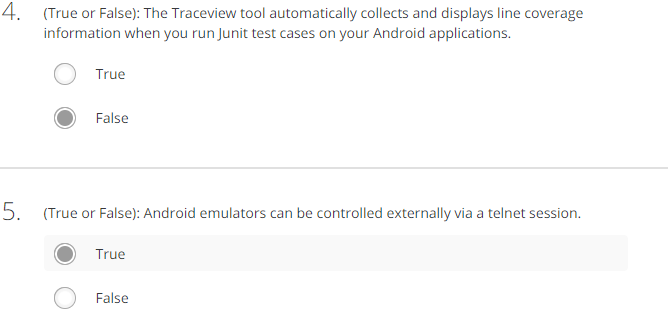
* Activitty Manager - Kiểm soát tất cả khía cạnh của vòng đời ứng dụng và ngăn xếp các Activity.
* Content Providers - Cho phép các ứng dụng chia sẽ dữ liệu với các ứng dụng khác.
* Resource Manager - Cung cấp quyền truy cập vào các tài nguyên như các chuỗi, màu sắc, các layout giao diện người dùng...
* Notifications Manager - Cho phép các ứng dụng hiển thị cảnh báo và các thông báo cho người dùng.
* View System - Tập các thành phần giao diện (view) được sử dụng để tạo giao diện người dùng.

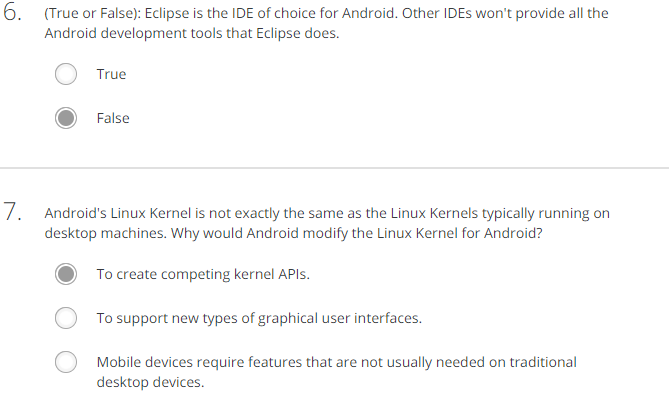
## Application

Lớp trên cùng của kiến trúc là Application. Các ứng dụng bạn tạo ra sẽ được cài đặt trên lớp này. Ví dụ như: Danh bạ, nhắn tin, trò chơi...

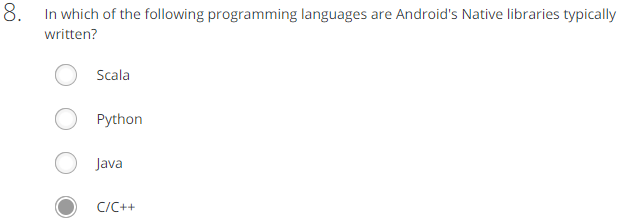
**Quick**

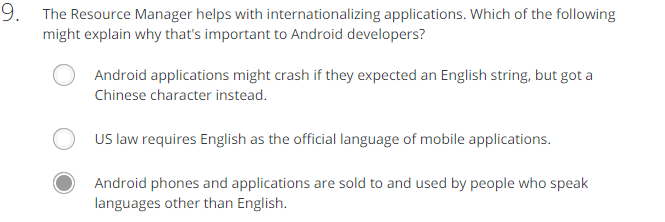


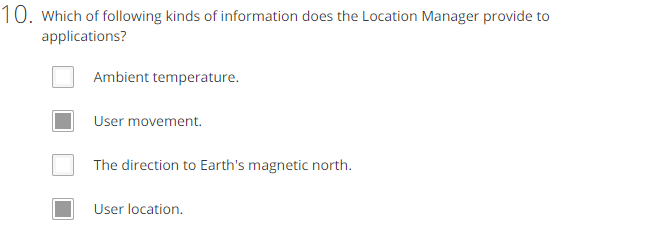


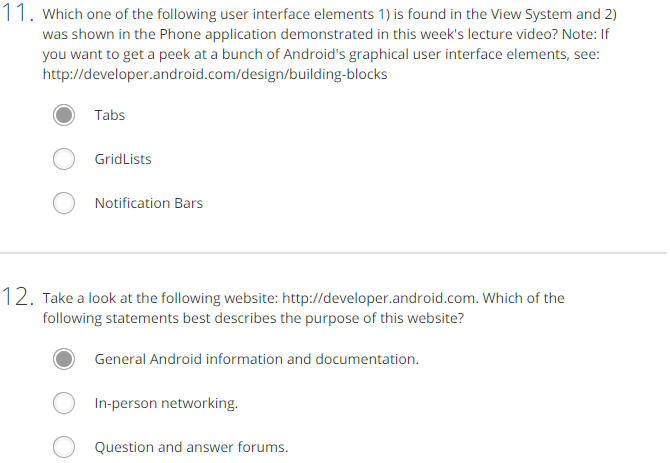


Câu 7 c nha









1. **Tuần 2**

#### Application Fundamentals

**Part 1:**

Có bốn khối xây dựng cơ bản mà tất cả các ứng dụng Android được xây dựng dựa trên chúng. Các khối xây dựng này được triển khai như các lớp Java. Và khối đầu tiên trong số các khối xây dựng này là lớp hoạt động (Activity class). Đó là lớp chính mà người dùng nhìn thấy khi họ chạy một ứng dụng. Activities được thiết kế để cung cấp giao diện đồ họa người dùng (GUI) cho người dùng. Và điều này cho phép người dùng cung cấp thông tin đến một ứng dụng và nhận thông tin từ một ứng dụng. Ba thành phần còn lại hoạt động ở phía sau, vì vậy chúng không có giao diện người dùng. Và các thành phần này bao gồm các dịch vụ để hỗ trợ hoạt động lâu dài hoặc trong các hoạt động nền.

Broadcast receiver là một trong 4 khối xây dựng của Android, với mục đích lắng nghe và phản hồi các sự kiện xảy ra trên thiết bị. Và các content provider cho phép nhiều ứng dụng lưu trữ và chia sẻ dữ liệu. Ứng dụng thường tạo ra từ nhiều thành phần cộng tác. Mà Android khởi động và chạy nếu cần. Và mỗi thành phần này phục vụ một mục đích khác nhau trong hệ sinh thái Android, do đó có điểm vào riêng và API riêng.

Đầu tiên,lớp hoạt động (Activity class) được thiết kế để trình bày giao diện đồ họa người dùng cho người dùng. Và để nắm bắt sự tương tác của người dùng thông qua giao diện đó. Theo nguyên tắc chung, một activity nên tập trung hỗ trợ một điều duy nhất mà người dùng có thể làm. Một điều duy nhất như quay số điện thoại, hoặc nhập thông tin liên lạc cho một người,... Ví dụ: ứng dụng quay số điện thoại có thể mở giao diện người dùng với nhiều tab. Một cho trình quay số điện thoại, một cho nhật ký cuộc gọi và một cho danh sách liên lạc. Trên Android 4.2, mã nguồn ứng dụng này thực sự là một phần của ứng dụng ngữ cảnh. Và nó được viết trong một tệp có tên Dial tacksActivity.java. Lớp Dial tacksActivity là một lớp hoạt động. Và các comment ​​ở đây, giải thích rằng hoạt động này được cho là hiển thị một tab chứa điện thoại, trình quay số điện thoại, nhật ký cuộc gọi.

Thành phần tiếp theo là lớp dịch vụ (Service class). Không giống như các activity, dịch vụ chạy trong nền (Background). Vì vậy, dịch vụ không cần có giao diện người dùng. Thay vào đó, các dịch vụ có hai mục đích chính, một là chúng có thể thực hiện các thao tác chạy kéo dài thường cách xa luồng giao diện chính. Và hai, chúng cung cấp một cách để các quy trình khác yêu cầu thao tác và chia sẻ dữ liệu. Ví dụ: Bạn mở ứng dụng phát nhạc và chọn một bài nhạc ưa thích, khi bạn thực hiện một hành động khác như kiểm tra email, nhạc vẫn phát. Android xử lý việc này bằng cách sử dụng một dịch vụ để phát nhạc, đi đến ứng dụng âm nhạc và mở tệp MediaPlaybackservice.java. Như bạn có thể thấy, dịch vụ MediaPlayback là một lớp con của dịch vụ. Và các ý kiến ​​giải thích rằng dịch vụ này cho phép âm nhạc tiếp tục phát, ngay cả khi người dùng chuyển đổi giữa các hoạt động khác nhau.

Thành phần tiếp theo là Broadcast Receiver. Broadcast Receivers lắng nghe và phản hồi các sự kiện.Trong Android, các sự kiện được đại diện bởi intent class. Các Publisher tạo các intent này và sau đó phát chúng bằng cách sử dụng một phương thức như các context class gửi broadcast. Bây giờ một khi broadcast các intent này sau đó được chuyển đến các broadcast receiver đã đăng ký hoặc đăng ký cho các intent cụ thể đó. Tại thời điểm đó nó có thể đáp ứng với sự kiện này. Ví dụ: Tưởng tượng rằng một nơi nào đó trên thế giới, ai đó quyết định rằng họ muốn gửi cho tôi một tin nhắn SMS. Tin nhắn SMS đó sẽ được tạo và gửi và sẽ truyền qua mạng điện thoại. Và cuối cùng đến điện thoại của tôi. Và khi có, Android sẽ đặt biểu tượng thông báo vào thanh thông báo để cho tôi biết rằng tin nhắn SMS đã đến với tôi. Bây giờ, tất nhiên, bạn không thể biết chính xác khi nào bạn sẽ nhận được một tin nhắn như vậy. Vì vậy, Android có một số phần mềm chỉ ngồi và chờ tin nhắn SMS đến. Và khi chúng thực hiện, phần mềm đó phát ra một intent, SMS\_ received. Và có một broadcast receiver khác đang lắng nghe intent đó. Và broadcast receiver đó cuối cùng sẽ nhận được intent và sau đó sẽ khởi động một dịch vụ sẽ tải xuống và lưu trữ tin nhắn SMS đến. Bây giờ ở đây tôi sẽ truy cập ứng dụng SMS và mở một trong những broadcast receivers của nó, SMSReceiver.Java. Như bạn có thể thấy, lớp SMSReceiver là một lớp con của broadcast receiver. Và các comment ​​giải thích rằng broadcast receiver này chuyển tin nhắn SMS đến cho một dịch vụ quan tâm đến việc tải xuống và lưu trữ tin nhắn.

Thành phần cuối cùng là nhà cung cấp nội dung (Content provider). Nhà cung cấp nội dung cho phép các ứng dụng lưu trữ và chia sẻ dữ liệu. Các nhà cung cấp nội dung sử dụng giao diện kiểu cơ sở dữ liệu. Nhưng chúng không chỉ là cơ sở dữ liệu. Ví dụ: các nhà cung cấp nội dung sẽ xử lý các chi tiết cấp thấp của giao tiếp giữa các quá trình. Vì vậy, các ứng dụng chạy trong các quy trình riêng biệt có thể giao tiếp và trao đổi dữ liệu một cách an toàn và dễ dàng. Ứng dụng trình duyệt là một ví dụ về ứng dụng sử dụng các nhà cung cấp nội dung. Ví dụ: Nếu tôi nhấp vào biểu tượng bên cạnh thanh địa chỉ, trình duyệt sẽ mở danh sách dấu trang hoặc địa chỉ đã lưu cho các trang web mà tôi có thể muốn truy cập nhanh trong tương lai. Khi người dùng thêm một trong những dấu trang này, ứng dụng trình duyệt sẽ lưu nó trong nhà cung cấp nội dung. Hãy mở mã ứng dụng trình duyệt và xem tệp BrowserProvider.java. Như bạn có thể thấy, nhà cung cấp trình duyệt là một lớp phụ của nhà cung cấp nội dung. Bạn có thể thấy một số gợi ý rằng có một bảng cơ sở dữ liệu được gọi là dấu trang, được sử dụng để lưu trữ các dấu trang mà người dùng thêm vào.

Quá trình viết và xây dựng các ứng dụng Android. Trước tiên, bạn tạo source code and the non-source code resources tạo nên ứng dụng của bạn. Tiếp theo, các công cụ biên dịch mã nguồn của bạn và chuẩn bị tài nguyên của bạn.

Đầu ra của bước này là gói Android hoặc APK, tương ứng với khả năng thực thi của ứng dụng của bạn.

Tiếp theo, APK được đăng ký để xác định bạn là nhà phát triển của nó.

Và cuối cùng APK được cài đặt trên thiết bị hoặc trình giả lập và chạy.

Là một nhà phát triển, việc bạn tham gia vào quá trình xây dựng thường sẽ bao gồm bốn bước sau đây. Xác định tài nguyên, thực hiện các lớp ứng dụng, đóng gói ứng dụng và cài đặt và chạy ứng dụng. Đặc biệt để thử nghiệm và gỡ lỗi.

Bước một là xác định tài nguyên của ứng dụng của bạn. Các ứng dụng Android không chỉ là mã nguồn. Chúng bao gồm các thực thể mã không nguồn là tốt. Những thứ như layout file, chuỗi, hình ảnh, menu, hình động, và nhiều hơn nữa. Và quản lý tài nguyên riêng biệt với ứng dụng có một số lợi ích, một trong số đó là bạn có thể dễ dàng thay đổi các tài nguyên đó. Không phải thay đổi hoặc biên dịch lại mã nguồn ứng dụng của bạn theo bất kỳ cách nào.

Một loại tài nguyên phổ biến là chuỗi. Trong Android, có ba loại tài nguyên chuỗi. Các chuỗi riêng lẻ, mảng của chuỗi và số nhiều. Số nhiều về cơ bản là các mảng chuỗi có thể được sử dụng để chọn các chuỗi cụ thể được liên kết với số lượng nhất định. Chẳng hạn như một cuốn sách hoặc hai cuốn sách.

Và các chuỗi thường được lưu trữ trong một tệp XML trong thư mục res / value của ứng dụng của bạn. Và định dạng là, thẻ chuỗi XML chứa thuộc tính name và sau đó chứa chính chuỗi đó. Và chuỗi thực tế đó có thể bao gồm định dạng và thông tin kiểu dáng, ví dụ như các thẻ HTML.

Cuối cùng, các tệp tài nguyên khác có thể tham chiếu đến các chuỗi mà bạn đã xác định là ký hiệu và string/string\_name. Trong mã Java, bạn cũng có thể truy cập các chuỗi này nhưng lần này bạn thực hiện nó dưới dạng R.string.string\_name.

Tôi đã xác định một số chuỗi cho ứng dụng vị trí bản đồ. Chúng ta hãy mở các tệp String.xml trong thư mục res / value.

Tập tin này cho thấy hai chuỗi. Show\_map\_string. Với giá trị hiển thị bản đồ và location\_string với giá trị Enter Location.

Bây giờ tôi cũng đã định nghĩa một tệp String.xml thứ hai, lần này là ngôn ngữ tiếng Ý.

Và nó nằm trong thư mục res / value-it. Và tiền tố IT trong tên thư mục là thứ cho Android biết rằng đây là tệp chuỗi cho ngôn ngữ tiếng Ý.

Bây giờ nếu chúng ta xem tệp đó, chúng ta sẽ thấy hai chuỗi giống nhau show \_map \_string và location\_string. Nhưng lần này, giá trị của chúng là từ tiếng Ý, thay vì từ tiếng Anh.

Trong ví dụ trên, nếu ngôn ngữ mặc định của bạn là tiếng Ý, thì chuỗi vị trí sẽ có giá trị, [FOREIGN]. Nếu không, nó sẽ có giá trị, enter location. Bây giờ, tôi sẽ chạy Bản đồ vị trí hai lần. Điện thoại bên trái sử dụng tiếng Anh làm ngôn ngữ mặc định trong khi điện thoại bên phải sử dụng tiếng Ý làm ​​ngôn ngữ mặc định.

**Part 2:**

Một loại tài nguyên khác là một layout file. Các Layout file xác định giao diện người dùng cho một phần ứng dụng của bạn sẽ trông như thế nào. Và một lần nữa, các tệp này được viết bằng XML. Mặc dù một số công cụ sẽ cho phép bạn tạo bố cục một cách trực quan. Và sau đó những công cụ đó sẽ tạo ra XML cho bạn. Trong thực tế, Eclipse sẽ làm điều đó.

Các Layout files thường được lưu trữ trong thư mục res / layout của ứng dụng của bạn. Và bạn có thể truy cập Layout trong Java dưới dạng R.layout.layout\_name. Và bạn có thể truy cập layout đó trong các tệp tài nguyên khác dưới dạng &layout / layout name.

Bây giờ giống như các tệp chuỗi mà việc sử dụng phụ thuộc vào ngôn ngữ mặc định của bạn, Android cho phép bạn tạo nhiều Layout file và sau đó Android có thể chọn từ các tệp đó trong thời gian chạy dựa trên cấu hình của thiết bị.

Tệp này xác định rằng bố cục bao gồm một thứ gọi là RelativeLayout, một bố cục mà khi chúng ta nói về giao diện người dùng. Và bên trong bố cục tương đối, có một yếu tố khác gọi là văn bản chỉnh sửa. Đó là nơi mà bạn sử dụng để nhập postal address. Ngoài ra còn có một nút, đó là nút được dán nhãn hiển thị bản đồ.

Eclipse cũng có thể hiển thị những gì nó nghĩ rằng tất cả sẽ trông giống như lúc chạy. Và bạn có thể thấy điều đó bằng cách nhấp vào tab graphical layout.

Bây giờ khi bạn ở đó, bạn cũng có thể nhấp vào các phần tử riêng lẻ và Eclipse sẽ hiển thị cho bạn thông tin chi tiết hơn về các thuộc tính bố cục của chúng.

Map location cũng có một tệp khác, còn được gọi là tệp main.xml nhưng tệp này nằm trong thư mục res / layout-land.

Nếu Map location đang chạy với một thiết bị ở chế độ ngang, tệp này được sử dụng thay cho vị trí mà chúng ta đã thấy trước đó.

Bây giờ bạn có thể thấy, tệp này sử dụng cùng các yếu tố, bố cục tương đối, văn bản chỉnh sửa và nút, nhưng tôi đã thay đổi vị trí của chúng một chút để tất cả các yếu tố xuất hiện trên một dòng, mà tôi nghĩ trông sẽ tốt hơn chế độ phong cảnh.

Điện thoại bên trái đang chạy ở chế độ dọc, do đó, nó sử dụng tệp main.xml mặc định. Điện thoại bên phải, tuy nhiên, đang chạy ở chế độ ngang. Vì vậy, nó sẽ sử dụng tệp main.xml trong res / layout-land.

Và như bạn có thể thấy các yếu tố thực sự được đặt ra khác nhau tùy thuộc vào hướng của thiết bị.

Bây giờ tôi đã đề cập một vài lần rằng các tài nguyên có thể được truy cập trong Java dưới dạng R.something hoặc khác. Android tạo ra một lớp gọi là R, từ ứng dụng của bạn. Và sau đó bạn có thể truy cập vào các trường của lớp R này để đến các chuỗi và bố cục đó và các tài nguyên khác mà bạn đã xác định trong các tệp XML. Chúng ta hãy xem một, tại một tệp R.Java thực tế.

Tập tin này được tạo bởi Android, do đó bạn không nên sửa đổi nó.

Bạn có thể thấy rằng tệp định nghĩa lớp R và lớp R này chứa một lớp khác gọi là Bố cục. Trong đó có một lĩnh vực được gọi là chính. Và điều đó thực sự mang đến cho bạn một tham chiếu hoặc xử lý tệp main.xml. Ngoài ra còn có một lớp id cung cấp các thẻ điều khiển cho bố cục tương đối, vào hộp văn bản chỉnh sửa và cho nút được xác định trong các tệp main.xml. Thực tế, nếu bạn quay lại và kiểm tra các tệp main.xml ngay bây giờ, bạn sẽ thấy rằng có các dòng ghi Android: id. Và đó là nơi mà các trường và ID đến từ.

Và cuối cùng, có một lớp gọi là chuỗi, cung cấp các thẻ điều khiển cho tất cả các chuỗi mà chúng ta đã nói đến.

Bước tiếp theo trong phát triển ứng dụng là triển khai các lớp Java của bạn. Và điều này thường liên quan đến việc viết ít nhất một hoạt động. Điểm vào cho các hoạt động là phương thức onCreate. Vì vậy, đó là nơi bạn sẽ thường khởi tạo ứng dụng của mình. Chúng ta hãy xem phương pháp onCreate chi tiết hơn một chút.

Trong onCreate, bạn thường làm bốn điều sau đây. Bạn khôi phục trạng thái ứng dụng đã lưu. Bạn đặt chế độ xem nội dung, cho Android biết những gì sẽ hiển thị dưới dạng giao diện người dùng hoạt động. Bạn khởi tạo các thành phần cụ thể của giao diện người dùng hoạt động và cuối cùng, bạn đính kèm mã vào các thành phần giao diện người dùng đó để các hành động cụ thể sẽ được thực hiện khi người dùng tương tác với các thành phần giao diện người dùng này. Hãy xem cách các bước này được triển khai trong MapLocation. Vì vậy, chúng ta hãy xem mã nguồn MapLocation. Tất cả đều chứa trong một tệp có tên MapLocation.java trong thư mục nguồn của ứng dụng. Bây giờ, như bạn có thể thấy tệp này thực hiện một lớp có tên là MapLocation, là một lớp con của hoạt động.

Là một hoạt động, nó có một phương thức onCreate, trong đó ứng dụng được khởi tạo. Bước đầu tiên của onCreate là gọi super.onCreate chuyển trạng thái cá thể đã lưu làm tham số. Bây giờ trạng thái thể hiện đã lưu này là một kiểu gì đó thuộc kiểu gói và về cơ bản, đó là cấu trúc dữ liệu chứa bất kỳ thông tin nào mà Android có thể đã lưu từ lần cuối hoạt động được chạy

Bây giờ khi bạn ở đó, bạn cũng có thể nhấp vào các phần tử riêng lẻ và Eclipse sẽ hiển thị cho bạn thông tin chi tiết hơn về các thuộc tính bố cục của chúng.

Map location cũng có một tệp khác, còn được gọi là tệp main.xml nhưng tệp này nằm trong thư mục res / layout-land.

Nếu Map location đang chạy với một thiết bị ở chế độ ngang, tệp này được sử dụng thay cho vị trí mà chúng ta đã thấy trước đó.

Bây giờ bạn có thể thấy, tệp này sử dụng cùng các yếu tố, bố cục tương đối, văn bản chỉnh sửa và nút, nhưng tôi đã thay đổi vị trí của chúng một chút để tất cả các yếu tố xuất hiện trên một dòng, mà tôi nghĩ trông sẽ tốt hơn chế độ phong cảnh.

Điện thoại bên trái đang chạy ở chế độ dọc, do đó, nó sử dụng tệp main.xml mặc định. Điện thoại bên phải, tuy nhiên, đang chạy ở chế độ ngang. Vì vậy, nó sẽ sử dụng tệp main.xml trong res / layout-land.

Và như bạn có thể thấy các yếu tố thực sự được đặt ra khác nhau tùy thuộc vào hướng của thiết bị.

Bây giờ tôi đã đề cập một vài lần rằng các tài nguyên có thể được truy cập trong Java dưới dạng R.s Something hoặc khác. Chà, để làm điều này, Android tạo ra một lớp gọi là R, từ ứng dụng của bạn. Và sau đó bạn có thể truy cập vào các trường của lớp R này để đến các chuỗi và bố cục đó và các tài nguyên khác mà bạn đã xác định trong các tệp XML. Chúng ta hãy xem một, tại một tệp R.Java thực tế.

Như tôi đã nói, tập tin này được tạo bởi Android, do đó bạn không nên sửa đổi nó.

Bạn có thể thấy rằng tệp định nghĩa lớp R và lớp R này chứa một lớp khác gọi là Bố cục. Trong đó có một lĩnh vực được gọi là chính. Và điều đó thực sự mang đến cho bạn một tham chiếu hoặc xử lý tệp main.xml. Ngoài ra còn có một lớp id cung cấp các thẻ điều khiển cho bố cục tương đối, vào hộp văn bản chỉnh sửa và cho nút được xác định trong các tệp main.xml. Thực tế, nếu bạn quay lại và kiểm tra các tệp main.xml ngay bây giờ, bạn sẽ thấy rằng có các dòng ghi Android: id. Và đó là nơi mà các trường và ID đến từ.

Và cuối cùng, có một lớp gọi là chuỗi, cung cấp các thẻ điều khiển cho tất cả các chuỗi mà chúng ta đã nói đến.

Bước tiếp theo trong phát triển ứng dụng là triển khai các lớp Java của bạn. Và điều này thường liên quan đến việc viết ít nhất một hoạt động. Điểm vào cho các hoạt động là phương thức onCreate. Vì vậy, đó là nơi bạn sẽ thường khởi tạo ứng dụng của mình. Chúng ta hãy xem phương pháp onCreate chi tiết hơn một chút.

Trong onCreate, bạn thường làm bốn điều sau đây. Bạn khôi phục trạng thái ứng dụng đã lưu. Bạn đặt chế độ xem nội dung, cho Android biết những gì sẽ hiển thị dưới dạng giao diện người dùng hoạt động. Bạn khởi tạo các thành phần cụ thể của giao diện người dùng hoạt động và cuối cùng, bạn đính kèm mã vào các thành phần giao diện người dùng đó để các hành động cụ thể sẽ được thực hiện khi người dùng tương tác với các thành phần giao diện người dùng này. Hãy xem cách các bước này được triển khai trong MapLocation. Vì vậy, chúng ta hãy xem mã nguồn MapLocation. Tất cả đều chứa trong một tệp có tên MapLocation.java trong thư mục nguồn của ứng dụng. Bây giờ, như bạn có thể thấy tệp này thực hiện một lớp có tên là MapLocation, là một lớp con của hoạt động.

Là một hoạt động, nó có một phương thức onCreate, trong đó ứng dụng được khởi tạo. Bước đầu tiên của onCreate là gọi super.onCreate chuyển trạng thái cá thể đã lưu làm tham số. Bây giờ trạng thái thể hiện đã lưu này là một kiểu gì đó thuộc kiểu gói và về cơ bản, đó là cấu trúc dữ liệu chứa bất kỳ thông tin nào mà Android có thể đã lưu từ lần cuối hoạt động được chạy

Tiếp theo có một cuộc gọi đến setContentView. Trong trường hợp này, chuyển qua tham chiếu của tệp bố cục cho ứng dụng này, R.layout.main. Sau đó, có một số mã thu được các tham chiếu đến các thành phần UI riêng lẻ trong bố cục. Chẳng hạn như hộp văn bản chỉnh sửa, được lưu trữ trong một biến gọi là addrText. Và nút, được lưu trữ trong một biến gọi là nút.

Như bạn có thể thấy, các tham chiếu này có được bằng cách gọi Activity.findViewById. Và truyền vào id của phần tử mong muốn.

Và cuối cùng, có một số mã xác định phải làm gì khi người dùng nhấn nút hiển thị bản đồ.

Mã này thực hiện giao diện người nghe nhấp chuột có phương thức onClick được gọi bất cứ khi nào người dùng nhấp vào nút này. Và mã trong phương thức onClick đó, trước tiên nhận bất kỳ văn bản nào mà người dùng đã nhập vào hộp địa chỉ. Sau đó, nó xử lý văn bản đó để xóa khoảng trắng và đặt nó ở định dạng mà Google Maps hiểu. Và sau đó, nó khởi động ứng dụng Google Map, ứng dụng Google Maps đi qua văn bản địa chỉ được sửa đổi này. Và một lần nữa, tôi sẽ giải thích những gì mã này đang làm chi tiết hơn trong các bài học sau này khi chúng ta đi sâu vào hoạt động của lớp hoạt động và lớp ý định.

Ngoài ra, để đơn giản hóa mọi thứ, tôi đã bỏ qua mọi kiểm tra lỗi hoặc xác minh chuỗi địa chỉ được nhập nhưng tất nhiên trong mã sản xuất bạn thực sự sẽ muốn làm điều đó.

Bước tiếp theo trong việc tạo các ứng dụng Android là cung cấp thông tin cho phép các công cụ xây dựng của Android tạo gói ứng dụng hoặc apk. Và thông tin này được viết trong một tệp XML có tên là AndroidManifest.XML.

Tệp AndroidManifest.XML chứa nhiều thông tin đóng gói, bao gồm tên của ứng dụng và danh sách các thành phần tạo nên ứng dụng đó. Nó cũng bao gồm các thông tin khác mà chúng ta sẽ thảo luận sau trong khóa học. Những thứ như quyền cần thiết để chạy ứng dụng này. Các tính năng phần cứng như máy ảnh mà ứng dụng này yêu cầu và phiên bản nền tảng sớm nhất mà ứng dụng này chạy. Chúng ta hãy xem tệp AndroidManifest.XML để biết vị trí bản đồ. Tệp AndroidManifest.XML nằm trong thư mục cấp cao nhất của ứng dụng. [SOUND] Hãy mở nó ngay bây giờ. [SOUND] Bên trong, bạn thấy rằng có một thẻ kê khai, nói với Android rằng tệp này chứa thông tin đóng gói mà nó cần. Có rất nhiều thông tin khác ở đây nữa, nhưng tôi sẽ chỉ thảo luận về một vài thông tin.

Một yếu tố chúng ta thấy ở đây là yếu tố sử dụng SDK. Phần tử này bao gồm một thuộc tính, phiên bản SDK tối thiểu chỉ định cấp API tối thiểu cho ứng dụng này.

Và trong trường hợp này, mức đó là mười tương ứng với một trong các bản phát hành Android 2.3. Phần tử này cũng bao gồm một thuộc tính khác, targetSDKVersion, chỉ định cấp API mới nhất mà ứng dụng này đã được thử nghiệm.

Trong trường hợp này, đó là cấp 17 và tương ứng với bản phát hành Android 4.2. Ngoài ra còn có một yếu tố ứng dụng chỉ định những thứ như biểu tượng cho ứng dụng này và nhãn được hiển thị trên thanh tiêu đề của ứng dụng. Bên trong phần tử ứng dụng, có một phần tử hoạt động liệt kê một hoạt động, trong trường hợp này là vị trí bản đồ, bao gồm ứng dụng này.

Và bước cuối cùng là cài đặt ứng dụng trên thiết bị hoặc trình giả lập để kiểm tra và gỡ lỗi. Trong một bài học trước đó, chúng ta đã thấy cách làm điều đó trong Eclipse. Và bạn cũng có thể làm điều này từ dòng lệnh, ví dụ bằng cách phát lệnh cho công cụ adb.

#### The Activity Class

**Part 1**

Trong Android, một tác vụ chỉ đơn giản là một tập hợp các hoạt động liên quan. Những hoạt động liên quan này có thể, nhưng không phải là một phần của cùng một ứng dụng. Vì vậy, các tác vụ có thể trải rộng trên nhiều ứng dụng và hầu hết các tác vụ bắt đầu ở màn hình chính. khi người dùng khởi chạy một ứng dụng từ màn hình chính, một tác vụ mới thường được bắt đầu. Và khi người dùng nhấn nút home, để quay lại màn hình chính. Nhiệm vụ hiện tại là nó, ít nhất là tạm thời đóng cửa.

Nhiệm vụ backstack hoạt động như sau. Khi một hoạt động được khởi chạy, nó sẽ được chuyển sang nhiệm vụ dự phòng, thông thường là một phần của nhiệm vụ hiện tại. Khi hoạt động đó bị hủy sau đó, chẳng hạn vì người dùng nhấn nút quay lại, vì hoạt động đó tự chấm dứt theo chương trình, hoặc thậm chí vì chính Android đã quyết định hủy hoạt động đó để lấy lại tài nguyên của mình và hoạt động bị tắt khỏi tác vụ.

Khi một hoạt động bị dừng vì một hoạt động mới bắt đầu, nó được thông báo về sự thay đổi trạng thái này qua các phương pháp gọi lại vòng đời của hoạt động. Có một vài phương pháp gọi lại vòng đời mà một hoạt động có thể nhận, do một thay đổi về trạng thái của nó—dù hệ thống đang tạo, dừng hay tiếp tục nó, hay hủy nó—và mỗi lần gọi lại cho bạn cơ hội thực hiện công việc cụ thể phù hợp với sự thay đổi trạng thái đó. Ví dụ, khi bị dừng, hoạt động của bạn sẽ giải phóng mọi đối tượng lớn, chẳng hạn như các kết nối mạng hoặc cơ sở dữ liệu. Khi hoạt động tiếp tục, bạn có thể thu lại những tài nguyên cần thiết và tiếp tục những hành động bị gián đoạn. Những chuyển tiếp trạng thái này đều là một phần của vòng đời hoạt động.

Vòng đời khác thay đổi phụ thuộc vào chính Android. Ví dụ: nếu thiết bị của bạn sắp hết bộ nhớ, Android có thể quyết định hủy các hoạt động hiện đang bị treo. Và nó sẽ làm điều đó khi biết rằng nó sẽ cần phải tạo lại chúng sau này nếu người dùng điều hướng trở lại chúng.

Vòng đời hoạt động và những thay đổi trong vòng đời cụ thể mà các hoạt động trải qua.

Ví dụ: một khi một hoạt động được bắt đầu, nó có thể ở trạng thái tiếp tục hoặc đang chạy. Và trong khi ở trạng thái này, hoạt động có thể nhìn thấy và người dùng có thể tương tác với nó.

Chẳng hạn, một hoạt động cũng có thể bị tạm dừng khi một hoạt động mới bắt đầu bật lên trước hoạt động đó. Trong tình huống này, hoạt động có thể vẫn hiển thị một phần nhưng người dùng không thể tương tác với nó vì người dùng sẽ tương tác với một hoạt động mới bắt đầu.

Trước phiên bản 3.0, Android có thể chấm dứt các hoạt động khi chúng chuyển sang trạng thái tạm dừng. Hoạt động có thể được dừng lại. Và khi nó dừng lại, hoạt động đó không còn nhìn thấy được nữa. Và Android là tự do để chấm dứt. Nó có thể chấm dứt các hoạt động dừng lại. Và nó làm như vậy biết rằng nó có thể cần phải tạo lại chúng sau này, nếu người dùng điều hướng trở lại chúng.

Bây giờ các hoạt động của bạn thường sẽ cần phải cư xử khác nhau trong các phần khác nhau của vòng đời. Ví dụ: nếu hoạt động của bạn đang hiển thị animation, sau đó bật lên một hoạt động kiểu hộp thoại trong suốt phía trước nó, bạn có thể muốn tạm dừng animation trong khi người dùng phản hồi hộp thoại và sau đó khởi động lại animation sau khi hoạt động hộp thoại kết thúc .

Để hỗ trợ các tình huống như thế này, Android thông báo thay đổi vòng đời cho hoạt động bằng cách gọi vòng đời cụ thể hoặc phương thức mẫu.

Khi ứng dụng được khởi chạy, Android sẽ gọi phương thức onCreate của hoạt động.

Sau đó, Android sẽ gọi phương thức onStart của nó. Và sau đó onResume. Sau đó, giao diện người dùng của hoạt động sẽ xuất hiện trên màn hình của thiết bị. Và người dùng có thể tương tác với nó. Sau một phút như vậy, hoạt động ví dụ của chúng tôi sẽ bắt đầu ngừng hoạt động.

Và tại thời điểm này, Android sẽ gọi phương thức onPause của hoạt động đó.

Sau đó, onStop và cuối cùng, onDestroy. Và tại thời điểm này, hoạt động hoàn toàn chết.

Vì vậy, như bạn có thể thấy toàn bộ thời gian hoạt động chạy từ khi bắt đầu onCreate đến hết onDestroy.

Bây giờ khi hoạt động đơn giản này bắt đầu, nó không hiển thị trên màn hình. Tại một số điểm, nó trở nên hữu hình, và tại một số điểm sau đó, nó trở nên vô hình khi nó bị xóa khỏi màn hình.

Khi các hoạt động sắp được hiển thị, Android sẽ gọi khởi động và đôi khi, phương thức khởi động lại.

Khi các hoạt động sắp trở nên vô hình, Android gọi phương thức onStop.

Và do đó, bạn có thể nghĩ về thời gian hiển thị của một hoạt động là xảy ra giữa các lệnh gọi khởi động đến onStart và kết thúc các cuộc gọi đến onStop. Và cuối cùng, trong khi một hoạt động hiển thị trên màn hình, có những lúc người dùng có thể tương tác với nó và có những lúc họ không thể. Và ví dụ, điều này có thể xảy ra khi một thiết bị rơi vào trạng thái ngủ.

Trong trường hợp đó, người dùng không thể tương tác với hoạt động nữa, mặc dù đó vẫn là hoạt động nền trước. Vì vậy, khi các hoạt động sắp sẵn sàng cho sự tương tác của người dùng, Android sẽ gọi phương thức onResume.

Khi các hoạt động sắp ngừng có thể tương tác với người dùng, Android sẽ gọi phương thức onPause. Và do đó, bạn có thể nghĩ về thời gian hiển thị và trong vòng đời của một hoạt động xảy ra giữa lúc bắt đầu các cuộc gọi đến onResume và kết thúc các cuộc gọi đến onPause.

**Part 2**

Vì vậy, hãy xem xét sâu hơn về các phương pháp vòng đời này. Phương thức đầu tiên được gọi là OnCreate. OnCreate được gọi khi hoạt động được tạo lần đầu tiên. Thông thường OnCreate được sử dụng để khởi tạo hoạt động và sẽ thực hiện bốn chức năng sau. Nó sẽ gọi super.onCreate để Android có thể thực hiện một số khởi tạo của riêng mình. Nó sẽ đặt chế độ xem nội dung của hoạt động, cho Android biết giao diện người dùng của hoạt động nên là gì.

Tiếp theo, nó sẽ nắm bắt và giữ lại bất kỳ tham chiếu nào nó cần cho các chế độ xem hoặc thành phần khác nhau của giao diện người dùng. Và cuối cùng, nó sẽ cấu hình các giao diện và các thành phần giao diện người dùng theo yêu cầu.

OnRestart là một phương pháp vòng đời khác. Nó được gọi nếu hoạt động đã bị dừng, nhưng sắp bắt đầu lại. Và bạn có thể sử dụng onRestart để xử lý bất kỳ xử lý đặc biệt nào mà bạn cần, hoặc điều đó chỉ cần thiết sau khi hoạt động bị dừng, nhưng sắp bắt đầu lại.

onStart được gọi khi hoạt động sắp được hiển thị. Và một số ứng dụng điển hình cho phương pháp này bao gồm bắt đầu khi các hành vi chỉ hiển thị, như yêu cầu cập nhật cảm biến vị trí hoặc tải hoặc đặt lại trạng thái ứng dụng liên tục, những thứ như cập nhật hàng đợi tin nhắn chưa đọc cho ứng dụng đọc email.

Phương thức onResume được gọi khi hoạt động hiển thị và sắp bắt đầu tương tác với người dùng. Và một số điều bạn làm trong phương pháp này là bắt đầu các hành vi chỉ có tiền cảnh, những việc như bắt đầu hoạt hình hoặc phát một bản nhạc nền. onPause được gọi khi hoạt động của bạn sắp mất tập trung. Trong phương pháp này, bạn có thể tắt các hành vi chỉ có tiền cảnh, những thứ như giết chết hình ảnh động. Và bạn cũng nên lưu bất kỳ trạng thái liên tục mà người dùng đã chỉnh sửa.

onStop được gọi khi hoạt động không còn hiển thị cho người dùng. Tại thời điểm này, có thể hoạt động có thể được khởi động lại sau. Vì vậy, một số điều điển hình cần làm ở đây bao gồm trạng thái hoạt động lưu trữ mà bạn sẽ muốn khôi phục khi hoạt động này được khởi động lại sau đó và onStart được gọi. Và hãy nhớ rằng, onStop không phải lúc nào cũng được gọi khi một hoạt động kết thúc. Chẳng hạn, nó có thể không được gọi nếu Android hủy quá trình của ứng dụng do bộ nhớ thấp. Vì vậy, đừng chờ đợi để lưu dữ liệu liên tục trong phương pháp này. Làm điều đó trở lại trong onPause.

onDestroy được gọi với hoạt động sắp bị hủy và một số hành động điển hình cần thực hiện ở đây bao gồm giải phóng tài nguyên hoạt động. Ví dụ: tắt các luồng riêng được bắt đầu bởi hoạt động này. Và một lần nữa, hãy nhớ rằng, onDestroy có thể không được gọi ở tất cả. Ví dụ: khi Android giết một ứng dụng do bộ nhớ thấp trên thiết bị.

khi Activity được kích hoạt, và được hệ thống để vào BackStack. Sau khi kích hoạt, lần lượt các callback onCreate(), onStart(), onResume() sẽ được hệ thống gọi đến.  
Sau khi gọi đến các callback trên, thì Activity mới chính thức được xem là đang chạy (Activity running).  
Lúc này, nếu có bất kỳ Activity nào khác chiếm quyền hiển thị, thì Activity hiện tại sẽ rơi vào trạng thái onPause(). Nếu cái sự hiển thị của Activity khác làm cho Activity mà chúng ta đang nói đến không còn nhìn thấy nữa thì onStop() sẽ được gọi tiếp theo nữa.  
Nếu Acvitity đã vào onPause() rồi, tức là đang bị Activity khác đè lên, mà người dùng sau đó quay về lại Activity cũ, thì onResume() được gọi. Còn nếu Activity đã vào onStop() rồi, mà người dùng quay về lại Activity cũ thì onRestart() được gọi. Trong cả hai trường hợp Activity rơi vào onPause() hoặc onStop(), nó sẽ rất dễ bị hệ thống thu hồi (tức là bị hủy) để giải phóng tài nguyên, khi này nếu quay lại Activity cũ, onCreate() sẽ được gọi chứ không phải onResume() hay onRestart(). Và cuối cùng, nếu một Activity bị hủy một cách có chủ đích, chẳng hạn như người dùng nhấn nút Back ở System Bar, hay hàm finish() được gọi,… thì onDestroy() sẽ được kích hoạt và Activity kết thúc vòng đời của nó.

**onCreate()**  
Hàm này được gọi khá sớm, ngay khi activity được kích hoạt và thầm chí người còn chưa thấy gì cả thì callback này đã đc gọi rồi. Ngoài ra thì bạn nên biết là callback này chỉ được gọi một lần duy nhất khi Activity được khởi tạo. Nó có thể được gọi lại nếu hệ thống xóa Activity này đi để lấy lại tài nguyên của hệ thống, nhưng rất hiếm khi xảy ra. Và nó còn có thể được gọi lại nếu bạn xoay màn hình ngang/dọc.  
Do đặc tính được gọi khá sớm và chỉ được gọi một lần duy nhất trong vòng đời của nó như vậy, nên bạn sẽ tận dụng để load giao diện cho Activity ở giai đoạn này, thông qua phương thức setContentView(). Ngoài giao diện ra, bạn có thể khởi tạo các logic nào đó chỉ chạy một lần ban đầu, như các lời gọi API, load database, tạo item list, tạo Navigation Drawer, và nhiều logic khác mà bạn đã từng làm quen ở callback này từ các bài thực hành trước.  
**onStart():**  
Sau khi gọi đến onCreate(), hệ thống sẽ gọi đến onStart(). Hoặc hệ thống cũng sẽ gọi lại onStart() sau khi gọi onRestart() nếu trước đó nó bị che khuất bởi Activity nào khác (một màn hình khác hoặc một ứng dụng khác) che hoàn toàn và rơi vào onStop().  
Khi hệ thống gọi đến callback này thì Activity được nhìn thấy bởi người dùng và nhưng chưa tương tác được. Bởi đặc tính này mà onStart() ít được dùng đến.  
**onResume()**  
Khi hệ thống gọi đến callback này thì bạn yên tâm rằng người dùng đã nhìn thấy và đã tương tác được với giao diện.  
onResume() được gọi khi Activity được khởi tạo rồi và bước qua onStart() trên kia. Hoặc khi Activity bị một giao diện nào khác che đi một phần (hoặc toàn phần), rồi sau đó quay lại Activity hiện tại. Bạn có thể thấy rằng callback này được gọi rất nhiều lần trong một vòng đời của nó.  
Chính đặc điểm này của onResume() mà bạn có thể tận dụng để quay lại tác vụ mà người dùng đang bị dang dở khi onPause() (được nói đến dưới đây) được gọi.  
Chẳng hạn như bạn đang soạn nội dung cho TourNote, mà có cuộc gọi đến, bạn sẽ lưu tạm nội dung này khi callback onPause(), để rồi khi onResume() được gọi lại sau đó khi người dùng kết thúc cuộc gọi và quay lại TourNote, bạn sẽ khôi phục nội dung đó để người dùng tiếp tục sử dụng TourNote như chưa có bất kỳ gián đoạn nào.  
**onPause()**  
Thông thường nếu có một thành phần nào đó che Activity hiện tại mà người dùng vẫn nhìn thấy Activity đó (nhìn thấy chứ không tương tác được). Chẳng hạn một popup hiện lên trên Activity. Thì onPause() của Activity sẽ được gọi. Sau này khi người dùng quay lại Activity thì onResume() sẽ được gọi.  
Bạn có thể tưởng tượng rằng onPause() cũng sẽ được gọi khá nhiều lần trong một vòng đời Activity. Theo như Google thì onPause() được gọi đến khá nhanh, nếu bạn muốn lưu trữ dữ liệu như mình nói trên kia, thì nên lưu những gì nhanh gọn lẹ thôi. Nếu bạn muốn lưu trữ các dữ liệu nặng, hoặc gọi API kết nối server chỗ này, nhiều khả năng ứng dụng sẽ không kịp thực hiện. Do đó, thay vì làm các thao tác nặng nề ở onPause(), bạn có thể cân nhắc gọi chúng ở onStop().  
**onStop()**  
Như mình có nói. onStop() được gọi khi Activity không còn được nhìn thấy nữa, có thể một màn hình nào khác che lên hoàn toàn, có thể một ứng dụng nào đó vào foreground, hoặc người dùng nhấn nút Home để về màn hình chính.  
Bạn có thể tận dụng onStop() để lưu trữ dữ liệu ứng dụng. Hoặc để giải phóng các tài nguyên đang dùng. Ngưng các API còn đang gọi dang dở.  
Tuy nhiên khi onStop() được gọi không phải là lúc chúng ta cũng nói lời tạm biệt Activity. Như mình đã nói, người dùng hoàn toàn có thể quay lại sử dụng Activity sau đó mà không cần phải khởi động lại Activity, khi này thì phương thức onRestart() và onStart() được gọi kế tiếp nhau.  
**onDestroy()**  
Bạn có thể tận dụng callback này để giải phóng các tài nguyên hệ thống mà ở onStop() bạn chưa gọi đến.  
Vòng đời của một Activity kết thúc ở đây.

Cấu hình của thiết bị đề cập đến các đặc điểm của thiết bị và ứng dụng liên quan đến tài nguyên ứng dụng, những thứ như ngôn ngữ được sử dụng, kích thước màn hình, tính khả dụng của bàn phím và hướng thiết bị. Bây giờ, những đặc điểm này có thể thay đổi trong thời gian chạy. Và khi họ làm như vậy, Android thường sẽ giết hoạt động hiện tại và sau đó khởi động lại nó với các tài nguyên phù hợp cho cấu hình đã thay đổi. Bây giờ, vì các thay đổi cấu hình có thể xảy ra thường xuyên trong khi sử dụng một ứng dụng, bạn muốn đảm bảo rằng mã phương thức vòng đời của bạn nhanh. Xem xét thay đổi định hướng thiết bị, ví dụ. Nếu bạn di chuyển thiết bị từ chế độ dọc sang chế độ nằm ngang và quay lại chế độ dọc, điều đó sẽ khiến hoạt động hiện tại bị hủy và khởi động lại hai lần.

Vì vậy, nếu mã khởi động của bạn chậm, điều đó thực sự sẽ ảnh hưởng đến trải nghiệm của người dùng với ứng dụng của bạn. Vì vậy, để cải thiện tốc độ thay đổi cấu hình nếu bạn cần, Android cho phép bạn thực hiện hai điều. Thứ nhất, khi sự thay đổi đang diễn ra, bạn có thể tạo và lưu một đối tượng Java tùy ý, về cơ bản lưu trữ thông tin trạng thái quan trọng. Và hai, bạn có thể tự xử lý thay đổi cấu hình, tránh toàn bộ chuỗi tắt và khởi động lại tất cả cùng nhau. Chúng ta hãy nói về từng người một.

Vì vậy, để tăng tốc cấu hình lại, tôi đã nói rằng bạn có thể lưu trữ cứng để tính toán lại dữ liệu và lưu trữ nó trong một đối tượng Java. Và một cách để làm điều này là ghi đè phương thức onRetainNonConfigurationInstance, sẽ được gọi một thời gian giữa onStop và onDestroy. Và đối tượng mà bạn xây dựng và trả về trong onRetainNonConfigurationInstance có thể được truy xuất khi hoạt động được tạo lại bằng cách gọi getLastNonConfigurationInstance. Và điều đó thường được thực hiện trong khi gọi phương thức onCreate của hoạt động khi bạn tạo lại hoạt động. Bây giờ, lưu ý rằng các phương thức này đã bị phản đối vì các phương thức khác trong lớp Fragment. Và chúng tôi chưa thảo luận về các mảnh vỡ, nhưng chúng tôi sẽ quay lại với nó trong một bài học sau khi chúng tôi làm.

Bây giờ, điều thứ hai bạn có thể làm để tăng tốc cấu hình lại là tránh toàn bộ chuỗi giết và khởi động lại hoàn toàn cho các thay đổi cấu hình cụ thể. Và để làm điều này, bạn khai báo những thay đổi cụ thể mà hoạt động của bạn sẽ xử lý trong tệp AndroidManifest.xml. Ví dụ: đoạn mã XML này chỉ ra rằng MyActivity sẽ xử lý thủ công các thay đổi về hướng thiết bị và kích thước màn hình và khả năng truy cập bàn phím.

Vì vậy, nếu bạn xử lý một số thay đổi cấu hình theo cách thủ công, thì khi những thay đổi cấu hình đó xảy ra trong thời gian chạy, hoạt động của bạn sẽ nhận được một cuộc gọi đến onConfigurationChanged.

Và như một tham số, phương thức này sẽ nhận được một đối tượng Cấu hình chi tiết cấu hình mới, do đó mã của bạn có thể đọc đối tượng này và thực hiện bất kỳ thay đổi nào cần thiết.

1. **Tuần 3**

**Fragment**

Các fragment đại diện cho một phần của giao diện người dùng của một hoạt động. Các fragment được lưu trữ bởi các activity. Một activity có thể lưu trữ không hoặc nhiều fragment. Và fragment có thể được sử dụng trên 0 hoặc nhiều activity.

Các fragment được lưu trữ bởi các activity. Vì vậy, các fragment phải được tải vào activity và sau đó được hiển thị hoặc loại bỏ,… khi activity thay đổi trạng thái. Nghĩa là vòng đời của một fragment gắn kết liền với vòng đời của activity lưu trữ nó. Tuy nhiên, các fragment có một số vòng đời callback:

Một fragment có thể ở trạng thái chạy hoặc tiếp tục. Và khi ở trạng thái này, fragment có thể nhìn thấy trong hoạt động đang chạy.

Một fragment cũng có thể ở trạng thái tạm dừng và các fragment sẽ bị tạm dừng khi activity lưu trữ chúng vẫn hiển thị nhưng một activity khác được focus.

Một fragment cũng có thể ở trạng thái ngừng. Ở trạng thái này, fragment không nhìn thấy được.

Khi activity lưu trữ của fragment được tạo, một số phương thức của fragment đó có thể được gọi:

+ Khi một fragment được gắn vào activity lưu trữ nó, Android gọi phương thức onAttach ()

+ Tiếp theo, Android gọi đến phương thức onCreate() (không giống như trong Activity.onCreate() không thiết lập UI trong Fragment.onCreate())

+ sau onCreate, Android gọi onCreateView() trên fragment để thiết lập và trả về chế độ xem chứa UI của fragment đó. View của fragment sẽ dược lưu trữ vào activity để nó có thể cài đặt view đó trong phân cấp chế độ xem của activity.

+ sau khi activity lưu trữ fragment được tạo và UI của fragment đã được cài đặt, thì Android gọi đến onActivityCreated()

Khi fragment được gắn vào activity lưu trữ, vòng đời của nó phụ thuộc vào vòng đời của activity mà nó gắn vào.

Vd: Fragment sẽ nhận được một cuộc gọi đến phương thức onStart () khi activity lưu trữ nhận được cuộc gọi onStart(). Tương tự với onResume (), onPause (), onStop ().

Khi activity lưu trữ sắp bị hủy bất kỳ fragment nào mà nó lưu trữ cũng phải ngừng hoạt động. Và, điều này xảy ra theo các bước:

+ View được tạo bởi lệnh gọi onCreateView() sẽ bị tách ra khỏi hoạt động, đoạn này nhận được lệnh gọi onDestroyView () và dọn sạch các tài nguyên được liên kết với chế độ xem.

+ Khi fragment không còn được sử dụng, nó sẽ nhận được một cuộc gọi đến onDestroy () và giải phóng tài nguyên fragment.

+ Fragment này sẽ nhận được một cuộc gọi đến phương thức onDetach () khi không còn gắn với activity lưu trữ của nó và không thể tham chiếu đến hoạt động lưu trữ của đoạn đó.

Có hai cách mà các fragment được thêm vào các hoạt động:

+ Được thêm tĩnh vào hoạt động, đưa chúng vào tệp bố cục của hoạt động, sau đó được sử dụng trong lệnh gọi đến setContentView().

+ Thêm các fragment bằng cách sử dụng FragmentManager.

Tuy nhiên, sau khi fragment được thêm vào, Android sẽ thực hiện cuộc gọi tới onCreateView.

Và trong onCreateView(), fragment có thể sử dụng bố cục XML của riêng nó, tương tự như các hoạt động thực hiện khi chúng gọi setContentView() hoặc chúng có thể lập trình xây dựng giao diện người dùng của chúng. Khi đã xong, UI được xây dựng, onCreateView() phải trả về view. Và view đó cuối cùng sẽ được gắn cho activity lưu trữ và được thêm vào UI của activity lưu trữ.

Bạn cũng có thể thêm các fragment vào một activity mà không cần gán cứng các fragment đó vào layout của activity. Và để làm điều đó, trong khi activtiy đang chạy, bạn phải làm bốn điều sau:

Một, có được một tham chiếu đến FragmentManager.

Hai, bắt đầu một FragmentTransaction.

Ba, thêm một fragment vào activity.

Và bốn, commit FragmentTransaction

…(Xem video để biết thêm chi tiết ☺)

FragmentTransaction cho phép bạn thay đổi linh hoạt UI của ứng dụng và làm cho giao diện người dùng linh hoạt hơn và tận dụng tốt hơn không gian màn hình có sẵn.

…(Xem video để biết thêm chi tiết ☺)

Nếu setRetainInstance() là true, thì khi thay đổi cấu hình xảy ra, Android sẽ tắt activity, nhưng không phá hủy các fragment mà nó lưu trữ. Thay vào đó, nó sẽ lưu trạng thái fragment đó và nó sẽ tách rời fragment khỏi activity. Nó sẽ không hủy fragment đó và vì vậy, nó sẽ không phải tạo lại nó sau khi activity lưu trữ được khởi tạo lại.

…(Xem video để biết thêm chi tiết ☺)

1. **Tuần 4**

View class là một block xây dựng cơ bản đảm nhận mọi thứ có thể thấy được trên màn hình

View Groups là phức hợp của nhiều view để đa dạng các giao diện vs hành cụ thể của view group là adapter views, and layouts

menus and the action bar cho phép ng dùng truy cập nhanh và cụ thể các view

The view class là nơi xây dựng UI components. Views chiếm khoảng không gian hình chữ nhật trên screen. và nó phụ trách vẽ chính nó vs xử lý sự kiện đến nó  
  
buttons, toggle buttons. Checkboxes, the RatingBar, and the AutoCompleteTextView. là các view được xây dựng sẵn  
  
AutoCompleteTextView. This view is first of all a text view. That is, it's a view that displays text. And we've seen lots of .  
  
view xử lý sự kiện. Ví dụ như view nhận sự kiện thì android sẽ sắp xếp lại và vẽ lại view.Một cách phổ biến trong xử lý sự kiện là chúng ta gắn listener vào view.  
Ví dụ như view có " on click" listener interface, có "on click" method. method này được gọi mỗi khi được người dùng click vào  
view còn có định nghĩa long click listener. Sự kiện được gọi khi người dùng nhấn giữ tong một khoảng thời gian  
View còn có định nghĩa focus change listener. Hàm này được được gọi khi view nhận hay mất focus  
Và còn một số listeners interface khác sẽ được gọi khi người dùng nhấn phím vật lý  
view trong một ứng dụng đuọc tổ chức theo cây  
  
Khi ta custom một subclass thì có nhiều hàm chúng ta có thể ghi đè được, ví dụ như giai đoạn 1 khi đo lường lại view. Android sẽ gọi onMeasure trên view của bạn. sau đó nó sẽ tính toán và set lại không gian của nó  
  
Giai đoạn 2:Khi ta bố trí view thì  Android sẽ gọi onLayout method. Hàm này sẽ bố tri 1la5i chính nó và các bố trí của view con của nó.

Ở giai đoạn thứ ba, onDraw method sẽ được gọi và nó sẽ vẽ view lên màn hình người dùng.  
  
Trên thực tế chúng ta sẽ cần view phức hợp (compound view) là tập hợp của nhiều view độc lập, để có được những tính năng đòi hỏi sữ phức tạp. một ví dụ của việc này là radio group.

A radio group là tập hợp của nhiều  check boxes. Ví dụ như giao diện cho phép người dùng chọn trường của tuổi, say under 20, 20 to 34, 35 to 49, and over 50.Để làm được việc này ta ần nhiều text views, với mổi trường tuổi khác nhau ta cần text view khác nhau, và cần đặt 1 check box. Và giờ ta cần bảo đảm chỉ có 1 check box được họn, để làm được điều này, Android có class gọi là ViewGroup. ViewGroups là thành phần vô hình bao gồm nhiều views.  
ViewGrouplà 1 base class cho view containers and layouts.  
Android cung cấp nhiều viewGroups được định nghĩa trước, như là RadioGroup, TimePicker, DatePicker, WebView, MapView, Gallery, and Spinner.

RadioGroup là một ViewGroup bao gồm nhiều check boxes độc lập hay radio buttons. Nên chỉ có một trong số chúng được chọn

ViewGroup tiếp theo là một WebView. WebView là một Viewgroup hiển thị các trang web.

Viewgroup tiếp theo là MapView và như tên cho thấy, MapView là Viewgroup hiển thị bản đồ và cho phép người dùng tương tác với chúng.

Xem xét, ví dụ, một cái nhìn danh sách, có thể được sử dụng để hiển thị một danh sách các số điện thoại để quay số, một danh sách các bài hát để chơi, một danh sách các hình ảnh để chọn làm hình nền của điện thoại, và vân vân. Để cho ViewGroups như xem danh sách để làm việc với tất cả các kiểu dữ liệu khác nhau,

Android cung cấp một lớp con Viewgroup có tên là AdapterView. AdaptorView là các Viewgroup có khung nhìn con và dữ liệu cơ bản, được quản lý không phải bởi chính Viewgroup, mà bởi một lớp khác gọi là Adaptor. Và lớp Adaptor này chịu trách nhiệm quản lý dữ liệu cũng như tạo và cung cấp các khung nhìn của dữ liệu đó cho AdapterView khi Adaptor yêu cầu chúng. Adaptor sau đó chỉ chịu trách nhiệm hiển thị các Chế độ xem dữ liệu.

ListView là một Adaptor hiển thị danh sách có thể cuộn của các mục có thể chọn. Those items are managed by an Adapter called the ListAdapter.

ListView cũng có thể tùy chọn lọc danh sách các mục dựa trên kiểu nhập văn bản, giống như những gì chúng ta đã thấy với TextView tự động hoàn thành.

Bố cục tiếp theo là GridView, GridViews sắp xếp children của chúng trong một lưới có thể cuộn hai chiều.

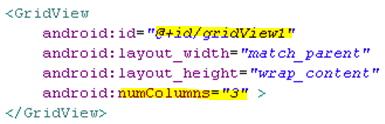
Nếu người dùng sau đó cuộn GridView, một số chế độ xem có thể nhìn thấy, sẽ trở nên vô hình vì họ cuộn khỏi màn hình. Vì vậy, Android sẽ cố gắng sử dụng lại các chế độ xem đó và nó sẽ chuyển một trong các chế độ xem mới đó sang bộ điều hợp để chuyển sang phương thức GetView. Và sau đó, bạn sẽ chỉ sử dụng chế độ xem đó và đặt lại bất kỳ trường nào bạn cần cho mục dữ liệu hiện tại của mình.

Bài tập này bạn sẽ làm quen với control Gridview, cũng tương tự như ListView. Gridview cũng dựa vào Datasource và ArrayAdapter. ListView bạn làm như thế nào thì Gridview y xì.

– Điểm khác nhau là GridView có thiết lập số cột. Dữ liệu luôn đưa vào dưới dạng hình ống (mảng, list một chiều), nhưng dựa vào số cột ta thiết lập mà nó tự động ngắt hàng, xem hình minh họa:

– Bạn thấy đấy, Tôi có thể hiển thị Text hoặc hình ảnh vào GridView.

– Bạn có thể thiết lập số cột cho GridView theo hình dưới đây:

– Nếu bạn thiết lập **android:numColumns=”3″**,  Tức là Gridview sẽ ngắt dòng khi đủ 3 phần tử, nó chỉ khác chỗ này, còn việc đưa dữ liệu lên như thế nào thì nó y xì như làm với ListView.

**– Ví dụ 1: Hiển thị văn bản lên GridView (xem hình Tôi đánh số 1):**

– Bạn tạo một Android Project tên tùy thích, ở đây Tôi đặt tên: **Vidu\_Gridview\_Text**

– Đây là **activity\_main.xml** cho ứng dụng:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | <LinearLayout xmlns:android="<http://schemas.android.com/apk/res/android>"  xmlns:tools="<http://schemas.android.com/tools>"  android:id="@+id/LinearLayout1"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:orientation="vertical"  tools:context=".MainActivity" >  <TextView  android:id="@+id/selection"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:background="#8A9D6F"  android:hint="Slected here" />  <GridView  android:id="@+id/gridView1"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:numColumns="3" >  </GridView>  </LinearLayout> |

– Bạn xem dòng 15 là id của GridView, Tôi để mặc định **gridView1**

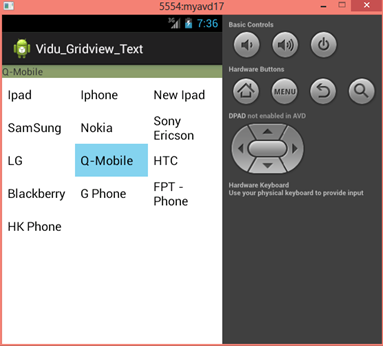
– Dòng 18 có thuộc tính: **android:numColumns= “3”**, tức là dữ liệu sẽ được hiển thị trong Gridview với định dạng 3 cột.

**– Tiếp theo bạn xem MainActivity.java:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45 | package tranduythanh.com;  import android.os.Bundle;  import android.app.Activity;  import android.view.View;  import android.widget.AdapterView;  import android.widget.ArrayAdapter;  import android.widget.GridView;  import android.widget.TextView;    public class MainActivity extends Activity {    //Dùng mảng 1 chiều hoặc ArrayList để lưu một số dữ liệu  String arr[]={"Ipad","Iphone","New Ipad",  "SamSung","Nokia","Sony Ericson",  "LG","Q-Mobile","HTC","Blackberry",  "G Phone","FPT - Phone","HK Phone"  };  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_main);  //Tối tượng này dùng để hiển thị phần tử được chọn trong GridView  final TextView selection=(TextView)  findViewById(R.id.selection);  final GridView gv=(GridView) findViewById(R.id.gridView1);  //Gán DataSource vào ArrayAdapter  ArrayAdapter<String>da=new ArrayAdapter<String>  (  this,  android.R.layout.simple\_list\_item\_1,  arr  );  //gán Datasource vào GridView  gv.setAdapter(da);  //Thiết lập sự kiện cho GridView,  gv.setOnItemClickListener(new AdapterView  .OnItemClickListener() {  public void onItemClick(AdapterView<?> arg0,  View arg1, int arg2,  long arg3) {  //Hiển thị phần tử được chọn trong GridView lên TextView  selection.setText(arr[arg2]);  }  });  }  } |

– Bạn thấy đó, cách sử dụng GridView là y xì như ListView, nên nếu như bạn đã rành về ListView rồi thì GridView hiển nhiên bạn cũng làm tốt.

– Thực thi chương trình bạn sẽ thấy như hình bên dưới:

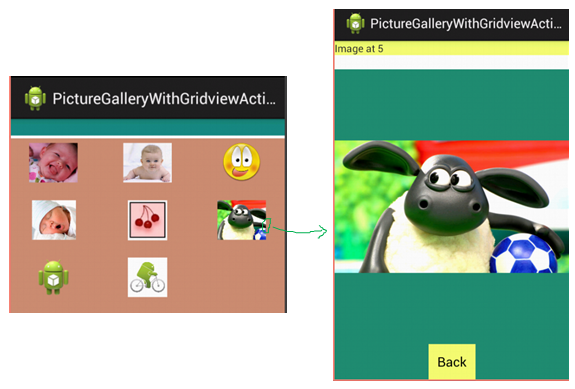


– Xem coding đầy đủ ở đây: <http://www.mediafire.com/?v3ww92ys1s5jth0>

**– Ví dụ 2 sẽ phức tạp hơn, hiển thị danh sách hình ảnh có sẵn lên GridView, mỗi lần chọn vào hình ảnh nào thì sẽ hiển thị chi tiết ảnh đó trên một màn hình mới:**

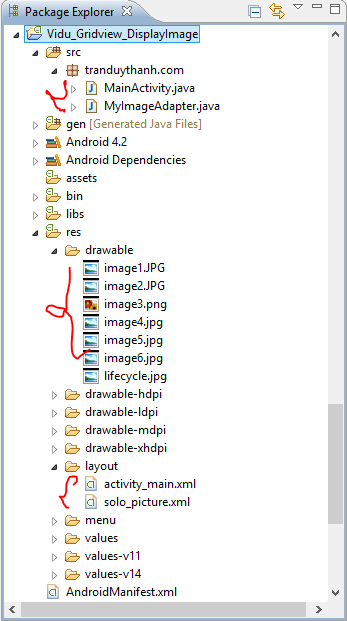
– Có thể Demo này đã có nhiều website và Ebooks làm rồi, ở đây Tôi cũng muốn demo lại cho các bạn.

– Bạn xem giao diện của ứng dụng trước:

– Bên trái là màn hình chính cho phép hiển thị danh sách hình ảnh vào GridView, mỗi lần chọn vào từng hình trong GridView thì sẽ mở hình được chọn đó vào một màn hình mới (ví dụ như khi chọn hình chú Cừu thì nó sẽ hiển thị ra như màn hình bên phải ), nhấn nút Back để trở về màn hình chính.

– Ở đây có một sự khác biệt lớn đó là chúng ta **chỉ sử dụng 1 MainActivity**, không hề tạo thêm bất kỳ một Activity nào khác, chúng ta chỉ thay đổi Layout mà thôi. Nên nó cũng là điểm nhấn của ứng dụng.

-Hãy tạo một Android Project tên: **Vidu\_Gridview\_DisplayImage** và xem cấu trúc của chương trình**:**

****-Layout sẽ có 2 cái: **activity\_main.xml** là của màn hình chính dùng để hiển thị danh sách hình ảnh. **solo\_picture.xml** là dùng để hiển thị từng hình riêng lẻ.

– Tạo thêm thư mục **drawble** và kéo thả một số hình ảnh vào.

–  Phần class có 2 class: MainActivity và MyImageAdapter kế thừa từ BaseAdapter.

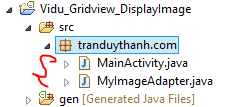
– Bây giờ ta đi vào chi tiết của từng cái:

**– activity\_main.xml:**

****– Bạn có thể nhìn vào hình trên để làm hoặc tải coding mẫu ở bên dưới.

**-solo\_picture.xml:**

****– Bây giờ ta vào các class xử lý nghiệp vụ:

**– Thứ nhất là class MyImageAdapter:**

+ class này sẽ kế thừa từ BaseAdapter, và dùng để hiển thị từng hình ảnh riêng lẻ:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55 | package tranduythanh.com;    import android.content.Context;  import android.view.View;  import android.view.ViewGroup;  import android.widget.BaseAdapter;  import android.widget.GridView;  import android.widget.ImageView;  /\*\*  \* Class dùng để hiển thị từng hình ảnh riêng lẻ  \* @author drthanh  \*  \*/  public class MyImageAdapter extends BaseAdapter {  private Context mContext;  private Integer []mThumbIds;  public MyImageAdapter(Context c){  mContext=c;  }  public MyImageAdapter(Context c,Integer []arrIds){  mContext=c;  mThumbIds=arrIds;  }  public int getCount()  {  return mThumbIds.length;  }  public Object getItem(int arg0)  {  return null;  }  public long getItemId(int arg0)  {  return 0;  }  /\*\*  \* Cần override lại hàm này để hiển thị hình ảnh  \*/  public View getView(int arg0, View convertView, ViewGroup arg2) {  ImageView imgView;  if(convertView==null){  imgView=new ImageView(mContext);  //can chỉnh lại hình cho đẹp  imgView.setLayoutParams(new GridView.LayoutParams(85, 85));  imgView.setScaleType(ImageView.ScaleType.CENTER\_CROP);  imgView.setPadding(8, 8, 8, 8);  }else{  imgView=(ImageView) convertView;  }  //lấy đúng vị trí hình ảnh được chọn  //gán lại ImageResource  imgView.setImageResource(mThumbIds[arg0]);  return imgView;  }  } |

**– Thứ hai là class MainActivity:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71 | package tranduythanh.com;  import android.os.Bundle;  import android.app.Activity;  import android.view.View;  import android.widget.AdapterView;  import android.widget.Button;  import android.widget.GridView;  import android.widget.ImageView;  import android.widget.TextView;  import android.widget.AdapterView.OnItemClickListener;    public class MainActivity extends Activity  implements OnItemClickListener  {  TextView tvMsg;  GridView gv;  TextView tvSoloMsg;  //mảng lưu danh sách các id hình ảnh có sẵn  Integer[]mThumbIds;  //Adapter cho GridView  MyImageAdapter adapter=null;  //Vì không tạo thêm 1 Activity nên lấy luôn 2 Id ở bên solo\_picture.xml  ImageView ivSoloPicture;  Button btnBack;  //Lưu Bundle backup cho MainActivity  Bundle myBackupBundle;  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  //Lưu savedInstanceState trước vào myBackupBundle  myBackupBundle=savedInstanceState;  setContentView(R.layout.activity\_main);  tvMsg=(TextView) findViewById(R.id.tvMsg);  //gán mảng các Id hình ảnh cho mThumbIds  mThumbIds=new Integer[]{R.drawable.image1,R.drawable.image2,  R.drawable.image3,R.drawable.image4,  R.drawable.image5,R.drawable.image6,  R.drawable.ic\_launcher,R.drawable.lifecycle};  gv=(GridView) findViewById(R.id.gridView1);  //thiết lập Datasource cho Adapter  adapter=new MyImageAdapter(this, mThumbIds);  //gán Adapter vào Gridview  gv.setAdapter(adapter);  //thiết lập sự kiện để mở từng hình ảnh chitiết  gv.setOnItemClickListener(this);  }  public void onItemClick(AdapterView<?> arg0,  View arg1, int arg2, long arg3) {  //gọi hàm xem hình ảnh chi tiết tại ví trí thứ arg2  showdetail(arg2);  }  public void showdetail(int posistion)  {  //Không mở Activity mới mà chỉ thiết lập lại Layout  setContentView(R.layout.solo\_picture);  //Vừa gọi hàm trên thì màn hình sẽ thay đổi qua cái mới  //ta lấy các control trong layout mới này  tvSoloMsg=(TextView) findViewById(R.id.tvSoloMsg);  tvSoloMsg.setText("Image at "+posistion);  ivSoloPicture=(ImageView) findViewById(R.id.imgSolo);  //thiết lập hình ảnh đang chọn lên ImageView mới  ivSoloPicture.setImageResource(mThumbIds[posistion]);  btnBack=(Button) findViewById(R.id.btnBack);  //Thiết lập sự kiện click Back để phục hồi lại MainActivity  //bằng cách gọi lại onCreate(myBackupBundle);  btnBack.setOnClickListener(new View.OnClickListener() {  public void onClick(View arg0) {  onCreate(myBackupBundle);  }  });  }  } |

– Khởi chạy ứng dụng bạn sẽ có kết quả như mong muốn.

– Bài tập này bạn đã biết cách sử dụng GridView, biết cách đưa hình ảnh vào GridView, đặc biệt biết thêm một kỹ thuật mới đó là thay đổi Layout để đổi màn hình không cần chạy Activity

Activity có hỗ trợ menu, menu có thể được trình bày cho người dùng theo nhiều cách khác nhau, nhưng ý tưởng cơ bản là các menu cung cấp cho người dùng một cách nhanh chóng để truy cập các chức năng quan trọng. Vì vậy, các hoạt động có thể thêm các mục vào một menu và chúng có thể phản hồi khi người dùng gọi mục menu, nói bằng cách nhấp vào nó.

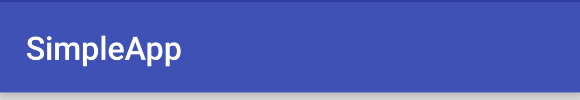
11:02  
  
Thanh ActionBar đã được thêm vào trong Android 3.0 và loại bắt chước Thanh ứng dụng, mà thường thấy trong các ứng dụng máy tính để bàn, ứng dụng máy tính để bàn. ý tưởng cơ bản đằng sau ActionBar là thay vì ẩn các hành động đằng sau menu bật lên truyền thống.

thì cung cấp cho người dùng quyền truy cập nhanh vào các hành động mà họ có thể sẽ sử dụng.

Activity và fragment, tạo và quản lý giao diện người dùng ứng dụng. cho phép bạn kết hợp các hoạt động lại với nhau theo kiểu mô đun để tạo các ứng dụng phức tạp hơn.

Cơ bản sử dụng về ActionBar

Mọi ứng dụng, nếu không có thiết lập gì đặc biệt (Bằng theme) thì sẽ xuất hiện một ActionBar như hình:



Đổi tiêu đề ActionBar

Tiêu đề của ActionBar hiện thị trên đỉnh màn hình được điều khiển bởi file AndroidManifest.xml, trong nút Activity gán tiêu đề vào thuộc tính android:label

<application

   android:allowBackup="true"

   android:icon="@mipmap/ic\_launcher"

   android:label="@string/app\_name"

   android:roundIcon="@mipmap/ic\_launcher\_round"

   android:supportsRtl="true"

   android:theme="@style/AppTheme">

   <activity

       android:name=".MainActivity"

       android:label="@string/myactivityname" >

       <intent-filter>

           <action android:name="android.intent.action.MAIN" />

           <category android:name="android.intent.category.LAUNCHER" />

       </intent-filter>

   </activity>

</application>

Trong đoạn mã trên, Activity có tên *MainActivity* được thiết lập tiêu đề bằng chuỗi *"@string/myactivityname"*

Nếu Activity không có thiết lập android:label thì nó lấy giá trị mặc định từ android:label của nút application

Thay đổi tiêu đề bằng mã khi chạy

Bạn có thể thay đổi tiêu đề, icon của ActionBar bất kỳ lúc nào muốn bằng viết mã Java, trong Activity gọi getSupportActionBar() để lấy đối tượng ActionBar sau đó bạn truy cập các thuộc tính, thiết lập các thuộc tính để thay đổi.

Tham khảo đoạn mã sau:

ActionBar actionBar = getSupportActionBar();

actionBar.setTitle("TIÊU ĐỀ ACTIVITY"); *//Thiết lập tiêu đề nếu muốn*

String title = actionBar.getTitle().toString(); *//Lấy tiêu đề nếu muốn*

actionBar.hide(); *//Ẩn ActionBar nếu muốn*

Thiết lập hiện thị ICON trên ActionBar Android

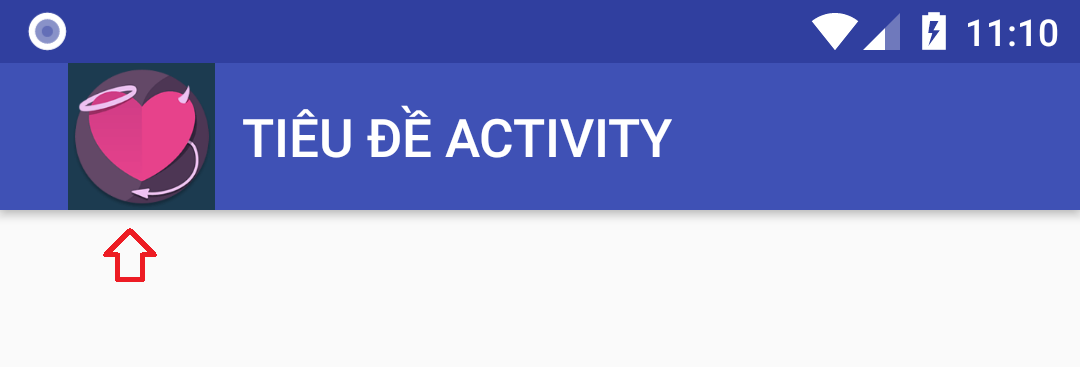
Tham khảo đoạn mã sau:

ActionBar actionBar = getSupportActionBar();

actionBar.setDisplayShowHomeEnabled(true);

actionBar.setLogo(R.mipmap.ic\_launcher);    //Icon muốn hiện thị

actionBar.setDisplayUseLogoEnabled(true);



Nhiều trường hợp muốn co kéo icon vừa với ActionBar, có thể tham khỏa mã sau

//Lấy chiều cao của ActionBar

TypedArray styledAttributes =

   getTheme().obtainStyledAttributes(new int[] { android.R.attr.actionBarSize });

int actionBarSize = (int) styledAttributes.getDimension(0, 0);

styledAttributes.recycle();

//Tạo Drawable mới bằng cách thu/phóng

Drawable drawable= getResources().getDrawable(R.drawable.iconsample);

Bitmap bitmap = ((BitmapDrawable) drawable).getBitmap();

Drawable newdrawable = new BitmapDrawable(getResources(),

       Bitmap.createScaledBitmap(bitmap, actionBarSize,  actionBarSize, true));

getSupportActionBar().setDisplayShowHomeEnabled(true);

getSupportActionBar().setLogo(newdrawable);

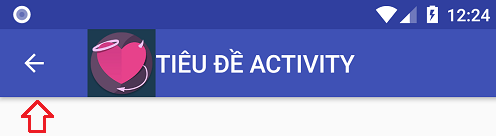
getSupportActionBar().setDisplayUseLogoEnabled(true);

Thêm các Action vào ActionBar

Tùy biến hiện thị biểu tượng bấm quay trở lại

Mặc định nếu bạn muốn hiện thị biểu tượng có hình mũi tên như là nút bấm quay trở lại cửa sổ trước, hãy dùng đoạn code sau:

getSupportActionBar().setDisplayHomeAsUpEnabled(true);



Bạn cũng có thể thay đổi hình mũi tên thành biểu tượng khác

Drawable drawable= getResources().getDrawable(R.drawable.myicon);

//...

getSupportActionBar().setDisplayHomeAsUpEnabled(true);

getSupportActionBar().setHomeAsUpIndicator(newdrawable);



Bắt sự kiện khi bấm vào nút mũi tên quay lại

Các nút bấm Action và kể các nút mũi tên trên ActionBar khi bấm vào sẽ gọi phương thức onOptionsItemSelected(MenuItem item) của Activity, nên muốn bắt các sự kiện khi bấm vào các loại nút bấm này bạn cần quá tải (overrided) phương thức trên, trong đó item chính là đối tượng biểu diễn nút bẩm, thông qua việc xác định ID của item sẽ biết nút nào được bấm.

Lưu ý riêng nút mũi tên bấm quay trở lại ở trên, có ID cố định là: android.R.id.home

Xem đoạn mã sau, khi bấm vào nút quay lại, sẽ tương ứng bấm vào nút back cứng của điện thoại

@Override

public boolean onOptionsItemSelected(MenuItem item) {

   switch (item.getItemId())

   {

       case android.R.id.home:

           onBackPressed();

           return true;

       default:break;

   }

   return super.onOptionsItemSelected(item);

}

Thêm menu vào ActionBar

Để nạp một menu vào ActionBar của Activity, bạn chỉ việc quá tải phương thức onCreateOptionsMenu, ví dụ nạp menu định nghĩa trong XML: R.menu.mymenu

@Override

public boolean onCreateOptionsMenu(Menu menu) {

   getMenuInflater().inflate(R.menu.mymenu, menu);

   return true;

}

Và nội dung \res\menu\mymenu.xml có dạng:

<?xml version="1.0" encoding="utf-8"?>

<menu xmlns:app="http://schemas.android.com/apk/res-auto"

   xmlns:android="http://schemas.android.com/apk/res/android">

   <item

       android:id="@+id/menu1"

       android:icon="@android:drawable/ic\_menu\_gallery"

       android:title="Menu 1"

       app:showAsAction="ifRoom" />

   <item

       android:id="@+id/menu2"

       android:icon="@android:drawable/ic\_lock\_lock"

       android:title="Menu 2"

       app:showAsAction="always" />

   <item

       android:id="@+id/menu3"

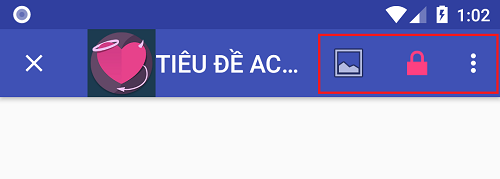
       android:checkable="false"

       android:icon="@android:drawable/ic\_lock\_silent\_mode"

       android:title="Menu3"

       app:showAsAction="collapseActionView" />

</menu>



Tương tự như trên, để bắt sự kiện khi bấm vào các menu này bạn chỉ việc chỉnh lại onOptionsItemSelected thêm các kiểm tra cho menu có id là menu1, menu2 ...

@Override

public boolean onOptionsItemSelected(MenuItem item) {

   switch (item.getItemId())

   {

       case android.R.id.home:

           onBackPressed();

           return true;

       case R.id.menu1:

           //code xử lý khi bấm menu1

           break;

       case R.id.menu2:

           //code xử lý khi bấm menu2

           break;

       case R.id.menu3:

           //code xử lý khi bấm menu3

           break;

       default:break;

   }

   return super.onOptionsItemSelected(item);

}

Toolbar trong Android

Toolbar được giới thiểu từ API 21, nó như là mở rộng của ActionBar, giúp cho việc tùy biến dễ dàng hơn ActionBar. Trong thư viện AppCompat, Toolbar được triển khai ở lớp android.support.v7.widget.Toolbar

Khác biệt giữa Toolbar và ActionBar đó là với Toolbar nó là một View (ViewGroup), nên nó có thể khai báo trong Layout, nó có thể chứa các View khác để bạn thoải mái tùy biến, nó có thể nằm ở vị trí tùy biến khác nhau trong Layout, có thể thực hiện các Animate dễ dàng, có thể chèn nhiều Toolbar trong một Active

Sử dụng Toolbar như ActionBar

Đầu tiên bạn cần đảm bảo tích hợp thư viện AppCompat-v7 vào build.gradle

dependencies {

   ....

   implementation 'com.android.support:appcompat-v7:26.1.0'

}

Tiếp theo cần hủy ActionBar bằng cách thay đổi thông số Theme có:

<item name="windowActionBar">false</item>

Ví dụ sử dụng trực tiếp Theme có tên Theme.AppCompat.Light.NoActionBar để xây dựng AppTheme đã hủy ActionBar

<resources>

   <!-- Base application theme. -->

   <style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">

       <!-- Customize your theme here. -->

   </style>

</resources>

Từ lúc này trong Layout của Activity bạn có thể thêm Toolbar, ví dụ:

<?xml version="1.0" encoding="utf-8"?>

<android.support.constraint.ConstraintLayout

   xmlns:android="http://schemas.android.com/apk/res/android"

   xmlns:app="http://schemas.android.com/apk/res-auto"

   xmlns:tools="http://schemas.android.com/tools"

   android:layout\_width="match\_parent"

   android:layout\_height="match\_parent"

   android:fitsSystemWindows="true"

   tools:context="net.xuanthulab.toolbar.MainActivity">

   <android.support.v7.widget.Toolbar

       android:id="@+id/toolbar"

       android:minHeight="?attr/actionBarSize"

       android:layout\_width="match\_parent"

       android:layout\_height="wrap\_content"

       app:titleTextColor="@android:color/white"

       android:background="?attr/colorPrimary">

   </android.support.v7.widget.Toolbar>

</android.support.constraint.ConstraintLayout>

Trong đoạn mã trên bạn cũng lưu ý android:fitsSystemWindows="true" để layout tính toán chính xác chiều cao khi đã hủy ActionBar.

Giờ thì Toolbar đã chạy với vai trò như là một View, để Toolbar này có chức năng như ActionBar thì bạn thể thêm đoạn mã như sau:

public class MainActivity extends AppCompatActivity {

   @Override

   protected void onCreate(Bundle savedInstanceState) {

       super.onCreate(savedInstanceState);

       setContentView(R.layout.activity\_main);

       Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);

       setSupportActionBar(toolbar);

       ActionBar actionBar = getSupportActionBar();

       //Toobar đã như ActionBar

   }

   @Override

   public boolean onCreateOptionsMenu(Menu menu) {

       getMenuInflater().inflate(R.menu.mymenu, menu);

       return true;

   }

   @Override

   public boolean onOptionsItemSelected(MenuItem item) {

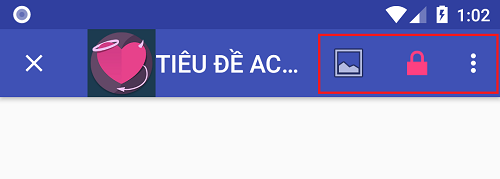
       //..

       return super.onOptionsItemSelected(item);

   }

}

Kết quả Toolbar đã giống với ActionBar



Tùy biến Style của Toolbar

Style của Toolbar có thể chỉ định và tùy biến với các thuộc tính android:theme, app:titleTextAppearance, app:popupTheme,

<android.support.v7.widget.Toolbar

   android:id="@+id/toolbar"

   android:layout\_width="match\_parent"

   android:layout\_height="wrap\_content"

   android:minHeight="?attr/actionBarSize"

   android:background="?attr/colorPrimary"

   android:theme="@style/ToolbarTheme"

   app:titleTextAppearance="@style/Toolbar.TitleText"

   app:popupTheme="@style/ThemeOverlay.AppCompat.Light"

/>

ToolbarTheme, Toolbar.TitleText soạn thảo trong res/values/styles.xml, ví dụ:

<resources>

   <!-- Base application theme. -->

   <style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">

       <!-- Customize your theme here. -->

       <item name="colorPrimary">@color/colorPrimary</item>

       <item name="colorPrimaryDark">@color/colorPrimaryDark</item>

       <item name="colorAccent">@color/colorAccent</item>

   </style>

   <style name="ToolbarTheme" parent="@style/ThemeOverlay.AppCompat.Dark.ActionBar">

       <!-- android:textColorPrimary: Màu chữ Tiêu đề  -->

       <item name="android:textColorPrimary">@android:color/holo\_blue\_light</item>

       <!-- actionMenuTextColor: Màu chữ Menu  -->

       <item name="actionMenuTextColor">@android:color/holo\_green\_light</item>

       <!-- Tints -->

       <item name="colorAccent">#a13636</item>

       <!-- Màu áp dụng cho các View ở trạng thái bình thường -->

       <item name="colorControlNormal">#d9cfcf</item>

       <!-- Màu View đang hoạt động -->

       <item name="colorControlActivated">#934848</item>

       <!-- Highlight (i.e ripples or list selectors) -->

       <item name="colorControlHighlight">#111</item>

   </style>

   <style name="Toolbar.TitleText"

           parent="TextAppearance.Widget.AppCompat.Toolbar.Title">

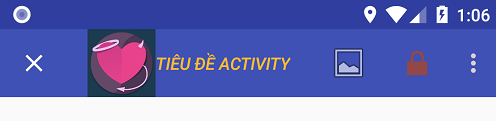
       <item name="android:textSize">21sp</item>

       <item name="android:textStyle">italic</item>

       <item name="android:textColor">@android:color/holo\_orange\_light</item>

   </style>

</resources>



Thiết lập hiện thị Icon ứng dụng

Làm tương tự như ActionBar thông thường, ngoài ra có thêm thiết lập vị trí hiện thị Icon ứng dụng

<android.support.v7.widget.Toolbar

     android:id="@+id/toolbar"

     app:contentInsetLeft="0dp"

     app:contentInsetStart="0dp"

     ...

     >

</android.support.v7.widget.Toolbar>

Tùy biến tiêu đề bằng chèn View

Ví dụ dùng TextView để thể hiện Title của Toolbar

<pre class="prettyprint">&lt;android.support.v7.widget.Toolbar

   android:id="@+id/toolbar"

   android:minHeight="?attr/actionBarSize"

   android:layout\_width="match\_parent"

   android:layout\_height="wrap\_content"

   app:titleTextColor="@android:color/white"

   android:background="?attr/colorPrimary"&gt;

    &lt;TextView

       android:id="@+id/toolbar\_title"

       android:layout\_width="wrap\_content"

       android:layout\_height="wrap\_content"

       android:text="Toolbar Title"

       android:textColor="@android:color/white"

       style="@style/TextAppearance.AppCompat.Widget.ActionBar.Title"

       android:layout\_gravity="center"

    /&gt;

&lt;/android.support.v7.widget.Toolbar&gt; </pre>

Trong code Java thực hiện

/\* Trong Activity - OnCreate \*/

Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);

setSupportActionBar(toolbar);

// Loại bỏ tiểu đề mặc định

getSupportActionBar().setDisplayShowTitleEnabled(false);

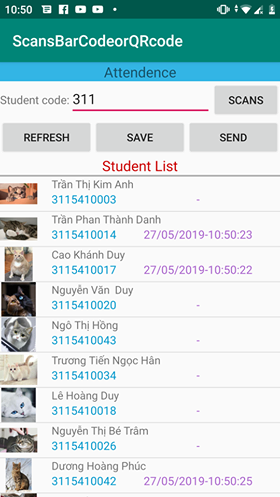
// Lấy đối tượng tiêu đề tùy biến

TextView mTitle = (TextView) toolbar.findViewById(R.id.toolbar\_title);

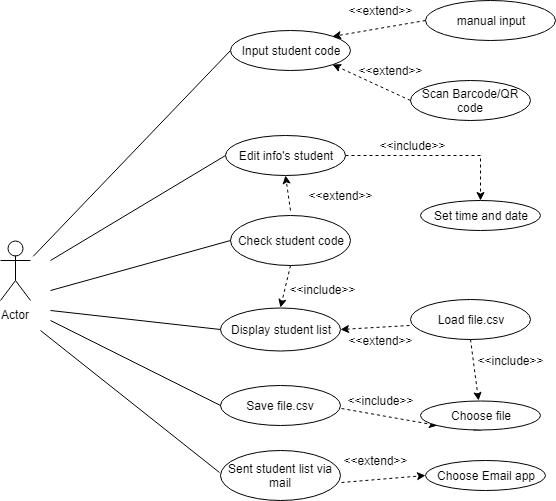
**V. ATTENDANCE**

1. UI

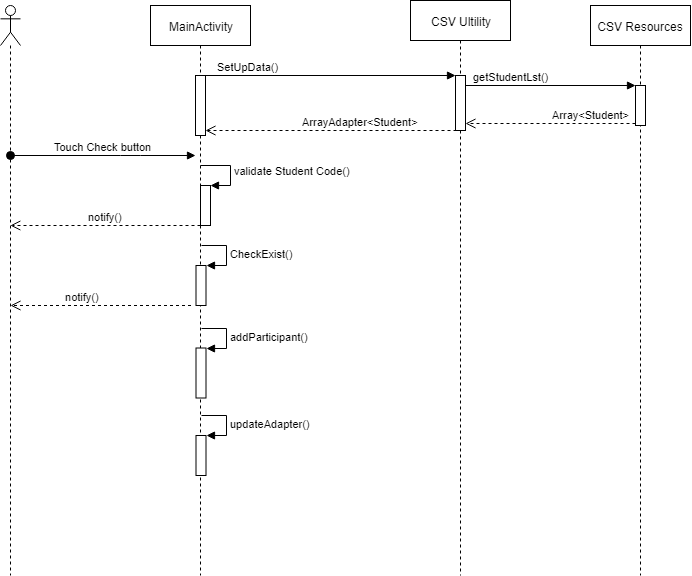
%3CmxGraphModel%3E%3Croot%3E%3CmxCell%20id%3D%220%22%2F%3E%3CmxCell%20id%3D%221%22%20parent%3D%220%22%2F%3E%3CmxCell%20id%3D%222%22%20value%3D%22%22%20style%3D%22endArrow%3Dclassic%3Bhtml%3D1%3BexitX%3D-0.01%3BexitY%3D0.671%3BexitDx%3D0%3BexitDy%3D0%3Bdashed%3D1%3BexitPerimeter%3D0%3BentryX%3D1%3BentryY%3D0.5%3BentryDx%3D0%3BentryDy%3D0%3B%22%20edge%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20width%3D%2250%22%20height%3D%2250%22%20relative%3D%221%22%20as%3D%22geometry%22%3E%3CmxPoint%20x%3D%22438.8333333333335%22%20y%3D%22495.33333333333337%22%20as%3D%22sourcePoint%22%2F%3E%3CmxPoint%20x%3D%22369.66666666666674%22%20y%3D%22498.66666666666674%22%20as%3D%22targetPoint%22%2F%3E%3C%2FmxGeometry%3E%3C%2FmxCell%3E%3C%2Froot%3E%3C%2FmxGraphModel%3E

%3CmxGraphModel%3E%3Croot%3E%3CmxCell%20id%3D%220%22%2F%3E%3CmxCell%20id%3D%221%22%20parent%3D%220%22%2F%3E%3CmxCell%20id%3D%222%22%20value%3D%22%22%20style%3D%22endArrow%3Dclassic%3Bhtml%3D1%3BexitX%3D-0.01%3BexitY%3D0.671%3BexitDx%3D0%3BexitDy%3D0%3Bdashed%3D1%3BexitPerimeter%3D0%3BentryX%3D1%3BentryY%3D0.5%3BentryDx%3D0%3BentryDy%3D0%3B%22%20edge%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20width%3D%2250%22%20height%3D%2250%22%20relative%3D%221%22%20as%3D%22geometry%22%3E%3CmxPoint%20x%3D%22438.8333333333335%22%20y%3D%22495.33333333333337%22%20as%3D%22sourcePoint%22%2F%3E%3CmxPoint%20x%3D%22369.66666666666674%22%20y%3D%22498.66666666666674%22%20as%3D%22targetPoint%22%2F%3E%3C%2FmxGeometry%3E%3C%2FmxCell%3E%3C%2Froot%3E%3C%2FmxGraphModel%3E

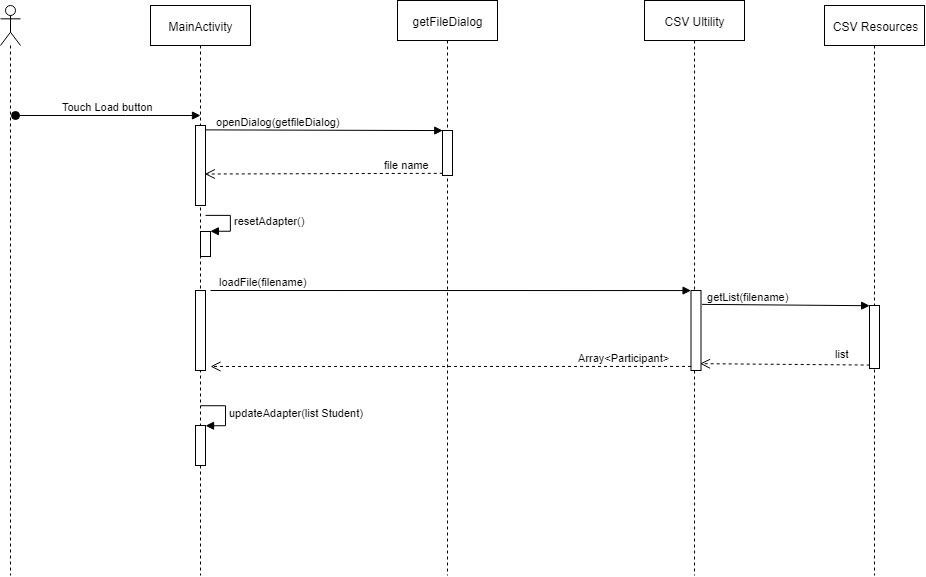
1. Use case



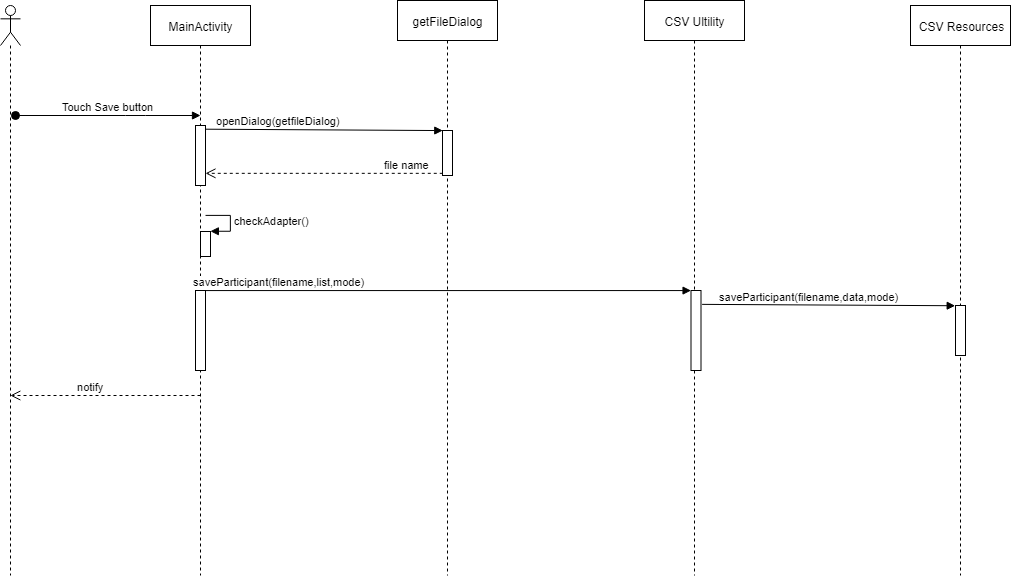
1. Sequence
2. Check student code



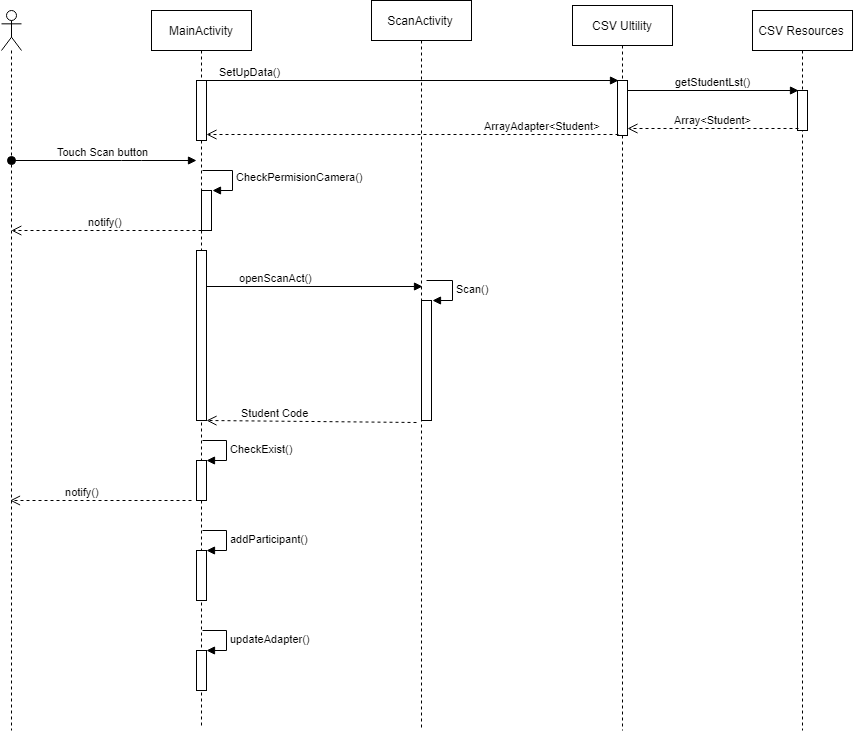
1. Load student list



1. Save student list



1. Scan student code



1. Sent mail

